Graphs

Joonho Kim

Instructions

- There will be questions on these slides. Please have a clean piece of paper to write your answers. Write your name on the top right corner for our record. At the end of lecture, we will collect these pieces of paper for your participation grade. Scribes should get ready to scribe.

Announcements

- 5 scribes please
  - Write names on white board to remember
- Homework 7 Pattern Match Due Wed
- Homework 8 Graphs is Due next Thu
- Class Observation Thursday

Last Time…

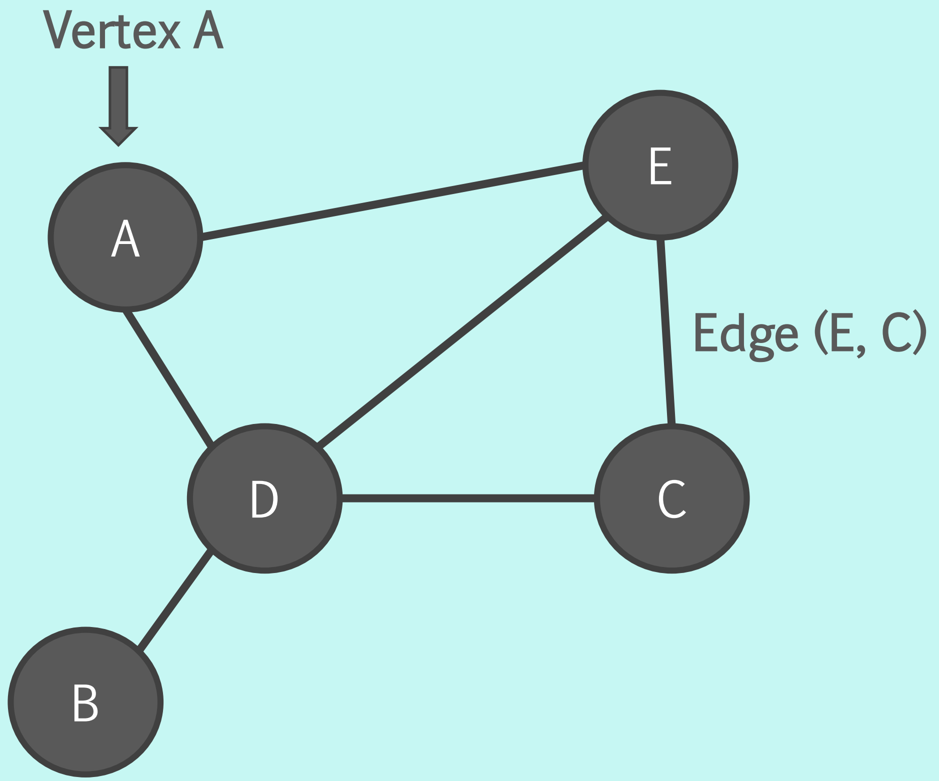
- Take 5 min to write down a the following:
  - Post-Exam
    - On a scale from 1-10, how well prepared were you for the exam.
    - How much did you study and what studying methods did you employ?
      - Did that work?
    - How fair was exam 2.
  - HW 7
    - How far are you on HW 7.

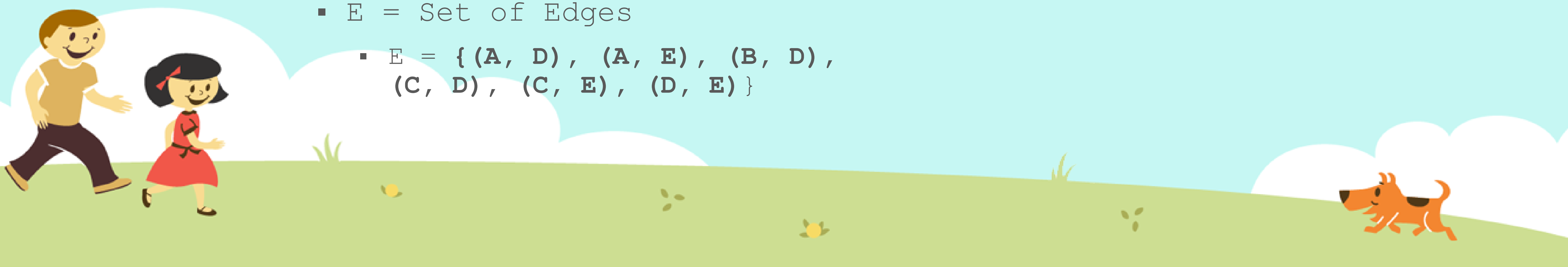
Data Structure Representations

- Sequences: Linked List
- Hierarchy / Decision Making: Trees
- Key-Value association: Maps
- What about Entity and Relationships?
  - Spatial: Airports and Flights
  - Social: People and Friendships
  - Geometric: Points and Edges (*What I work on*)

What is a Graph?

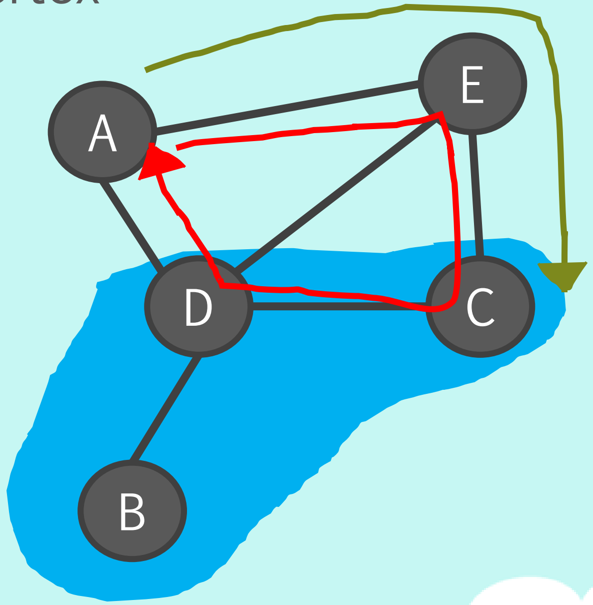
- A **Graph** is a set of Vertices and Edges.
- A **Vertex** represents an entity.
- An **Edge** is a connection between two Vertices.
- Graph notation: `G = (V, E)`
  - `V = Set of Vertices`
    - `V = {A, B, C, D, E}`
  - `E = Set of Edges`
    - `E = {(A, D), (A, E), (B, D), (C, D), (C, E), (D, E)}`





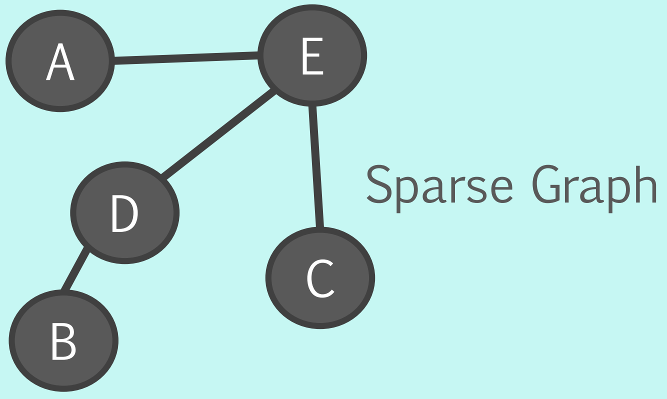
Graph Terminology

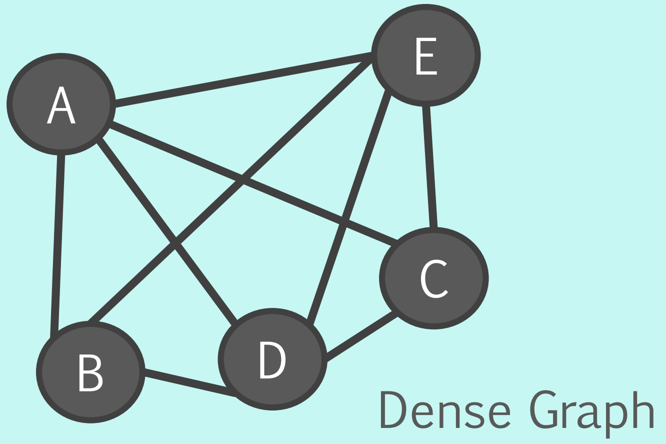
- **Path:** A sequence of Vertices $v_0, v_1, v_2, v_3 \ldots$ where each vertex is connected to the next vertex. Length of path is number of edges.
  - e.g. A, E, C
- **Cycle:** A path that starts and ends with the same vertex
  - e.g. A, E, C, D, A
- **Sub Graph:** A subset of vertices and edges
  - Sub Graph SG
    - `V′ = {C, D, E}`
    - `E′ = {(B, D), (C, D)}`

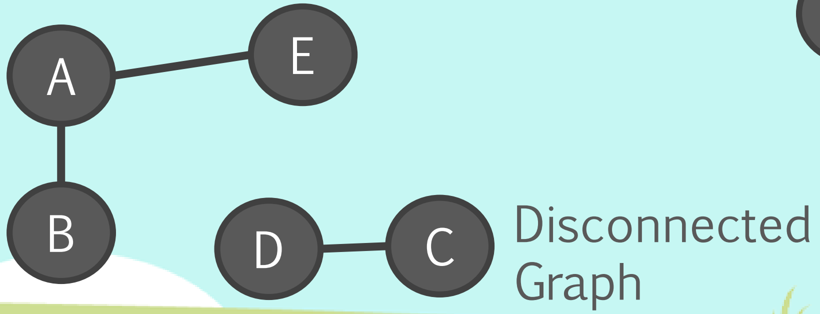


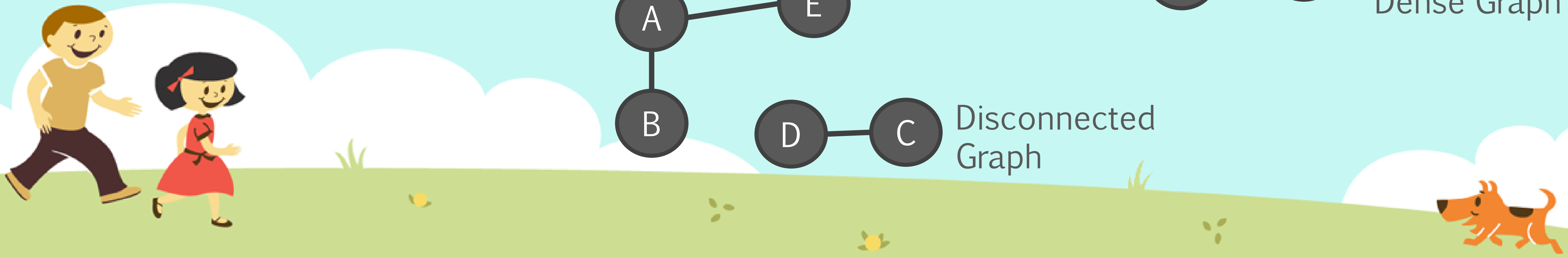
Types of Graphs

- **Sparse:** graph has few edges compared to vertices. (E ~ V)
- **Dense:** graph has many edges compared to vertices. (E ~ V^2)
- **Connected:** All pairs of vertices are connected by a path.
- **Disconnected:** Not Connected Graph.



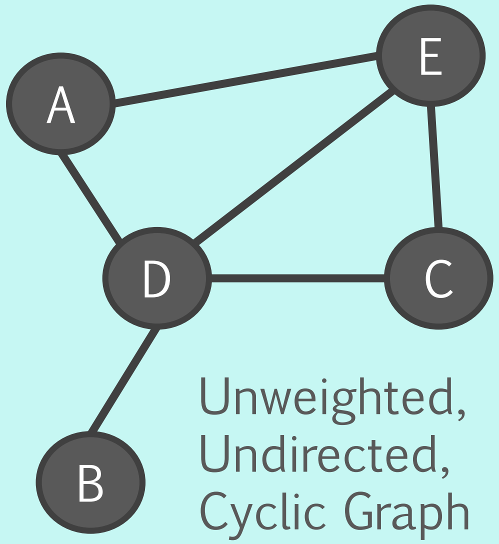




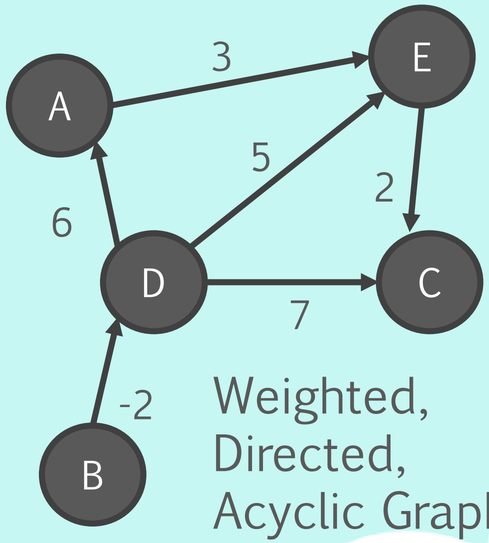


Types of Graphs cont.

- **Weighted** Edges have a numerical value (e.g. Distance cost, Time cost).
- **Undirected:** Edges are bidirectional. Edge (u, v) and (v, u) both exist.
- **Directed:** Edges are unidirectional. Edge (u, v) represents only u -> v.
- **Cyclic:** Graph has a cycle.
- **Acyclic:** Graph has no cycles.
- **What type of graph is a Tree?**
  - *Undirected, Acyclic Graph
    - *The tree's we've looked at are technically directed, acyclic graphs where removing any edge would disconnect the graph.



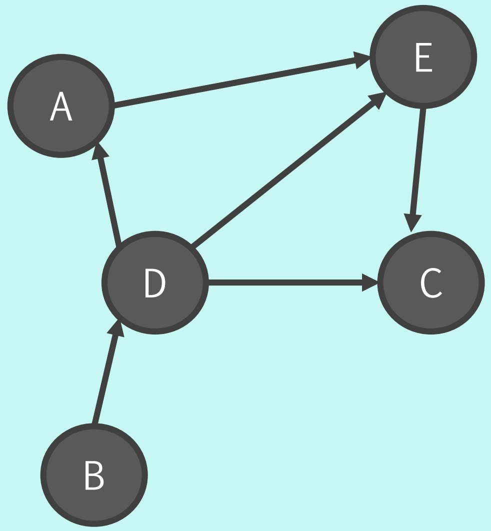
Unweighted, Undirected, Cyclic Graph



Weighted, Directed, Acyclic Graph

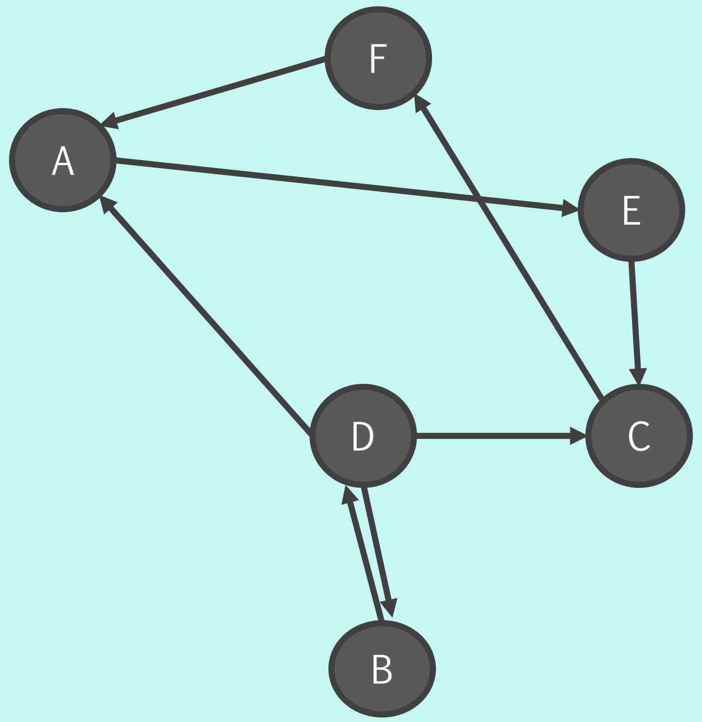
Vertex Terminology

- **Degree**: The number of edges incident to vertex v.
  - Degree(A) = 2
- **Indegree:** The number of incoming edges to vertex v.
  - InDegree(C) = 2
- **Out Degree:** The number of outgoing edges from vertex v.
  - OutDegree(D) = 3



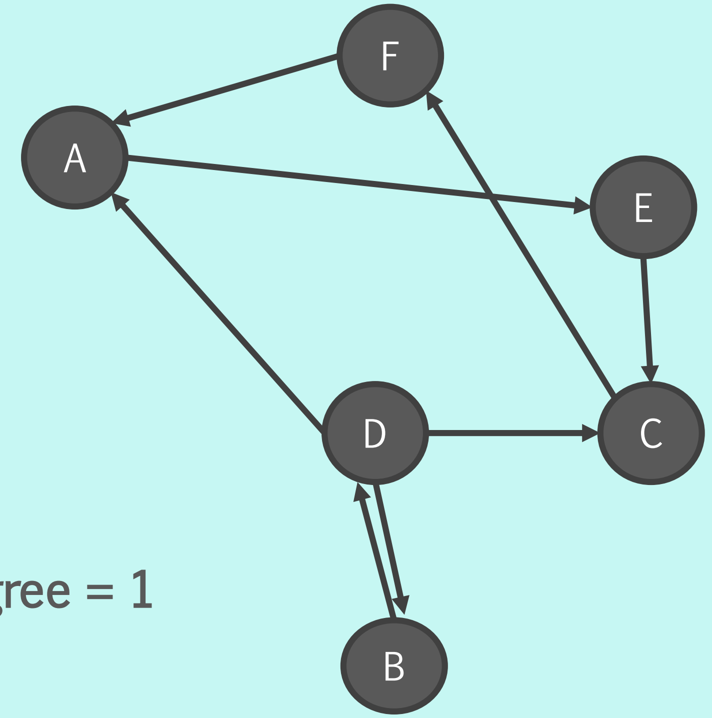
Practice

- On your paper answer for this graph:
  1. What is set V
  2. What is set E
  3. Find a path of length 2
  4. Find a Cycle
  5. Is this graph directed?
  6. What is the degree of C?



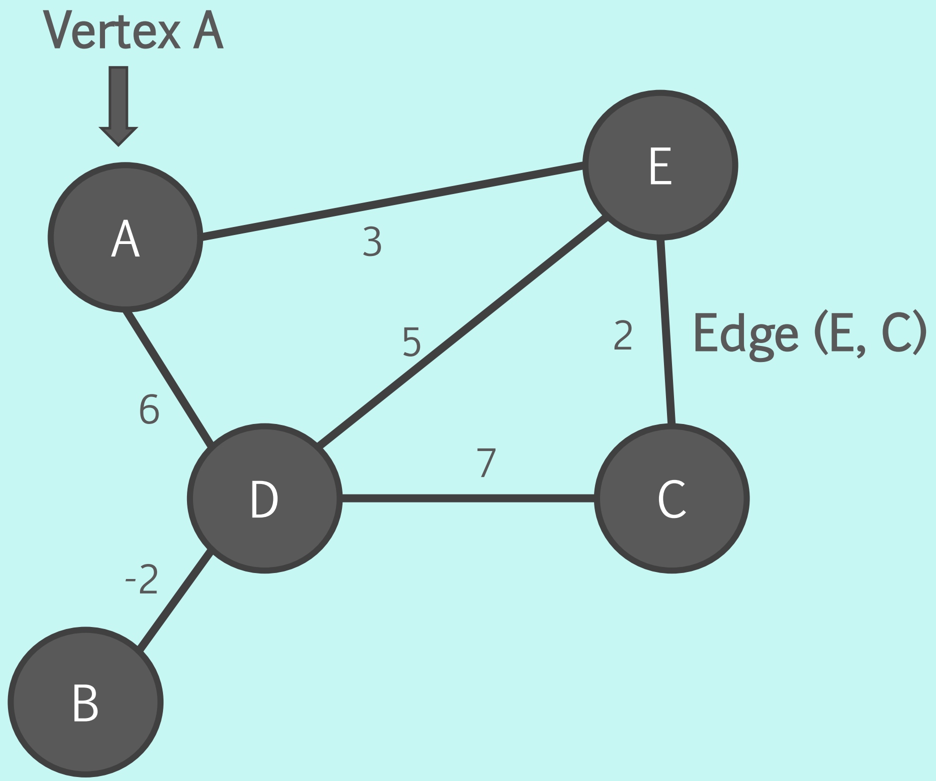
Practice

- On your paper answer for this graph:
  1. What is set V. **{A, B, C, D, E, F}**
  2. What is set E. **{(A, E), (B, D), (C, F), (D, A), (D, B), (D, C), (E, C), (F, A)}**
  3. Find a path of length 2. **(A, E, C) and more.**
  4. Find a Cycle. **(A, E, C, F, A) or (B, D, B)**
  5. Is this graph directed? **Yes**
  6. What is the degree of C? **3. indegree = 2. outdegree = 1**



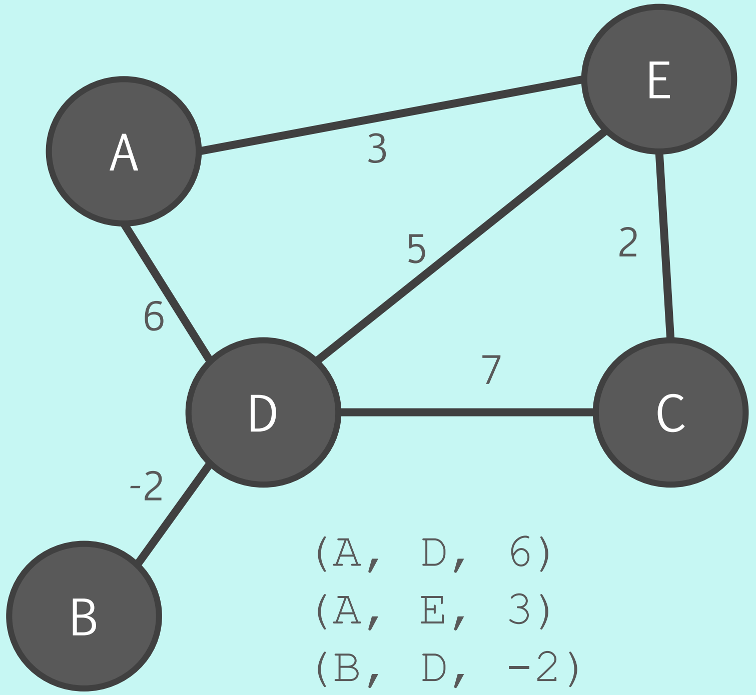
Representing Graphs

- Within our code, how do we store these graphs?
  - Unlike a tree where we start at Root or a LinkedList where we start at head, we want to access Vertices and Edges quickly.



Edge List

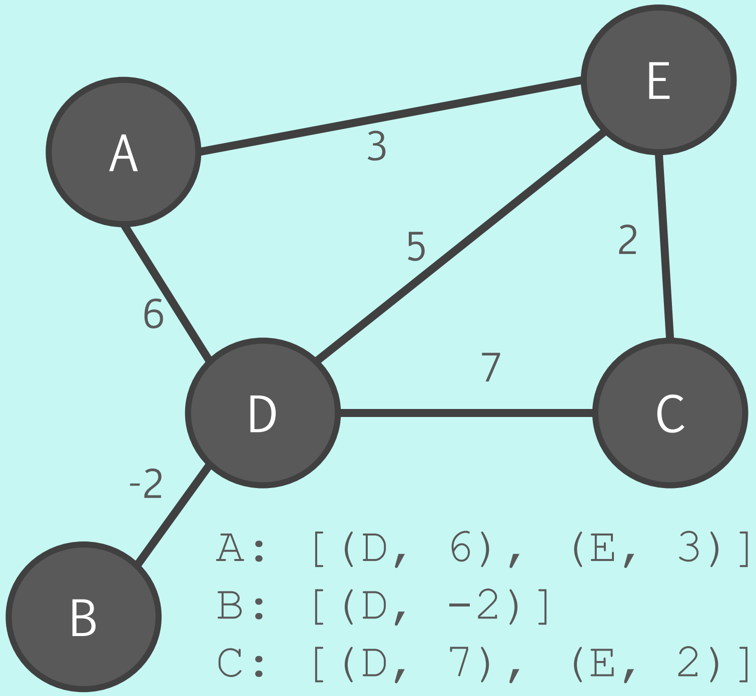
- The graph is stored as a list of all edges in the graph.
- Pros:
  - Very simple to implement.
- Cons:
  - Time consuming to search vertices
  - O(|E|) look up for specific edges.
  - Hard to find any paths.



```
(A, D, 6)
(A, E, 3)
(B, D, -2)
(C, D, 7)
(C, E, 2)
(D, E, 5)
```

Adjacency List

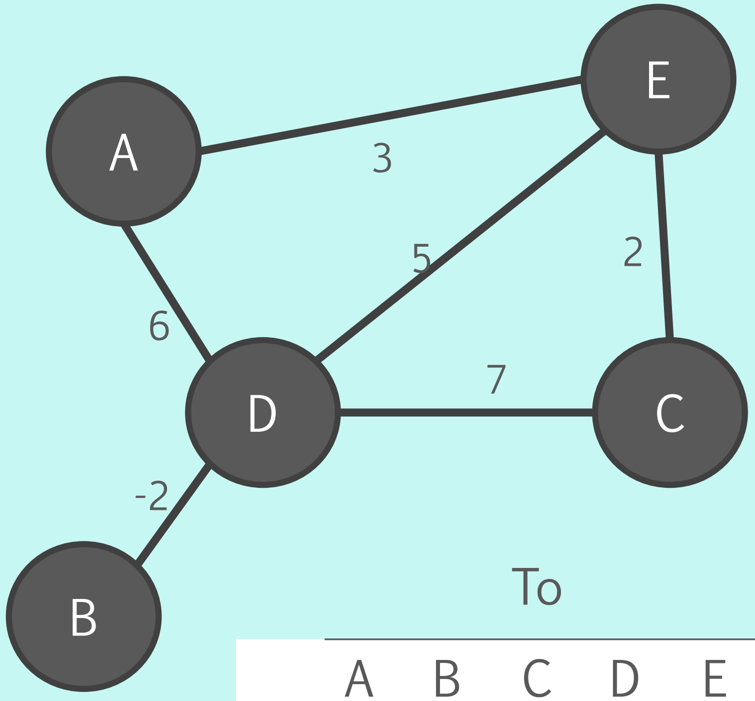
- The graph is represented as a list of Vertices where each Vertex has a list of all neighboring vertices.
  - E.g. Map<Vertex, List<Vertex>>
- Pros:
  - For sparse graphs, very space efficient. O(|E|+|V|)
  - Getting all neighbors for a Vertex is efficient.
- Cons:
  - Testing whether two vertices are connected is O(|V|).



```
A: [(D, 6), (E, 3)]
B: [(D, -2)]
C: [(D, 7), (E, 2)]
D: [(A, 6), (B, -2),
    (C, 7), (E, 5)]
E: [(A, 3), (C, 2), (D, 5)]
```

Adjacency Matrix

- The graph is represented as V x V matrix.   Accessing Matrix(u, v) represent edge (u, v).
  - E.g. Vertex[V][V]
- Pros
  - Fast look-up for vertex connection O(1).
  - Space efficient for dense graphs.
- Cons
  - For sparse graphs, very space inefficient. $O(V^2)$ space
  - Getting all neighbors of a Vertex is O(|V|)

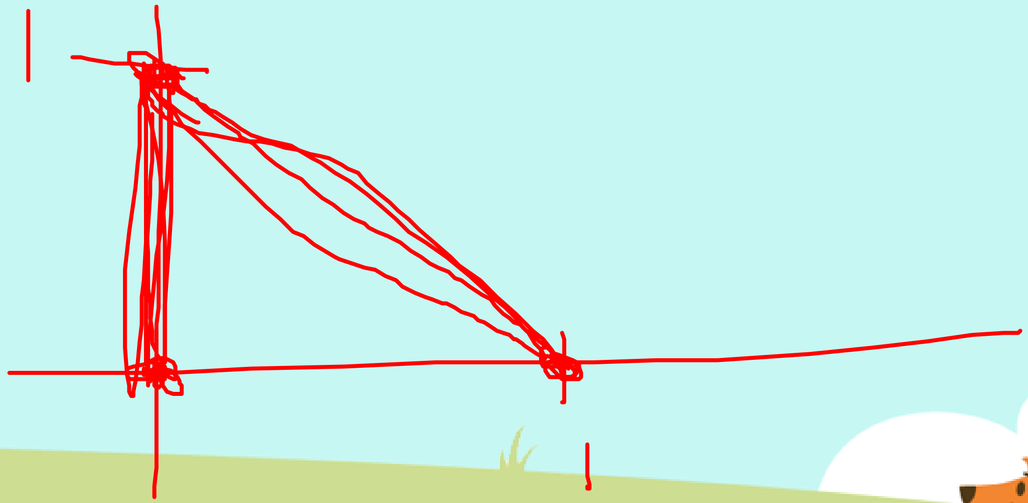


To

From	A	B	C	D	E
A				6	3
B				-2	
C				7	2
D	6	-2	7		5
E	3		2	5	

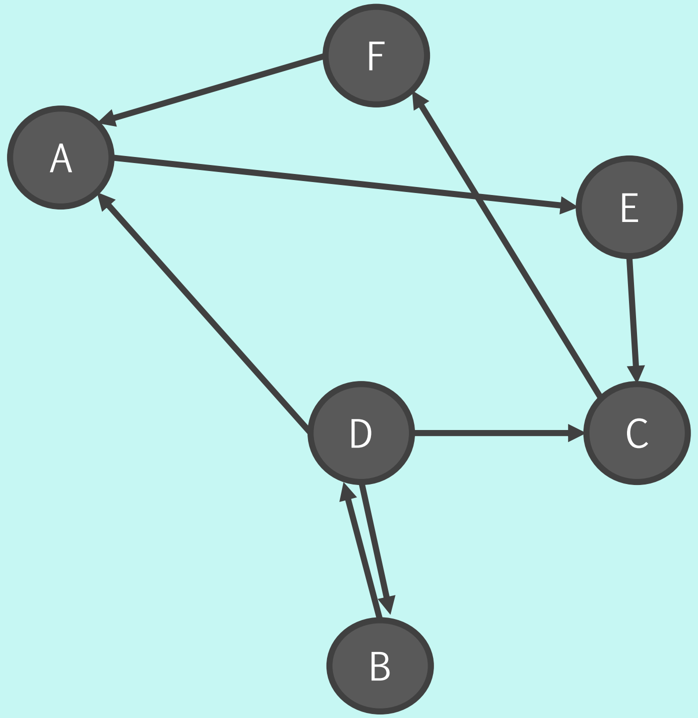
Which to Choose

- Choosing between Adjacency Lists and Matrices depends on what operations you want to do.
  - How many vertices and edges does your graph have? Dense vs sparse
  - Do you care about maintaining connections? Adjacency Matrix
  - Do you care about getting all neighbors of a vertex quickly? Adjacency List
  - Do you want a bad homework grade? Edge List
    - Vertex-Edge List
      - 3
        - (0,0)
        - (0, 1)
        - (1, 0)
      - 2
        - (0, 1)
        - (1, 2)



Practice

- Write the Edge List, Adjacency List, and Adjacency Matrix representation of the graph.
- Edge List
- Adjacency List
- Adjacency Matrix



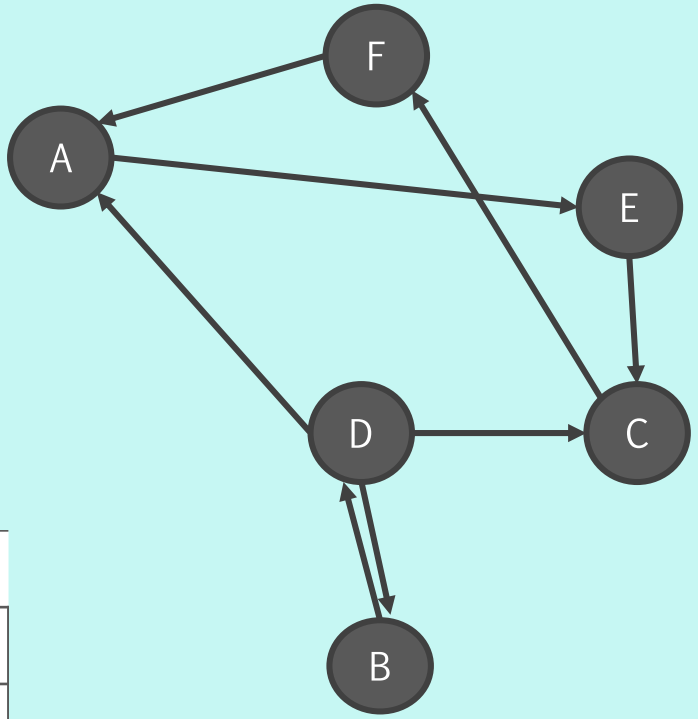
Practice

- Write the Edge List, Adjacency List, and Adjacency Matrix representation of the graph.
- Edge List
  - [(A, E), (B, D), (C, F), (D, A), (D, B), (D, C), (E, C), (F, A)]
- Adjacency List

  A: [E]
  B: [D]
  C: [F]
  D: [A, B, C]
  E: [C]
  F: [A]

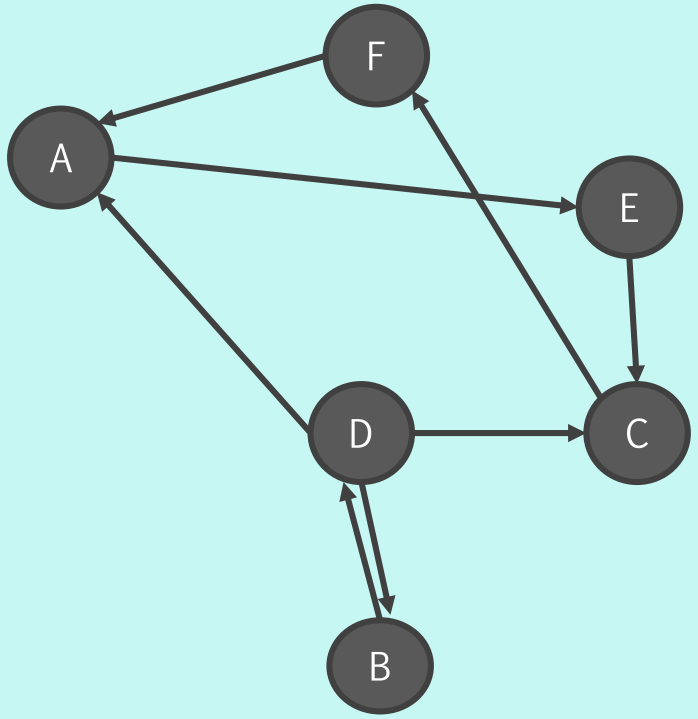
- Adjacency Matrix

$$\begin{array}{c|c|cccccc} & & & & \text{To} & & & \\ & & A & B & C & D & E & F \\ \hline & A & & & & & 1 & \\ & B & & & & 1 & & \\ \text{From} & C & & & & & & 1 \\ & D & 1 & 1 & 1 & & & \\ & E & & & 1 & & & \\ & F & 1 & & & & & \end{array}$$



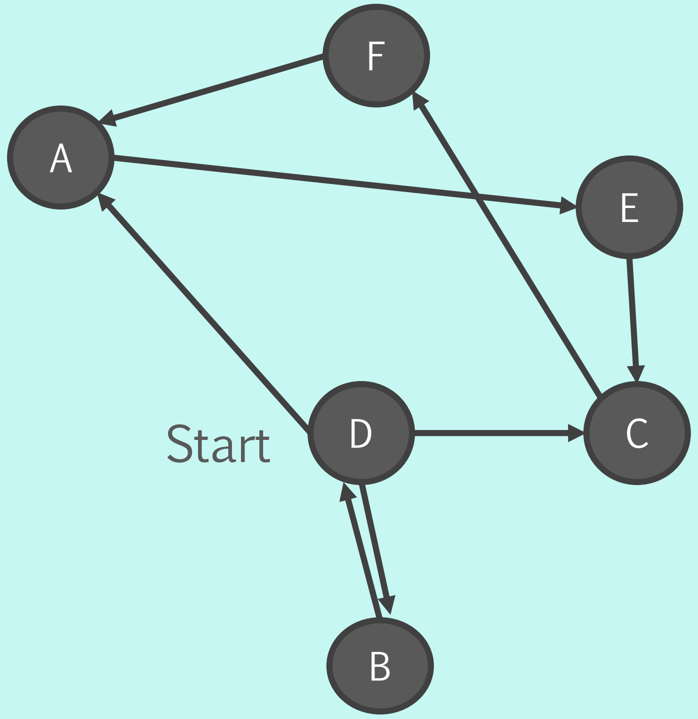
Graph Traversals

- Graph traversals are methods of traversing through a graph in a systematic way.
- Similar to pre/post/level order traversals for trees.
  - Difference is that trees are acyclic, so our tree traversals did not visit a vertex more than once.
  - Our traversal algorithm must handle this.

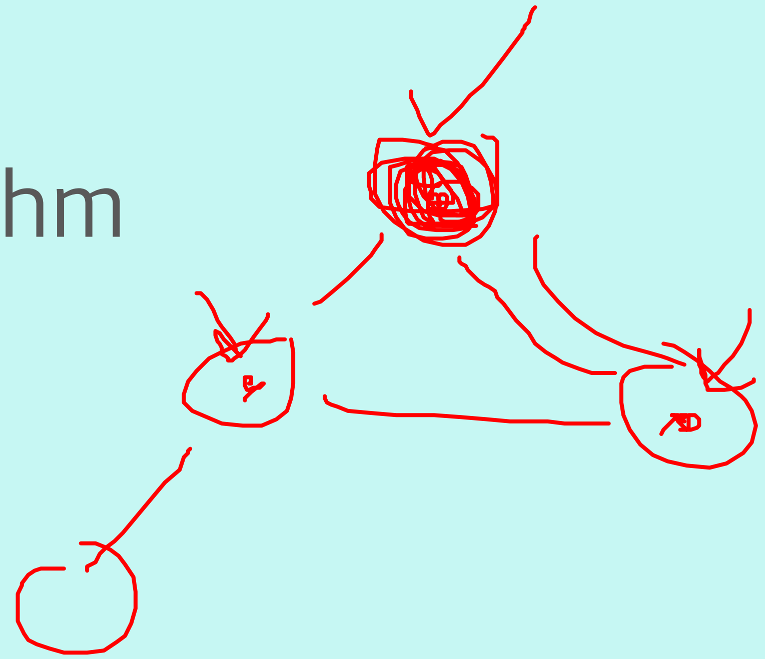


Graph Traversal Example

- Lets say we start at D.
- If I want to traverse the graph, I need to traverse from D to one of D's neighbors. (Let's say C).
- From C, I need to visit another neighbor. (Vertex F)
- But I can also visit another one of D's neighbors (A or B).
- What to do…



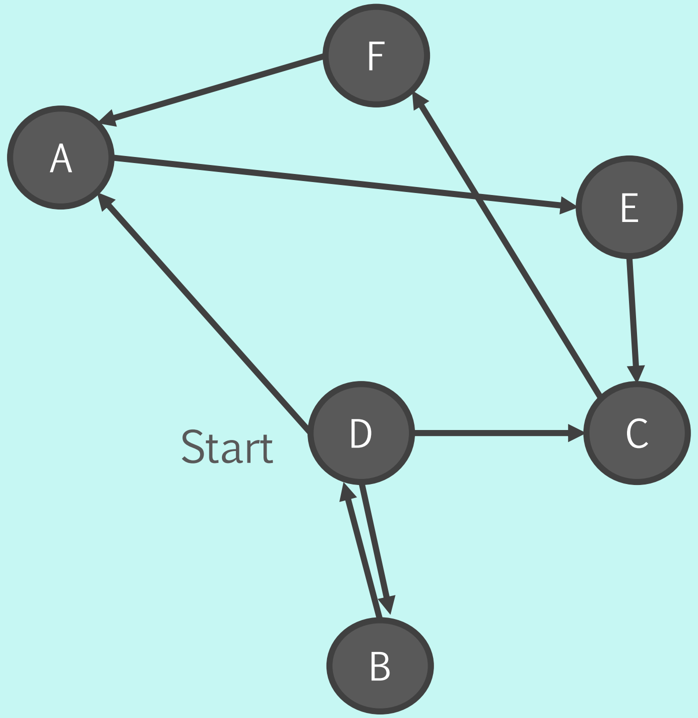
General Graph Search Algorithm



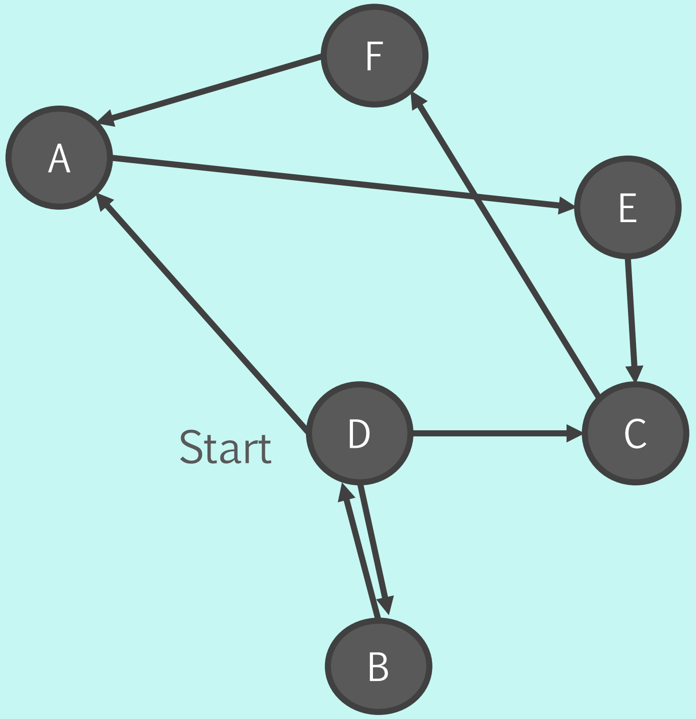
```
GraphSearch(start, goal)
  Set visited
  Structure s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // do something if curr is the goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search

- Using the General Graph Search Algorithm, **Structure s = Queue**
- Starting at a vertex v, the search will traverse to all of neighbor(v), then the traverse to the neighbor's neighbors, and so on.



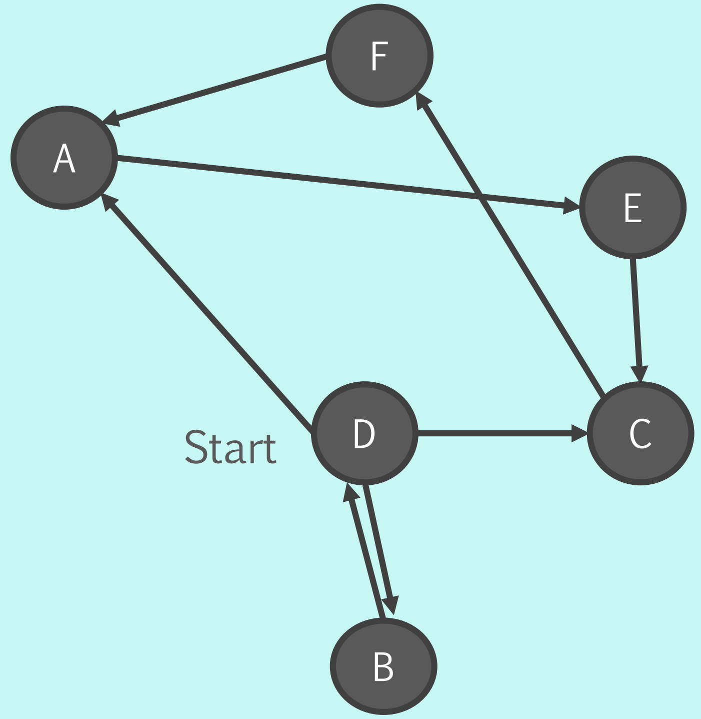
Breadth First Search Example



```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited:

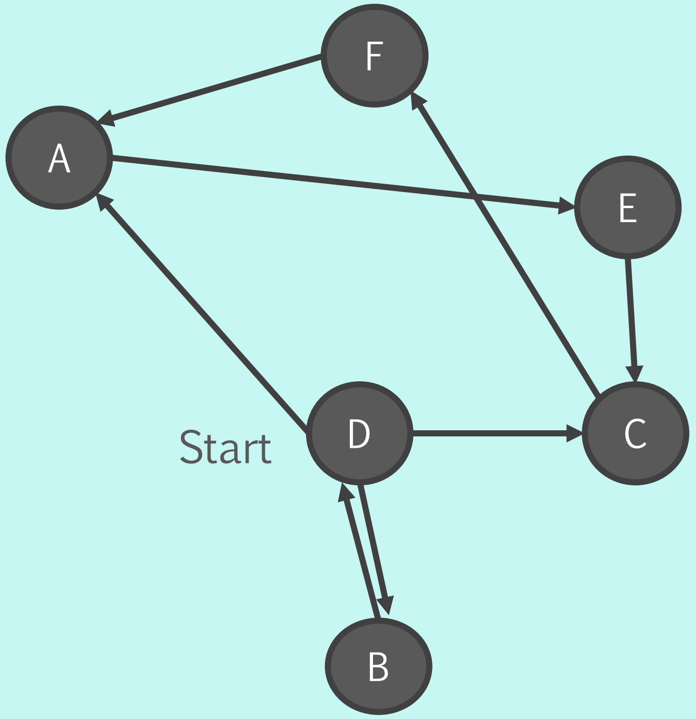


```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited:

Queue:

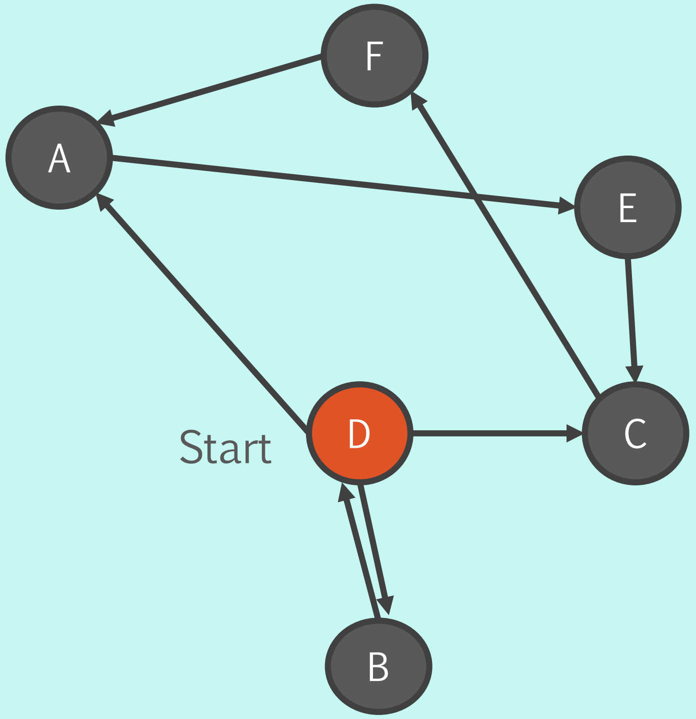


```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited:

Queue: D

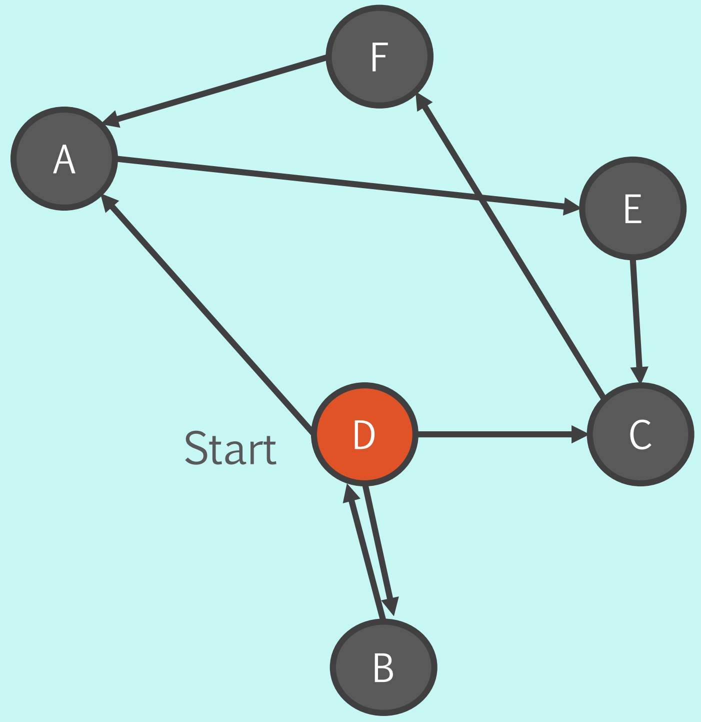


```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited:

Queue: D



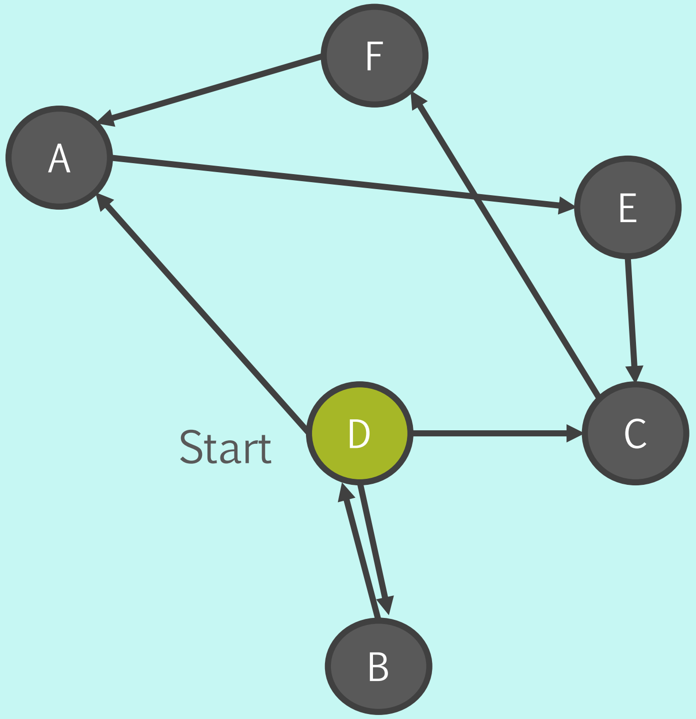
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited:

Queue:

Curr: D



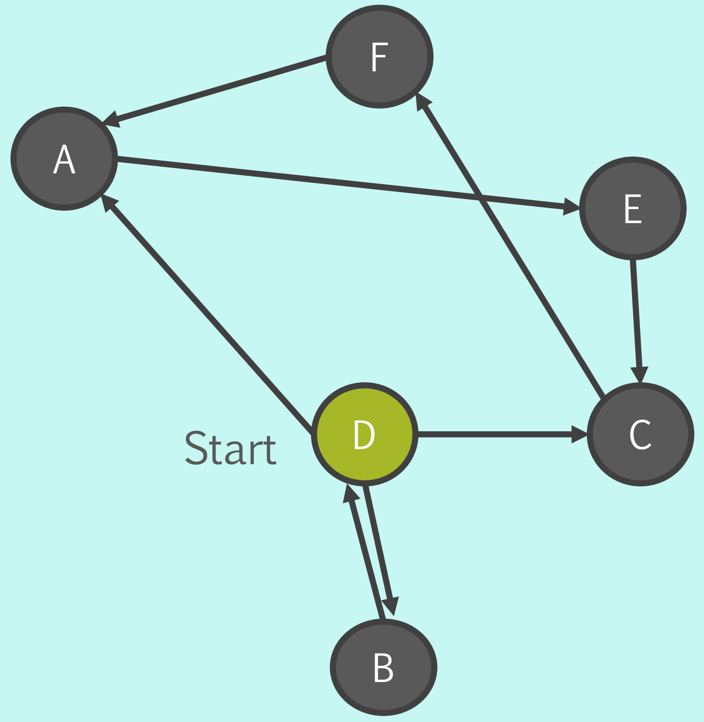
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited:

Queue:

Curr: D



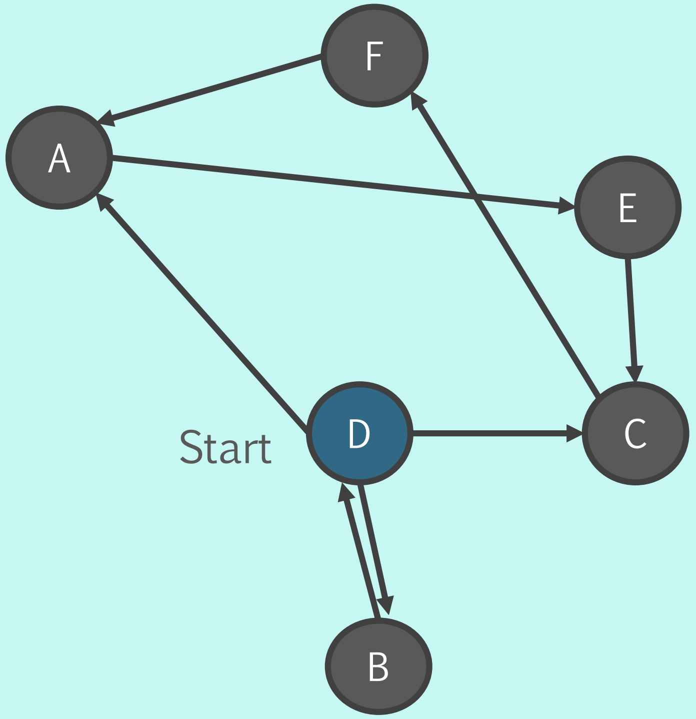
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D

Queue:

Curr: D



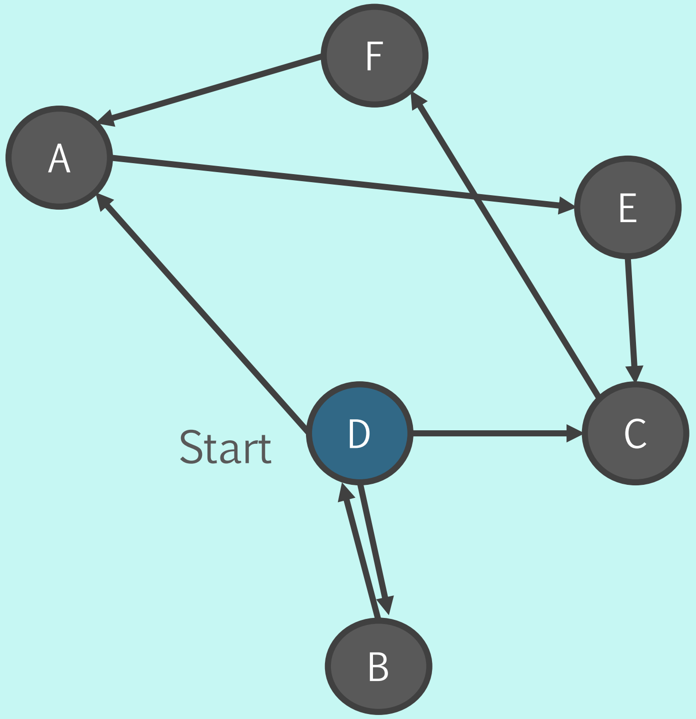
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D

Queue:

Curr: D



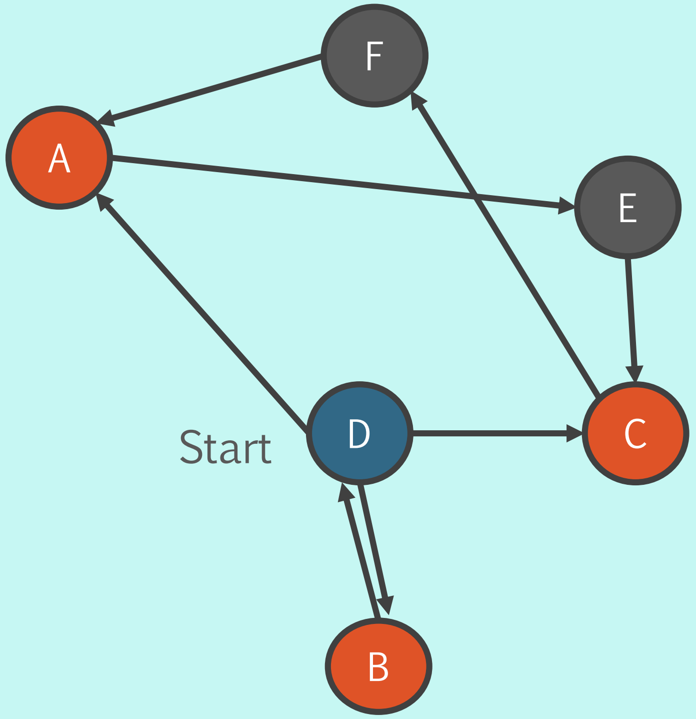
```
GraphSearch(start, goal)
   Set visited
   Queue s
   s.add(start)
   while (s not empty)
      curr = s.remove()
      if (curr is visited)
         continue
      visited.add(curr)
      evaluate(curr) // check if goal
      for Vertex u in neighbors(curr)
         s.add(u)
```

Breadth First Search Example

Visited: D

Queue:

Curr: D



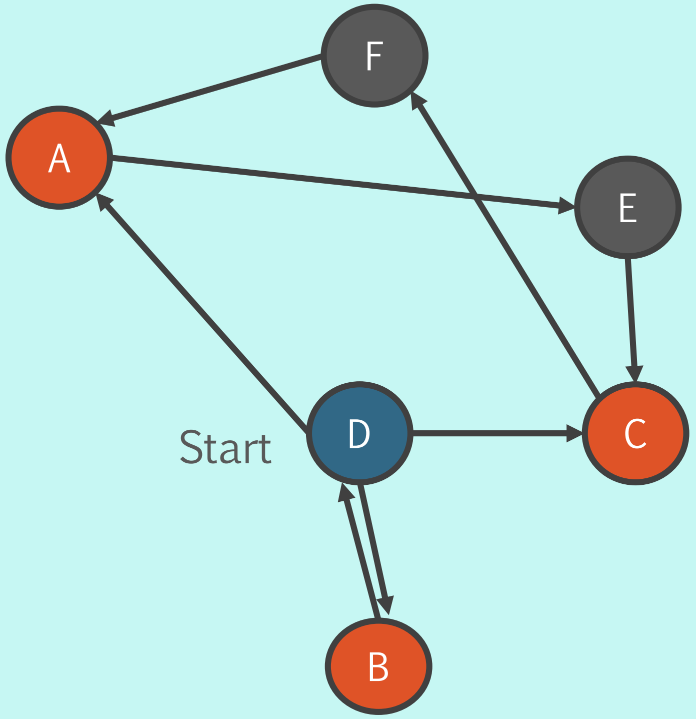
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D

Queue: A, B, C

Curr: D



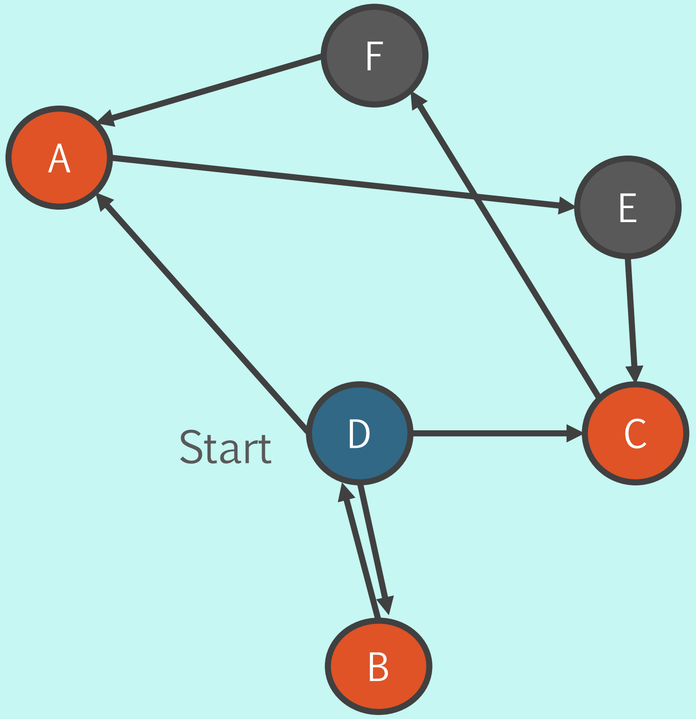
```
GraphSearch(start, goal)
  **Set** visited
  **Queue** s
  s.add(start)
  **while** (s **not** empty)
    curr = s.remove()
    **if** (curr **is** visited)
      **continue**
    visited.add(curr)
    evaluate(curr) // check if goal
    **for** Vertex u **in** neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D

Queue: A, B, C

Curr: D



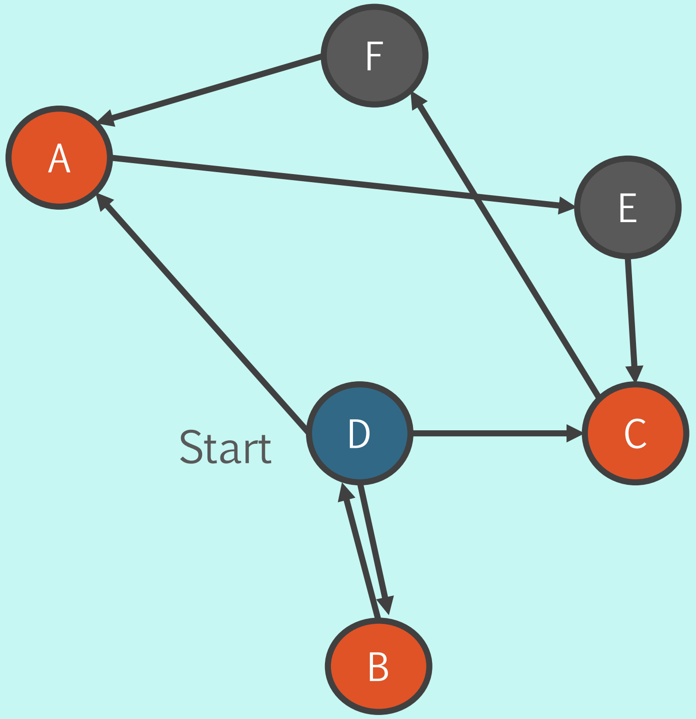
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D

Queue: A, B, C

Curr: D



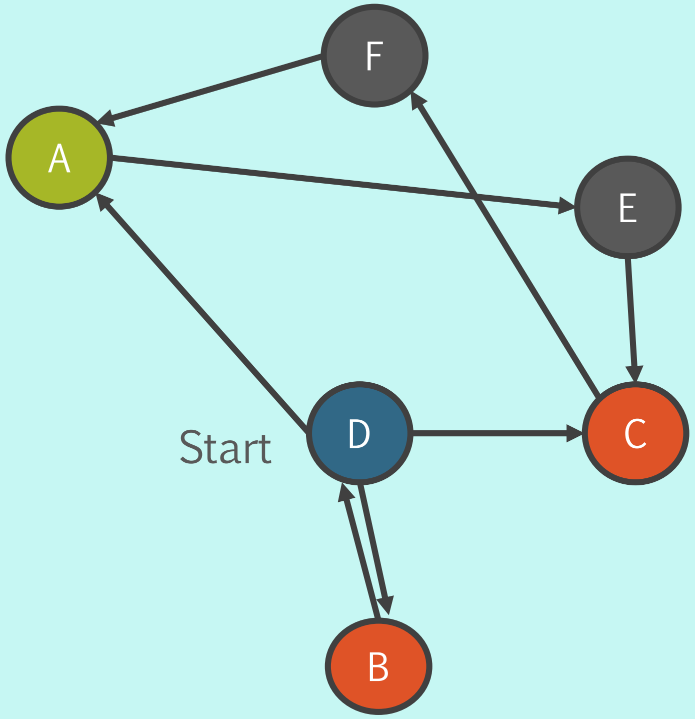
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D

Queue: B, C

Curr: A



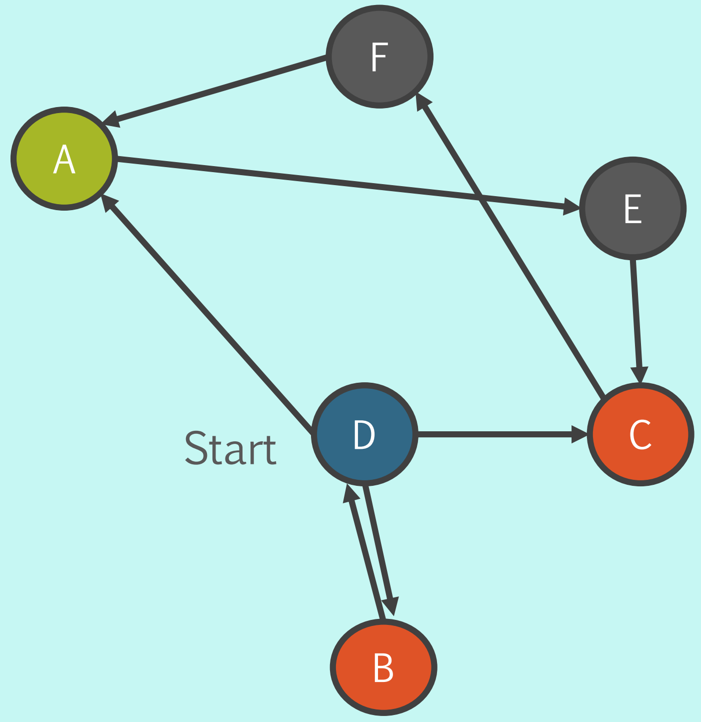
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D

Queue: B, C

Curr: A



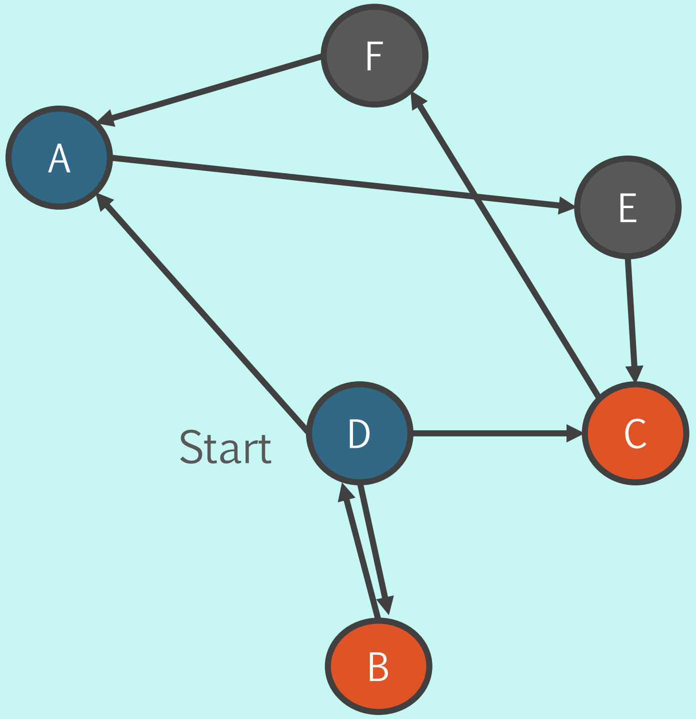
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A

Queue: B, C

Curr: A



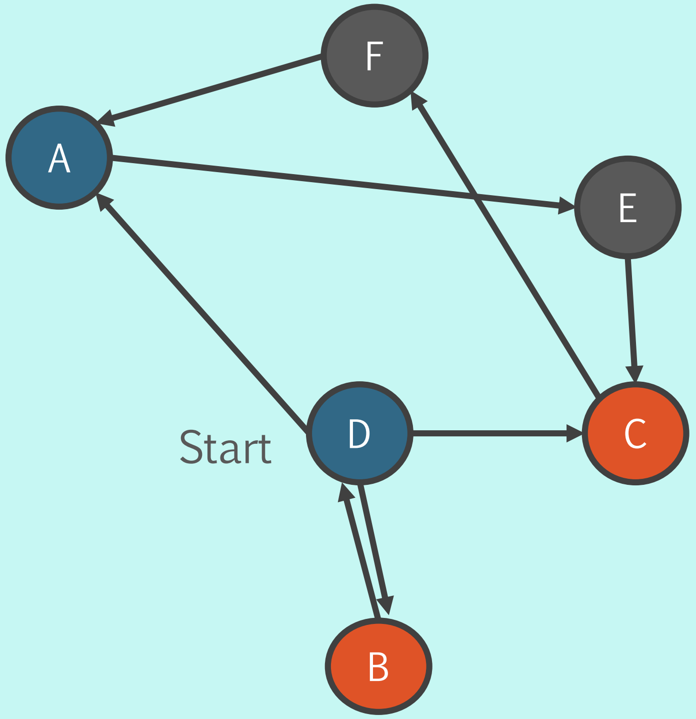
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A

Queue: B, C

Curr: A



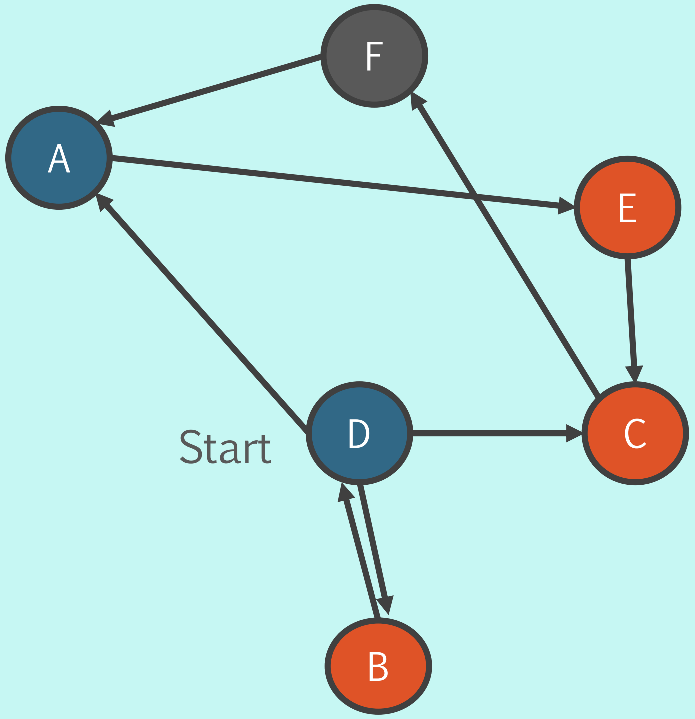
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A

Queue: B, C

Curr: A



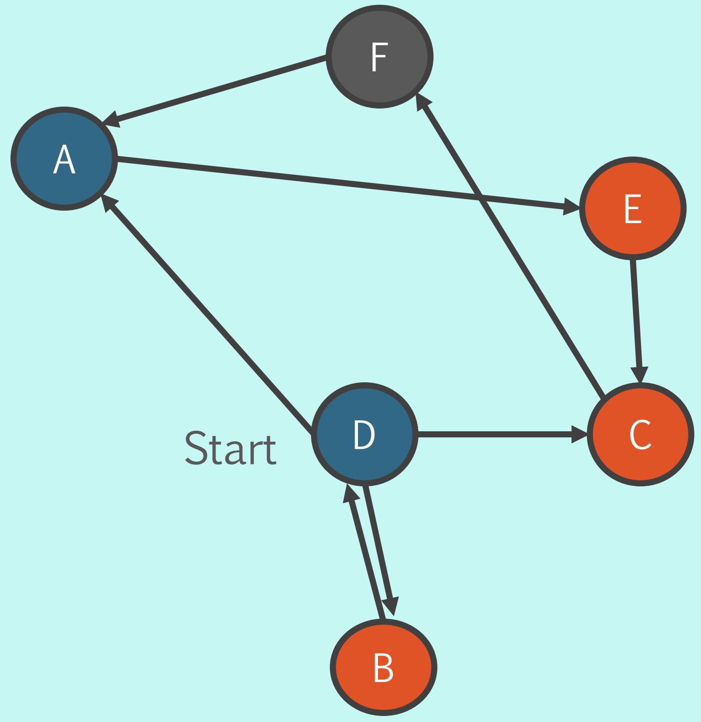
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A

Queue: B, C, E

Curr: A



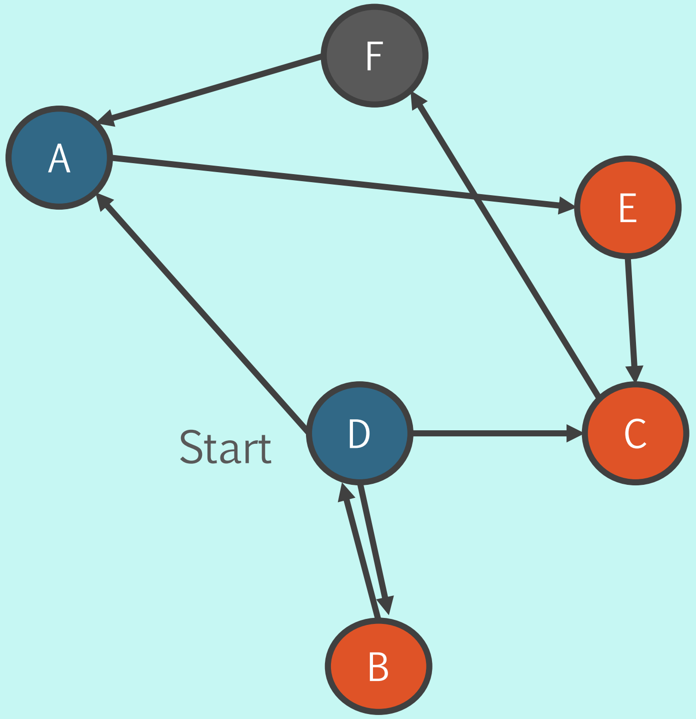
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A

Queue: B, C, E

Curr: A



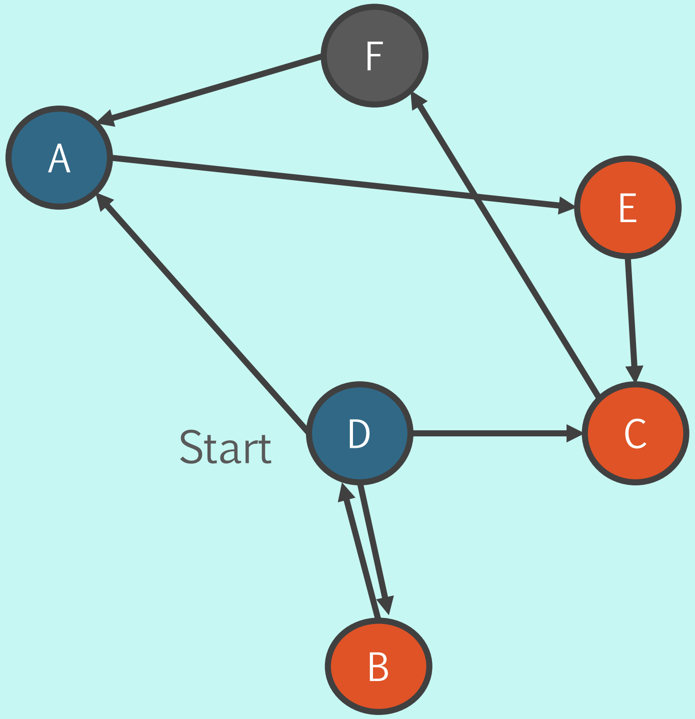
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A

Queue: B, C, E

Curr: A



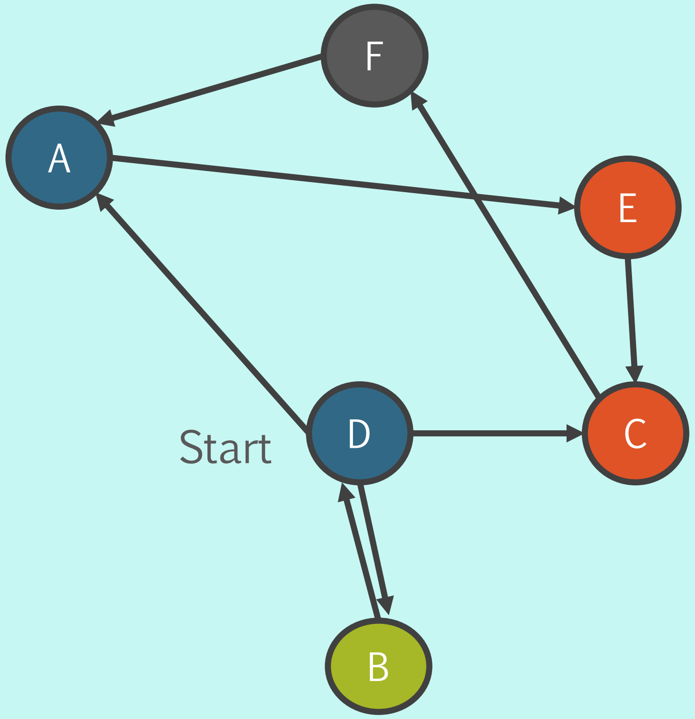
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A

Queue: C, E

Curr: B



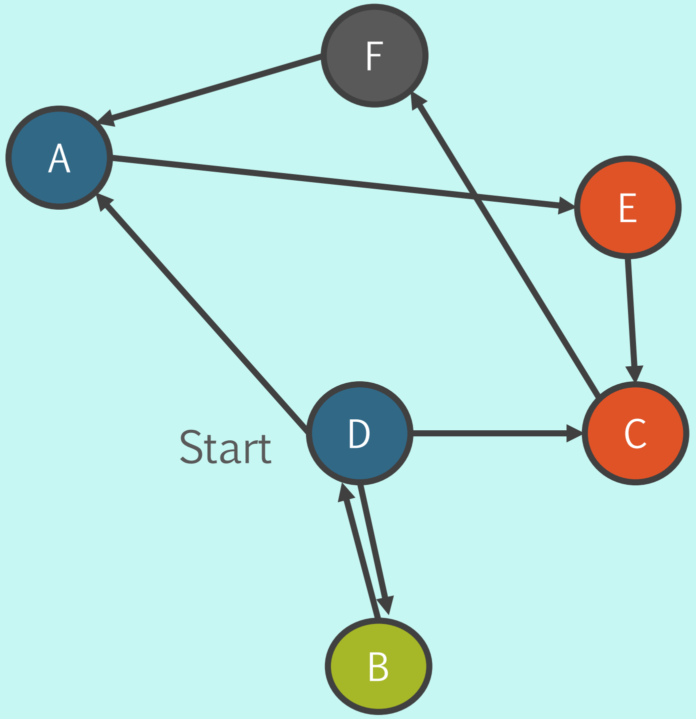
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A

Queue: C, E

Curr: B



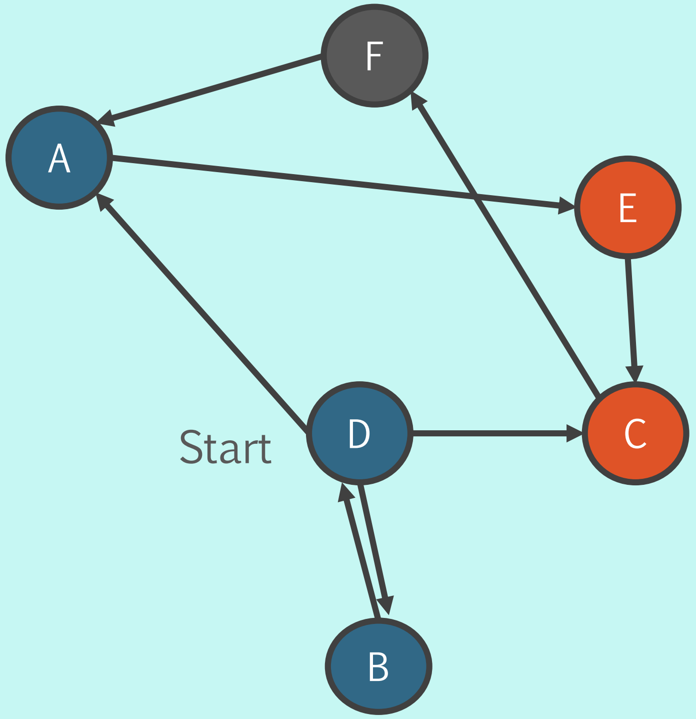
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B

Queue: C, E

Curr: B



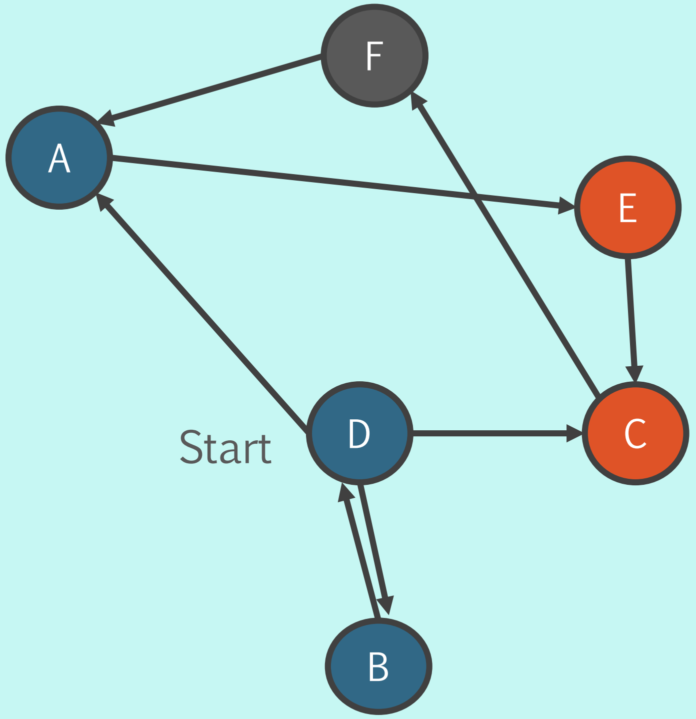
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B

Queue: C, E

Curr: B



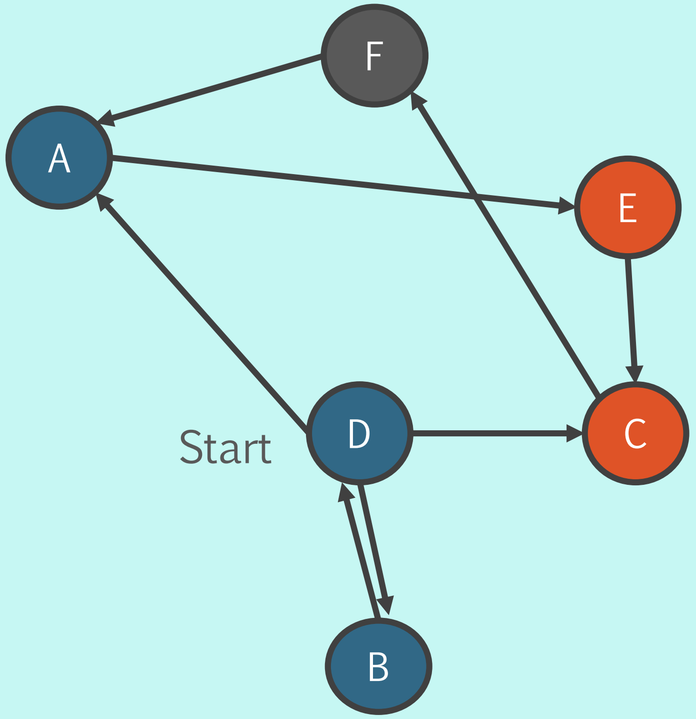
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B

Queue: C, E

Curr: B



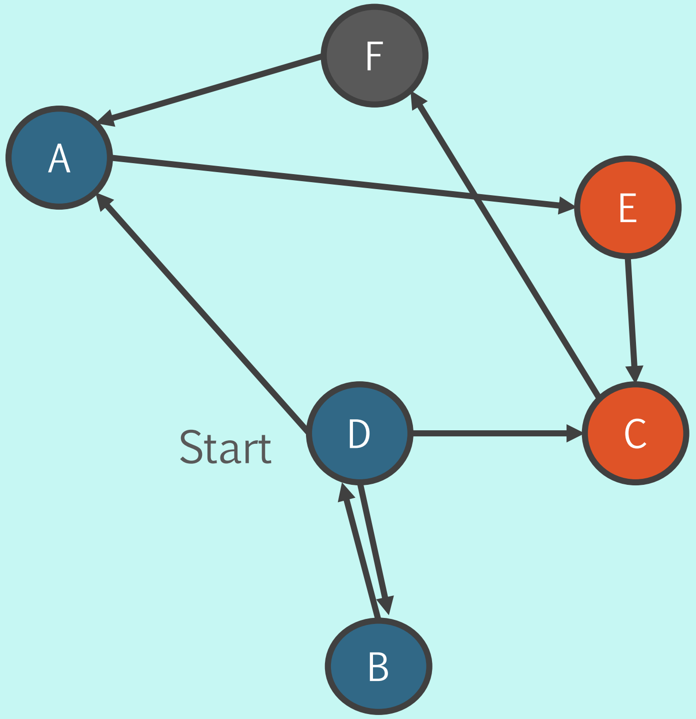
```
GraphSearch(start, goal)
  **Set** visited
  **Queue** s
  s.add(start)
  **while** (s **not** empty)
    curr = s.remove()
    **if** (curr **is** visited)
      **continue**
    visited.add(curr)
    evaluate(curr) // check if goal
    **for** Vertex u **in** neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B

Queue: C, E, D

Curr: B



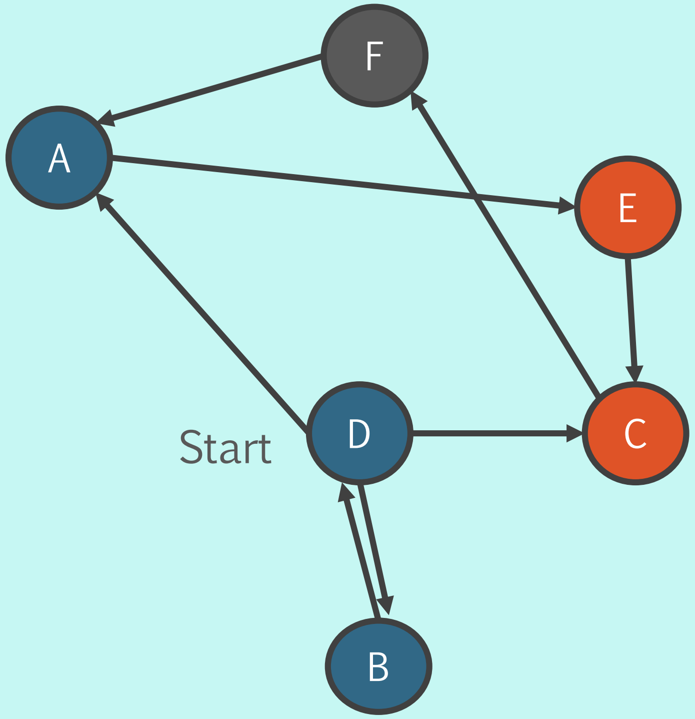
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B

Queue: C, E, D

Curr: B



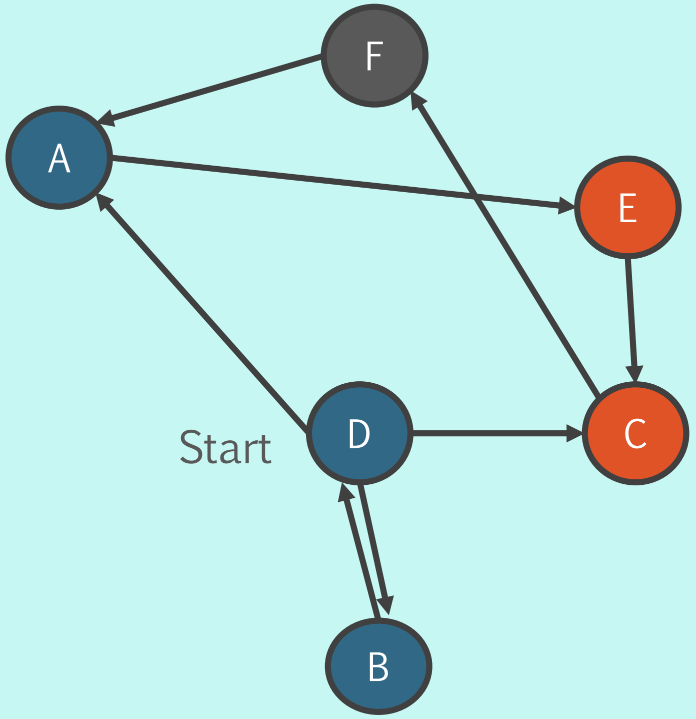
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B

Queue: C, E, D

Curr: B



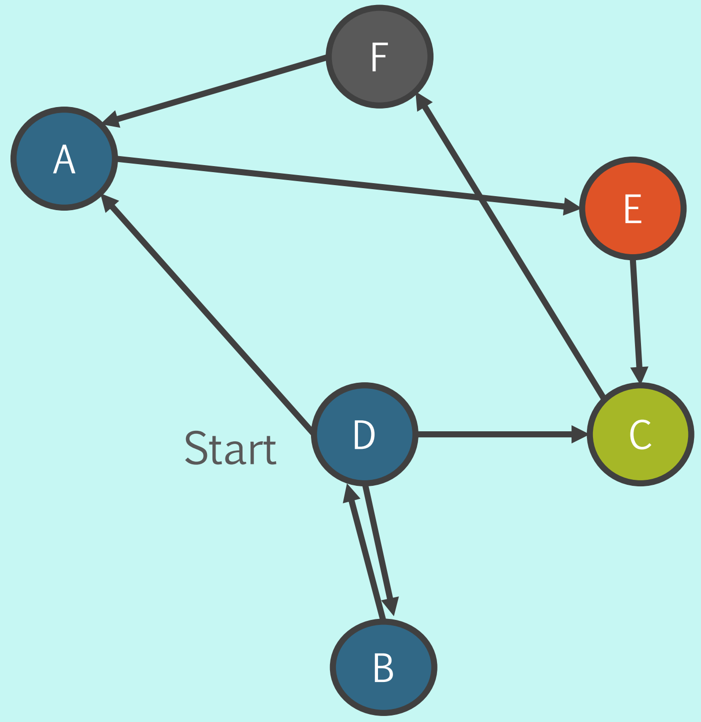
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B

Queue: E, D

Curr: C



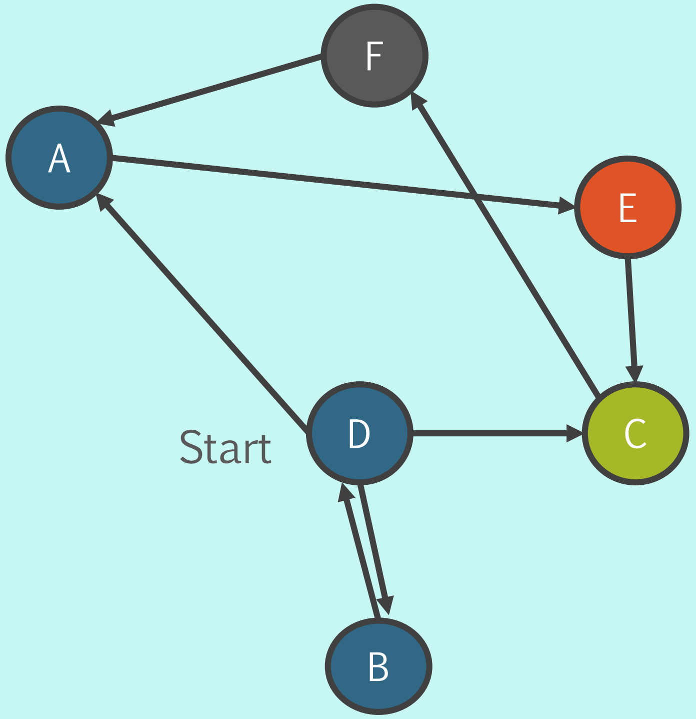
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B

Queue: E, D

Curr: C



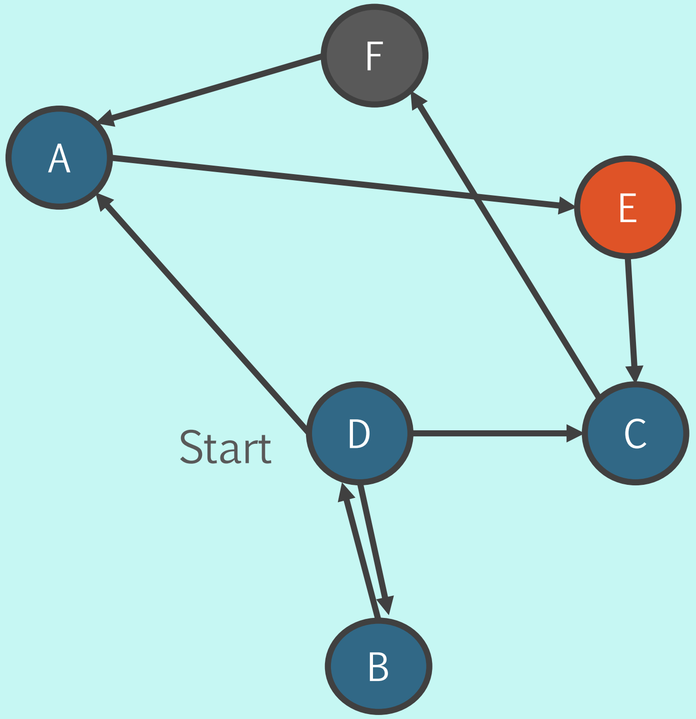
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C

Queue: E, D

Curr: C



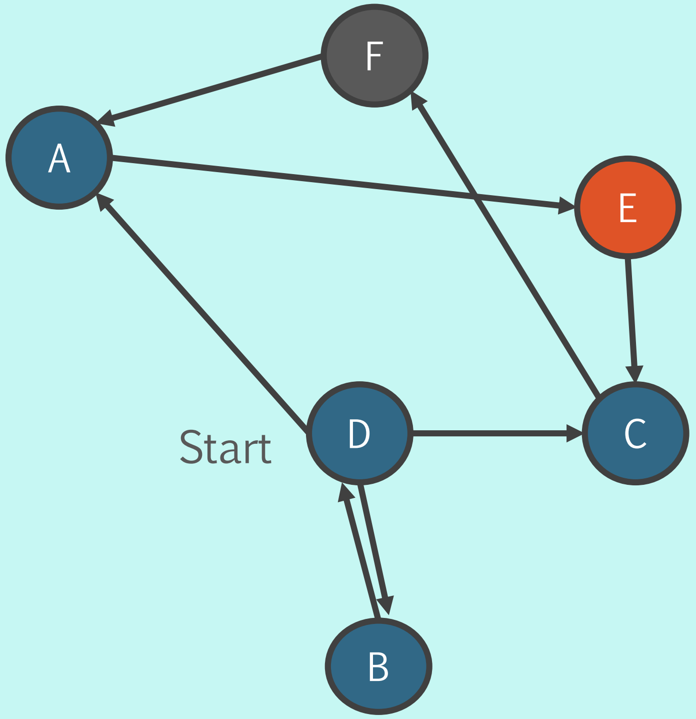
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C

Queue: E, D

Curr: C



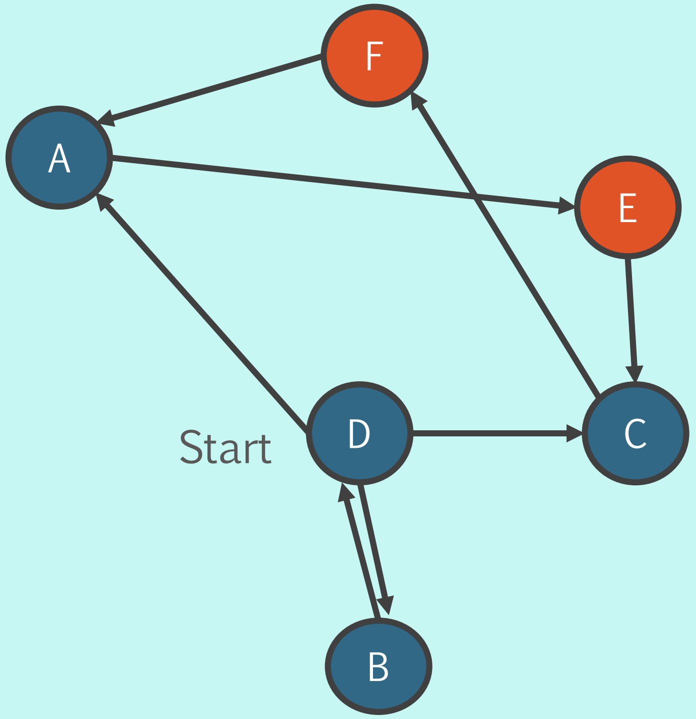
```
GraphSearch(start, goal)
   Set visited
   Queue s
   s.add(start)
   while (s not empty)
      curr = s.remove()
      if (curr is visited)
         continue
      visited.add(curr)
      evaluate(curr) // check if goal
      for Vertex u in neighbors(curr)
         s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C

Queue: E, D

Curr: C



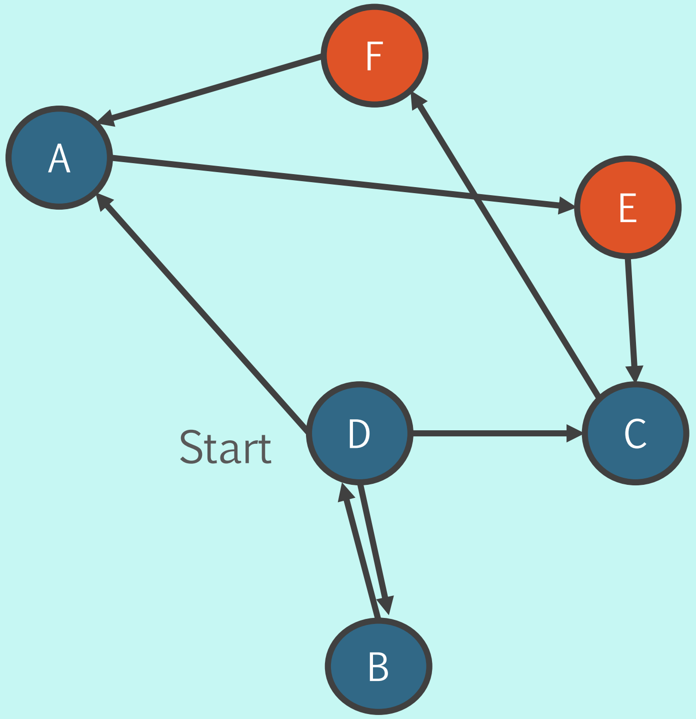
```
GraphSearch(start, goal)
   Set visited
   Queue s
   s.add(start)
   while (s not empty)
      curr = s.remove()
      if (curr is visited)
         continue
      visited.add(curr)
      evaluate(curr) // check if goal
      for Vertex u in neighbors(curr)
         s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C

Queue: E, D, F

Curr: C



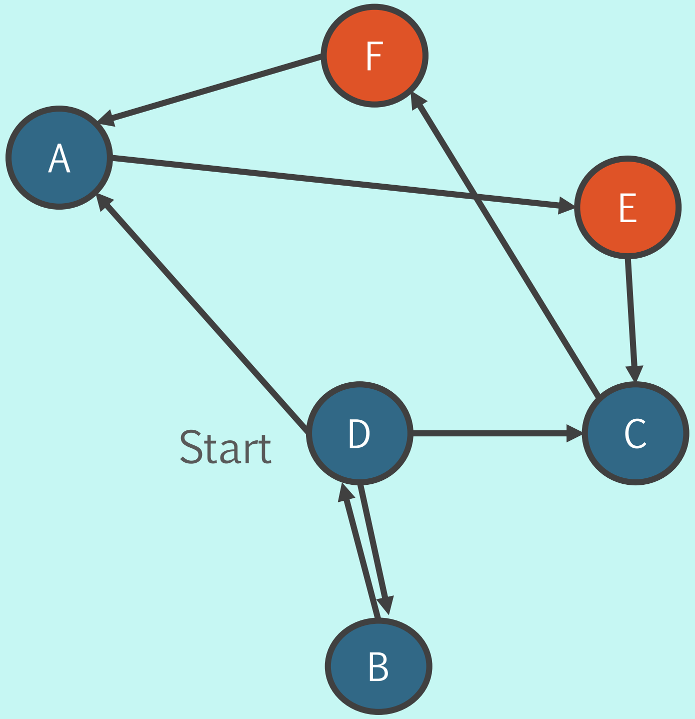
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C

Queue: E, D, F

Curr: C



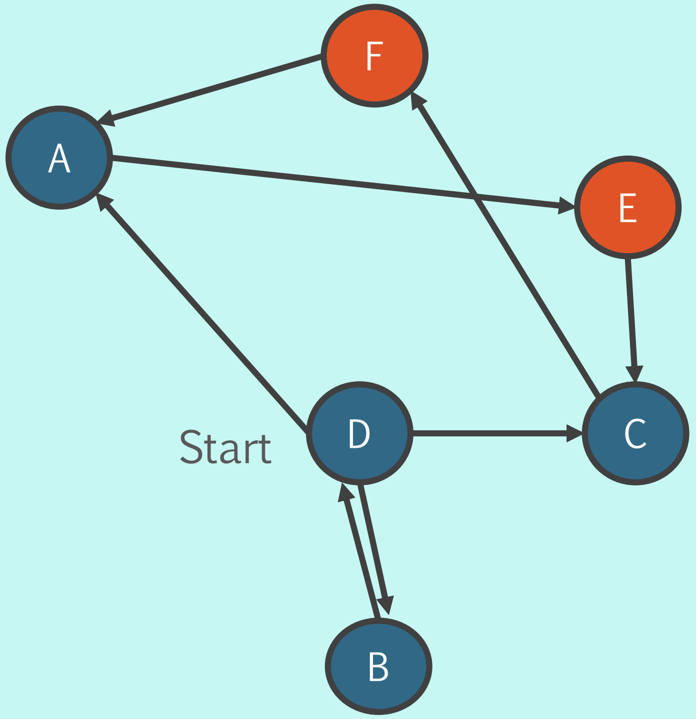
```
GraphSearch(start, goal)
   Set visited
   Queue s
   s.add(start)
   while (s not empty)
      curr = s.remove()
      if (curr is visited)
         continue
      visited.add(curr)
      evaluate(curr) // check if goal
      for Vertex u in neighbors(curr)
         s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C

Queue: E, D, F

Curr: C



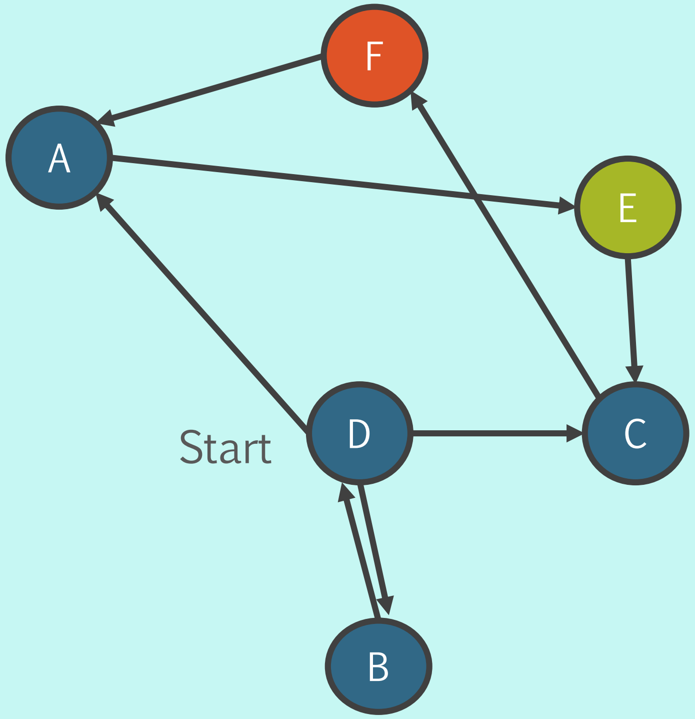
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C

Queue: D, F

Curr: E



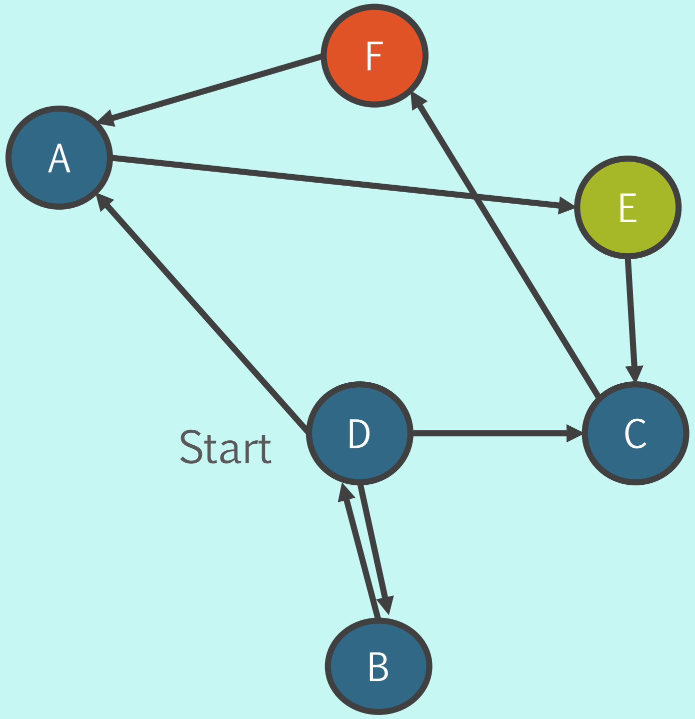
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C

Queue: D, F

Curr: E



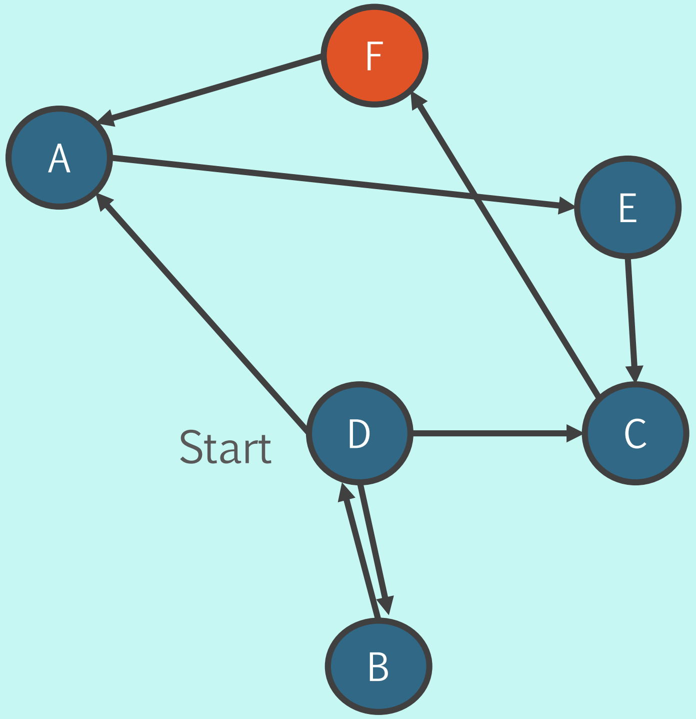
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E

Queue: D, F

Curr: E



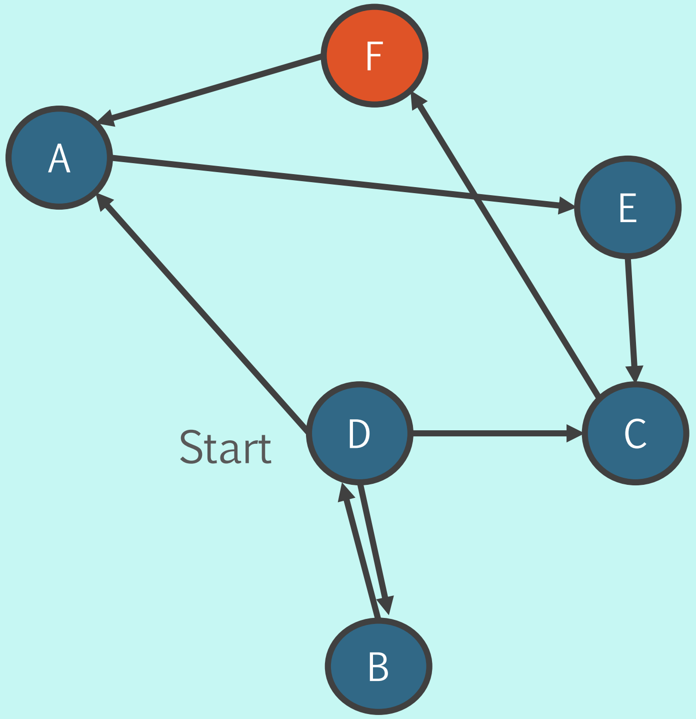
```
GraphSearch(start, goal)
   Set visited
   Queue s
   s.add(start)
   while (s not empty)
      curr = s.remove()
      if (curr is visited)
         continue
      visited.add(curr)
      evaluate(curr) // check if goal
      for Vertex u in neighbors(curr)
         s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E

Queue: D, F

Curr: E



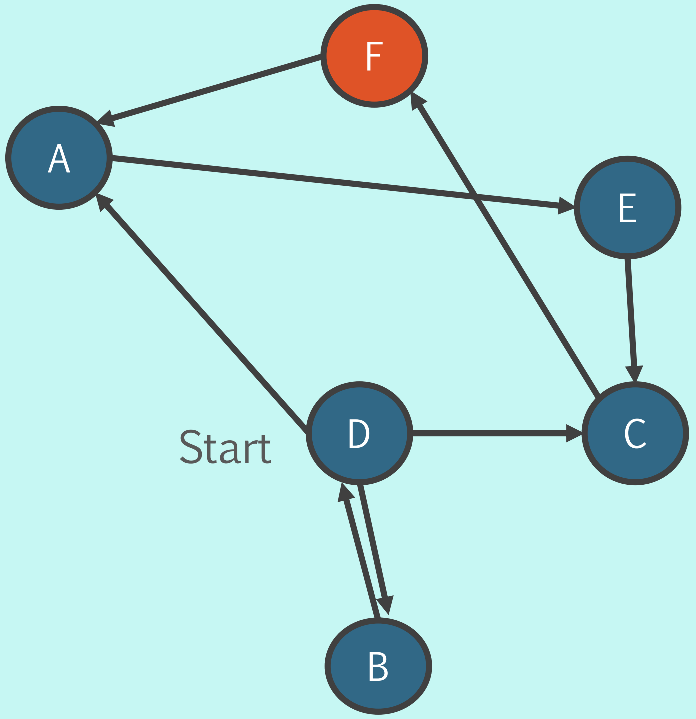
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E

Queue: D, F

Curr: E



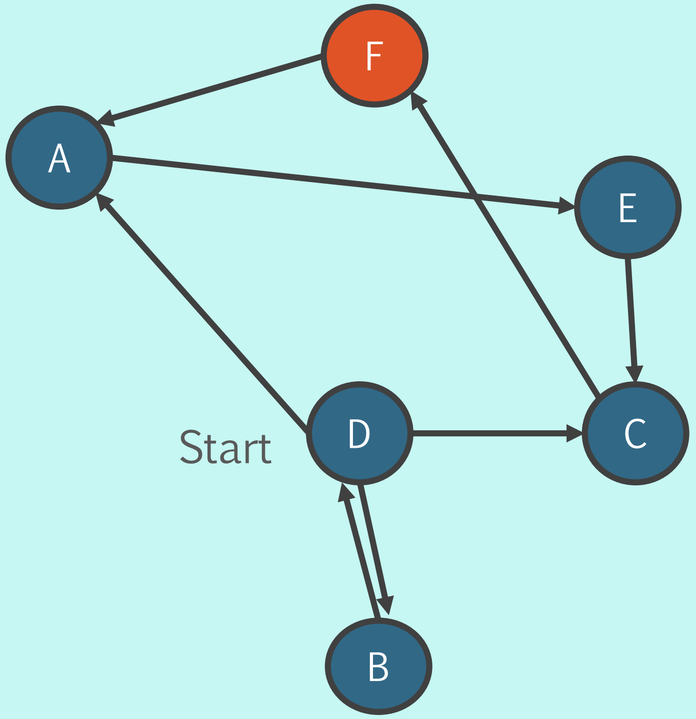
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E

Queue: D, F, C

Curr: E



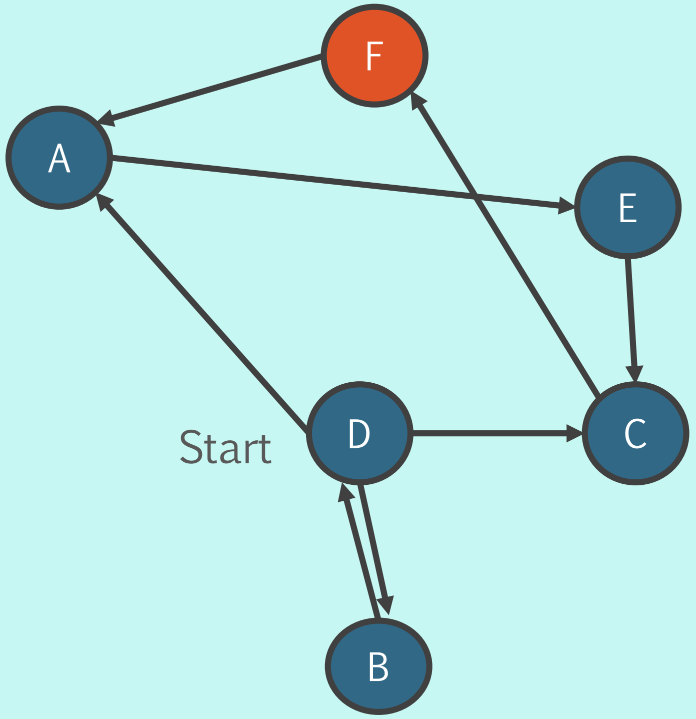
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E

Queue: D, F, C

Curr: E



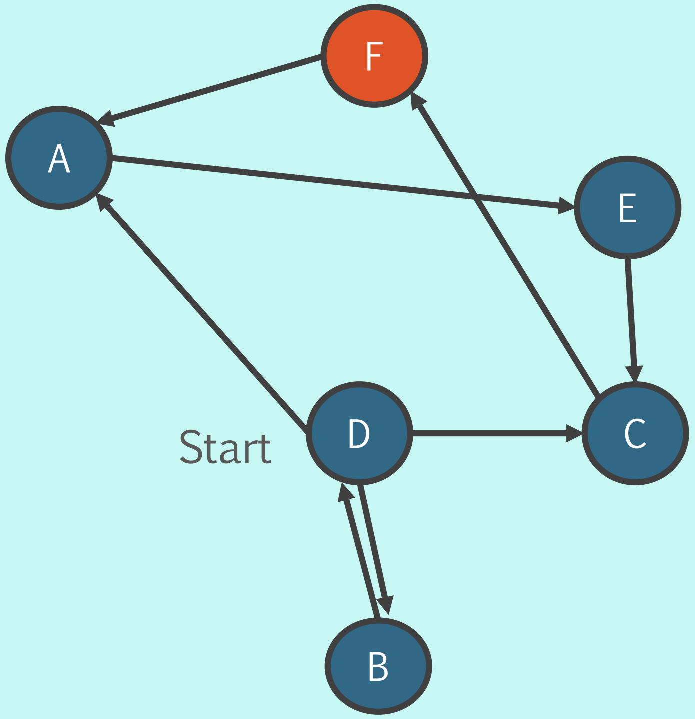
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E

Queue: D, F, C

Curr: E



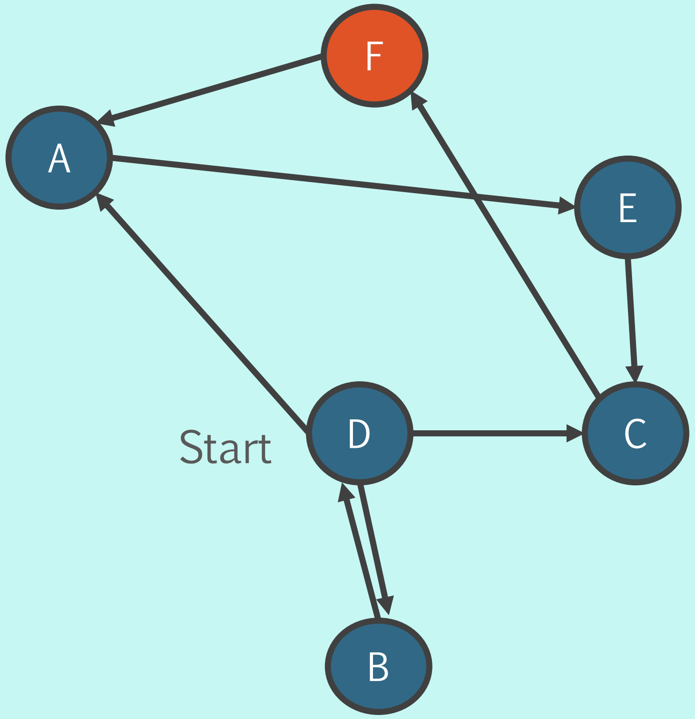
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E

Queue: F, C

Curr: D



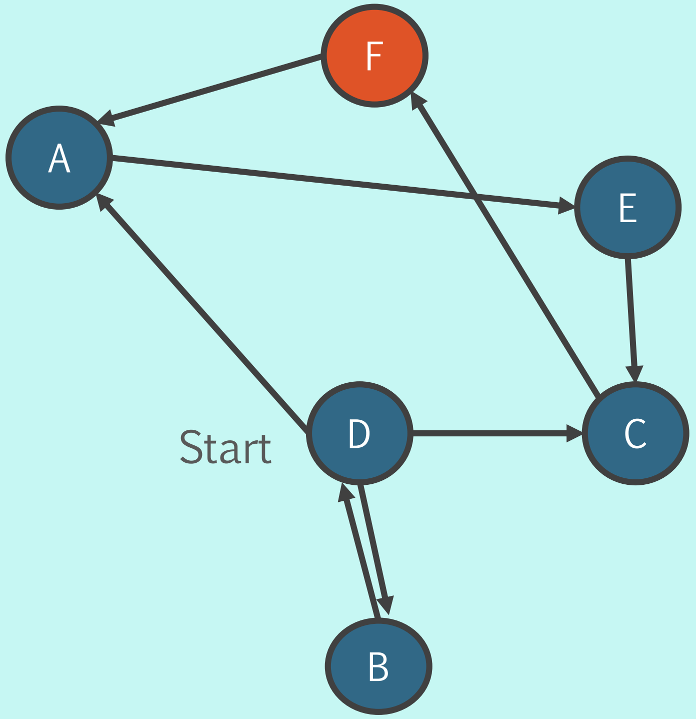
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E

Queue: F, C

Curr: D



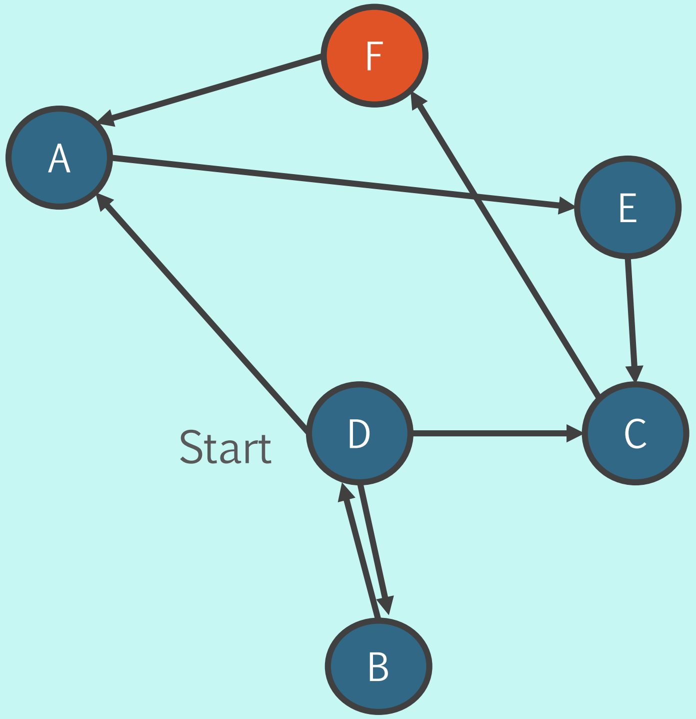
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E

Queue: F, C

Curr: D



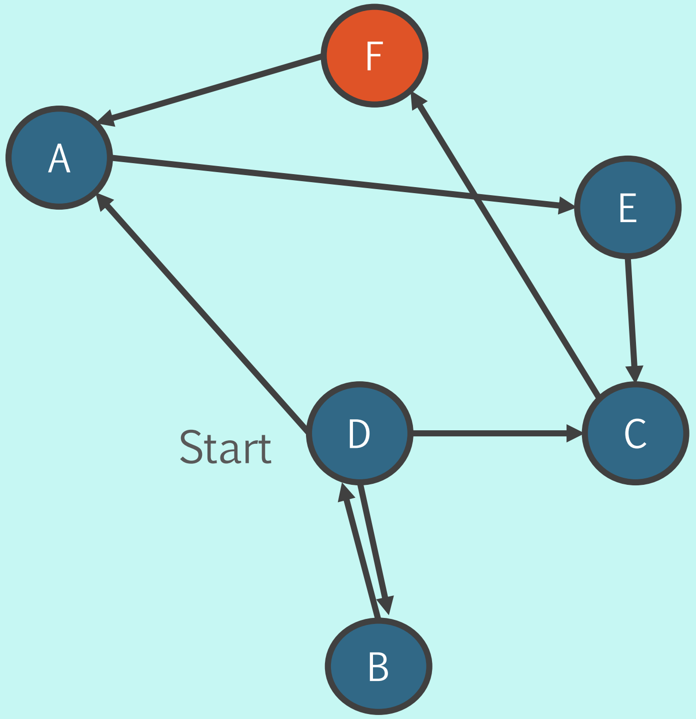
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E

Queue: F, C

Curr: D



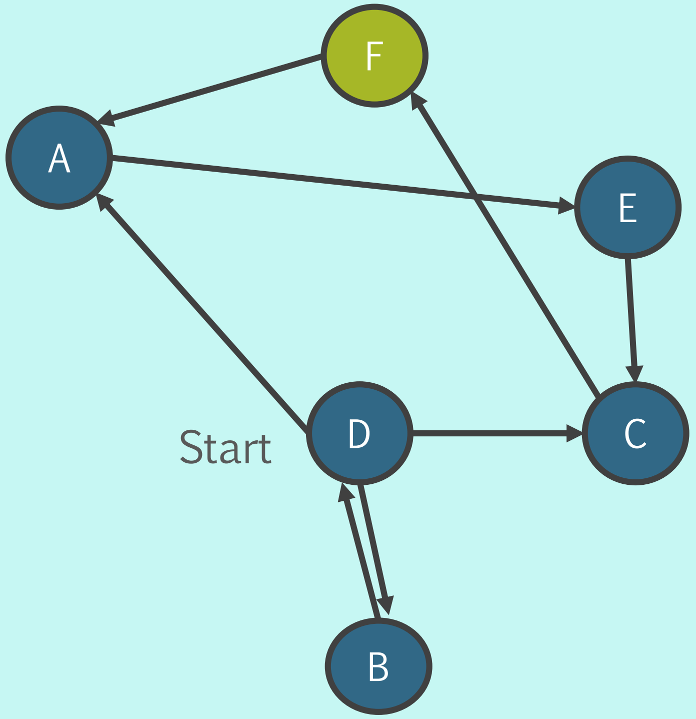
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E

Queue: C

Curr: F



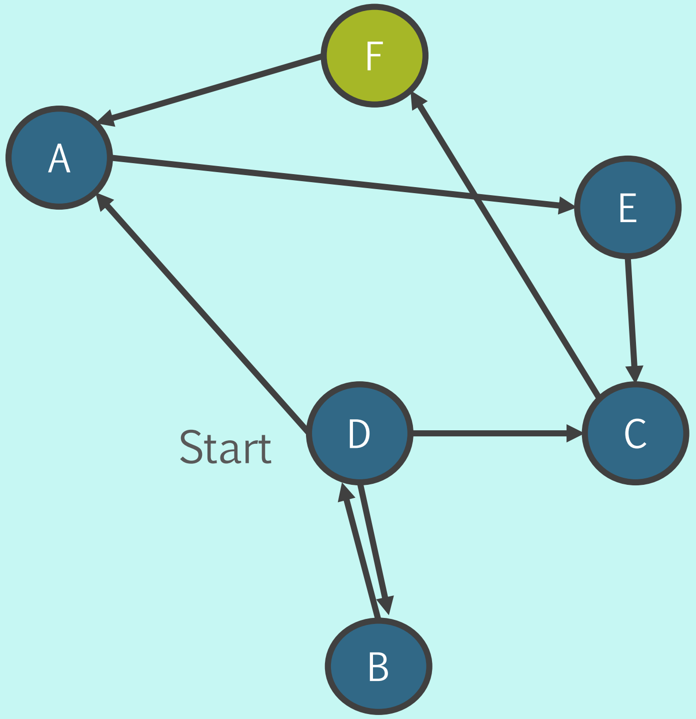
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E

Queue: C

Curr: F



$$\begin{tikzcd}
 & F \arrow[dl] & \\
A \arrow[rr] & & E \arrow[d] \\
 & D \arrow[ul] \arrow[r] \arrow[d, shift left] & C \arrow[uul] \\
 & B \arrow[u, shift left] &
\end{tikzcd}$$

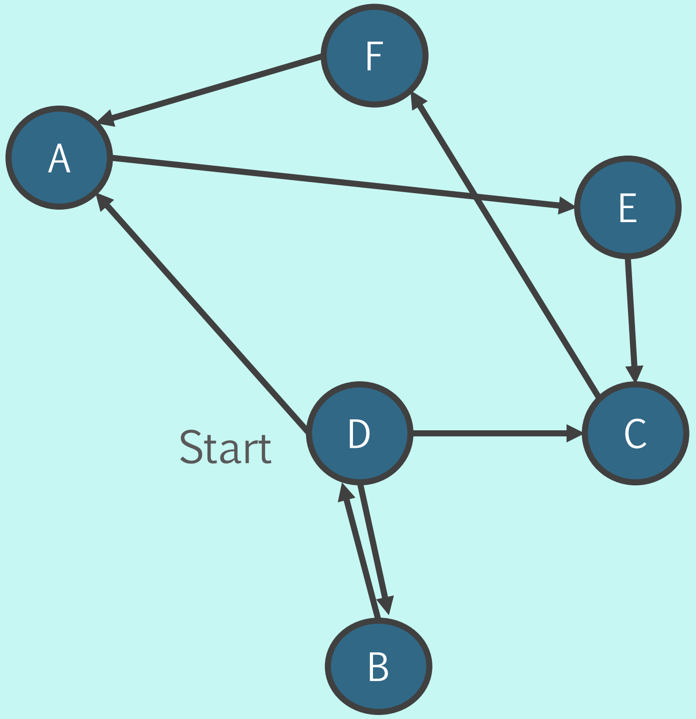
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue: C

Curr: F



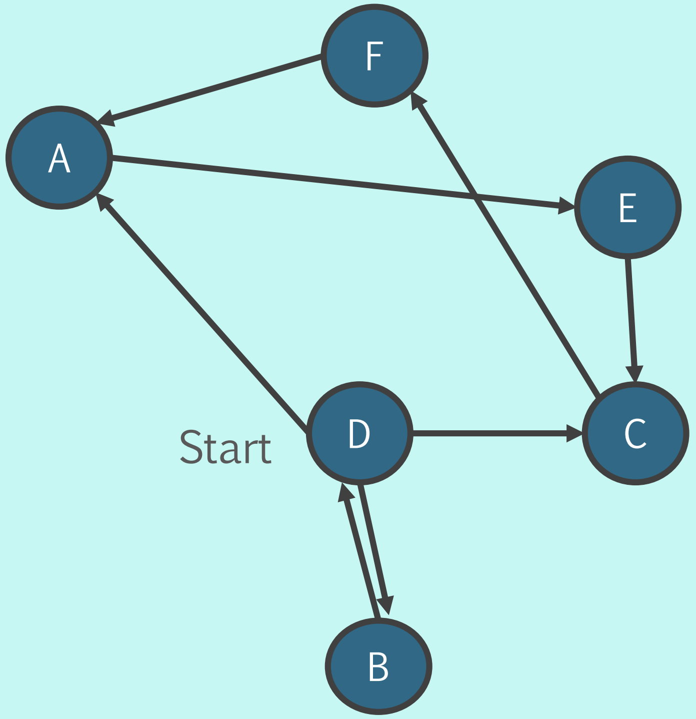
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue: C

Curr: F



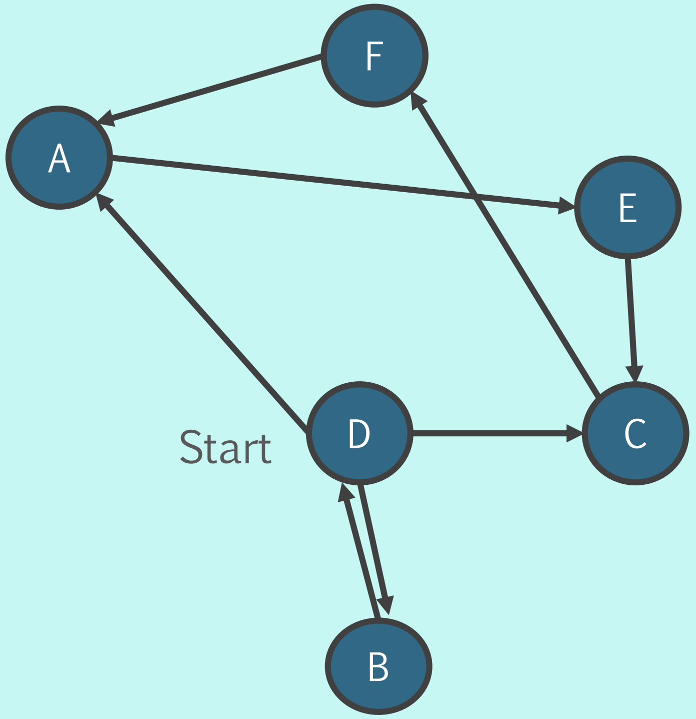
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue: C

Curr: F



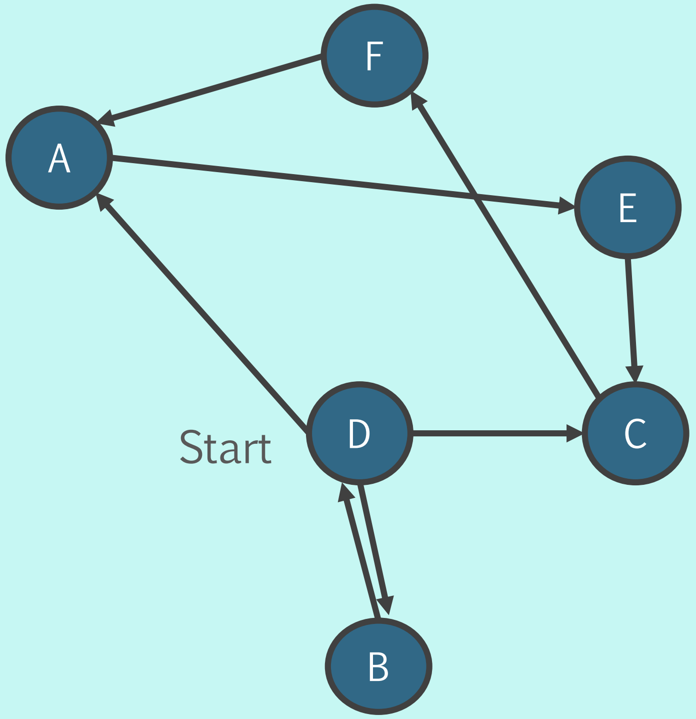
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue: C, A

Curr: F



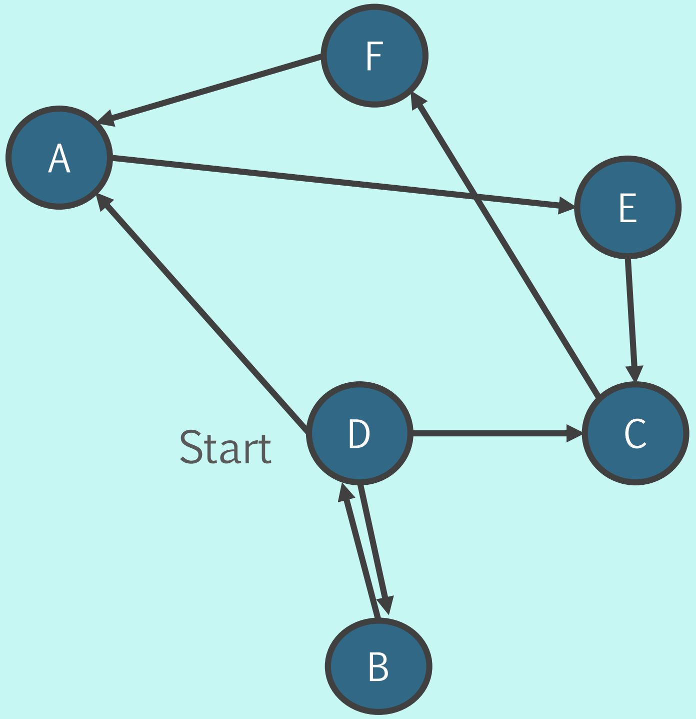
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue: C, A

Curr: F



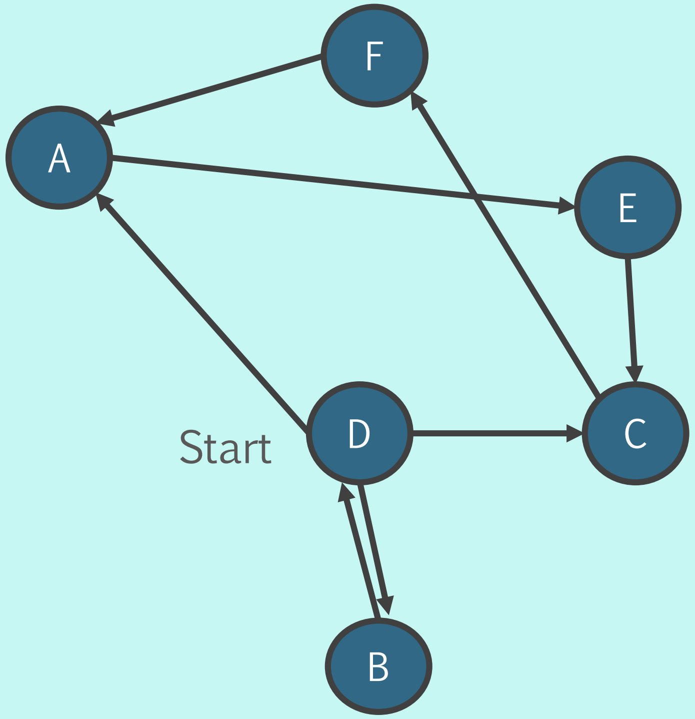
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue: C, A

Curr: F



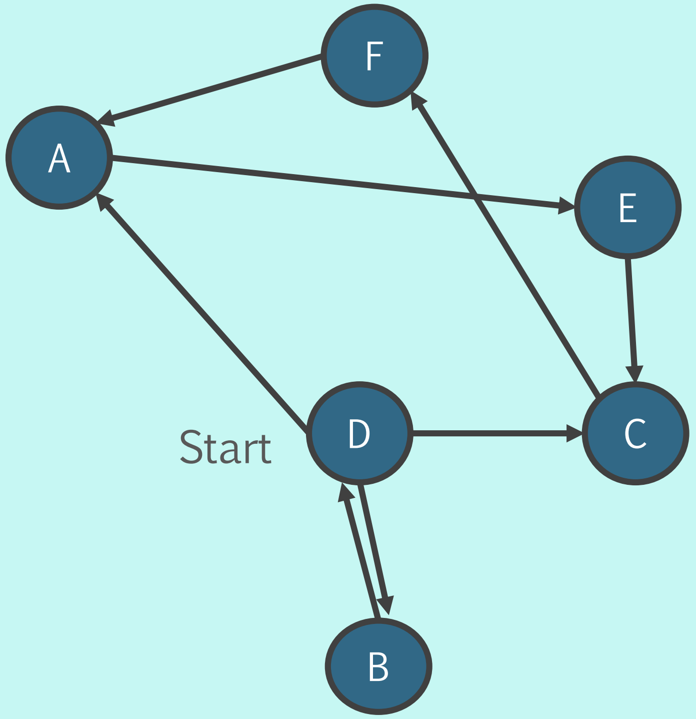
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue: A

Curr: C



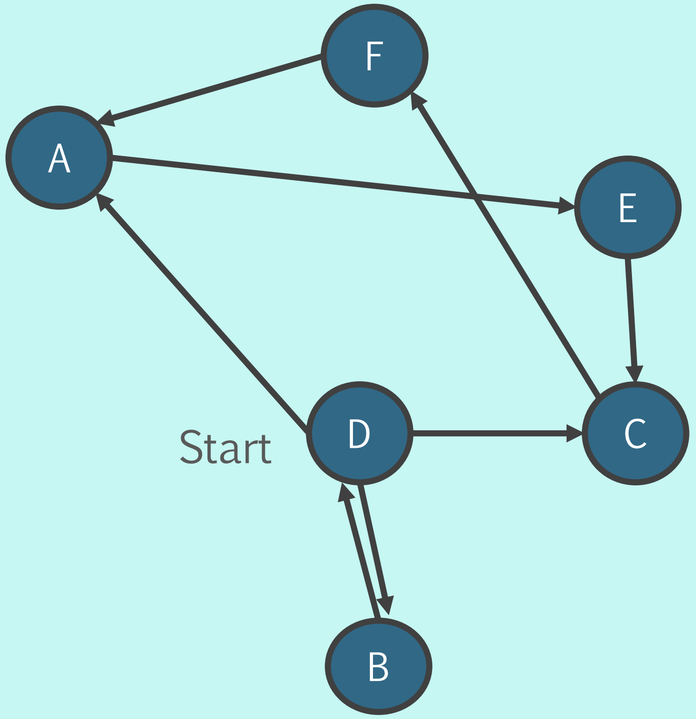
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue: A

Curr: C



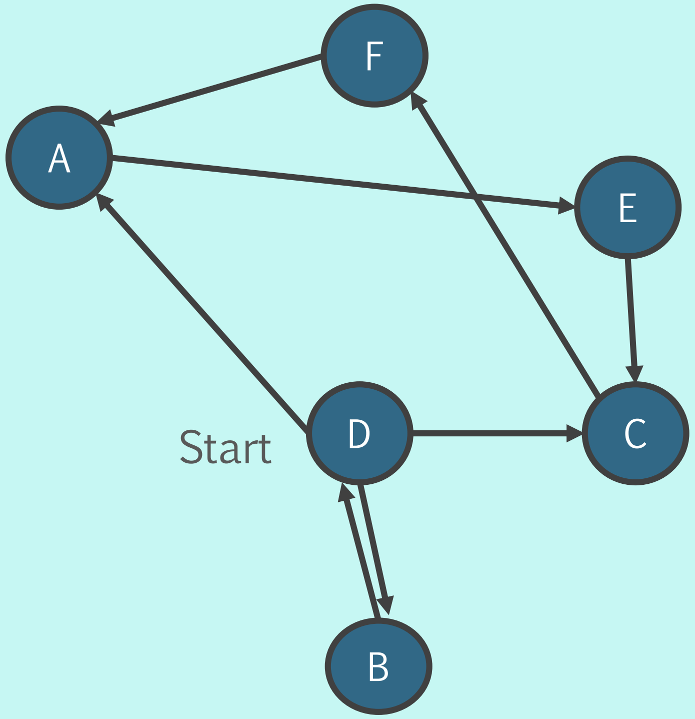
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

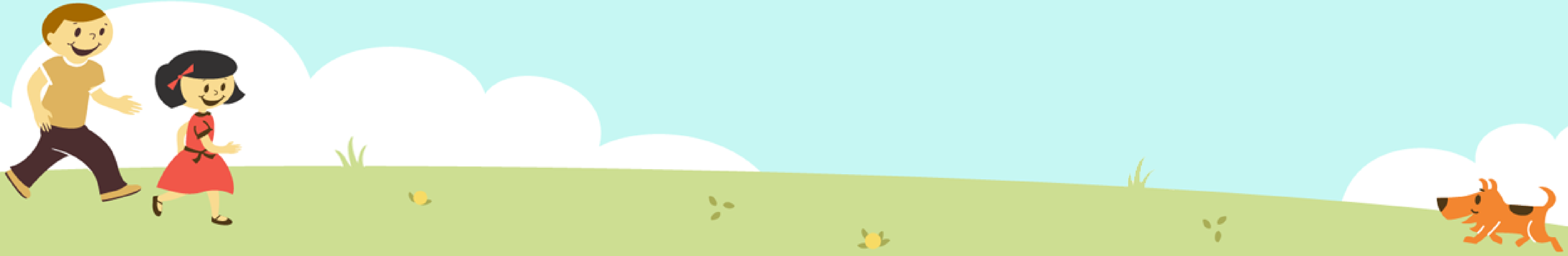
Visited: D, A, B, C, E, F

Queue: A

Curr: C



```
GraphSearch(start, goal)
  **Set** visited
  **Queue** s
  s.add(start)
  **while** (s **not** empty)
    curr = s.remove()
    **if** (curr **is** visited)
      **continue**
    visited.add(curr)
    evaluate(curr) // check if goal
    **for** Vertex u **in** neighbors(curr)
      s.add(u)
```

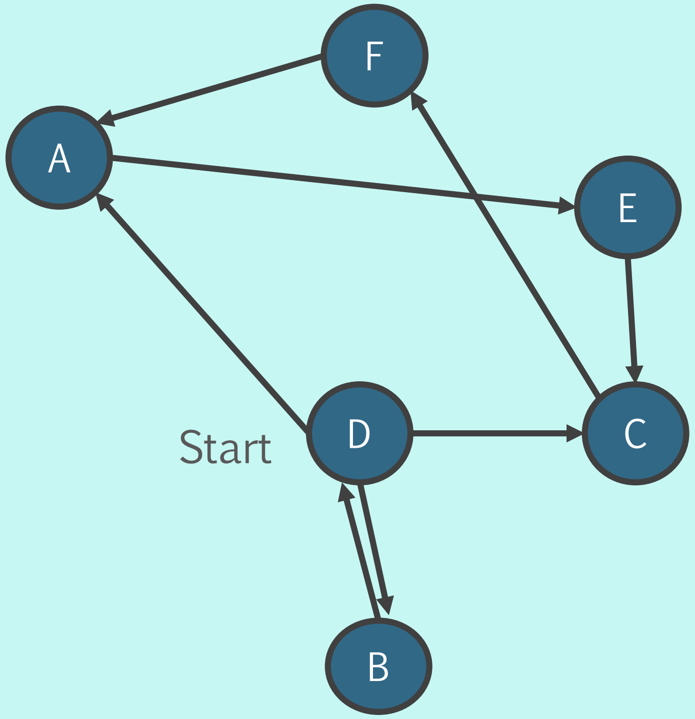


Breadth First Search Example

Visited: D, A, B, C, E, F

Queue: A

Curr: C



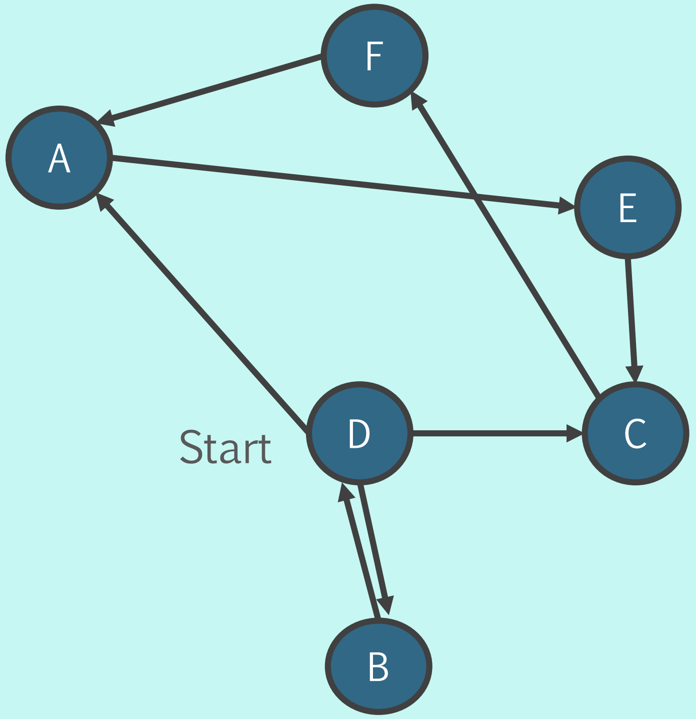
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue:

Curr: A



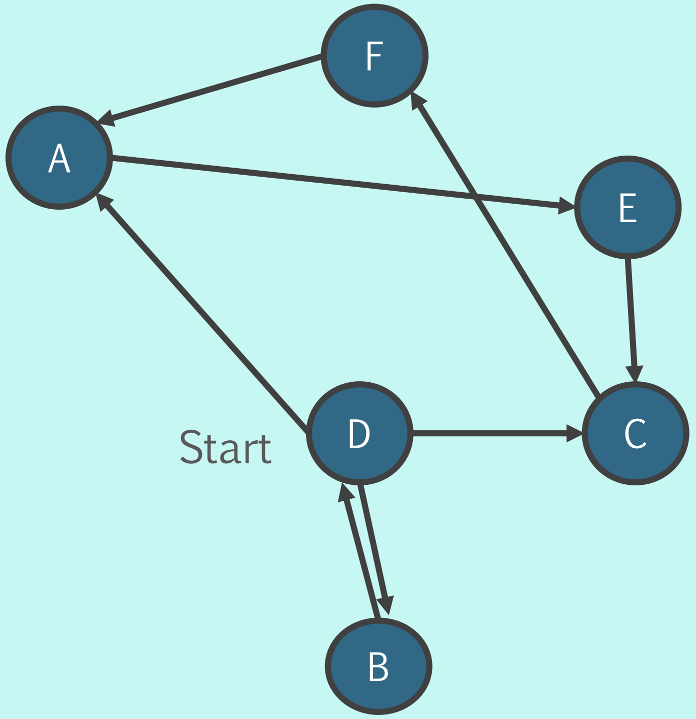
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue:

Curr: A



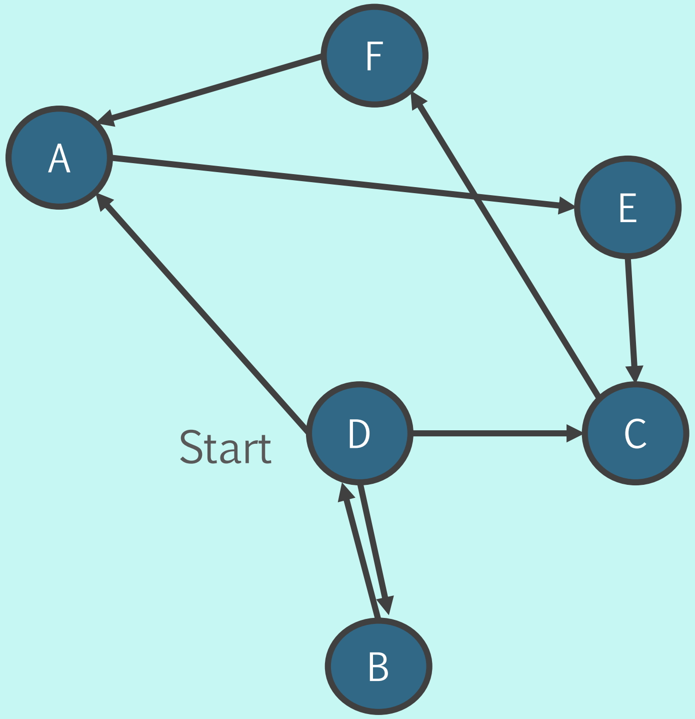
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue:

Curr: A



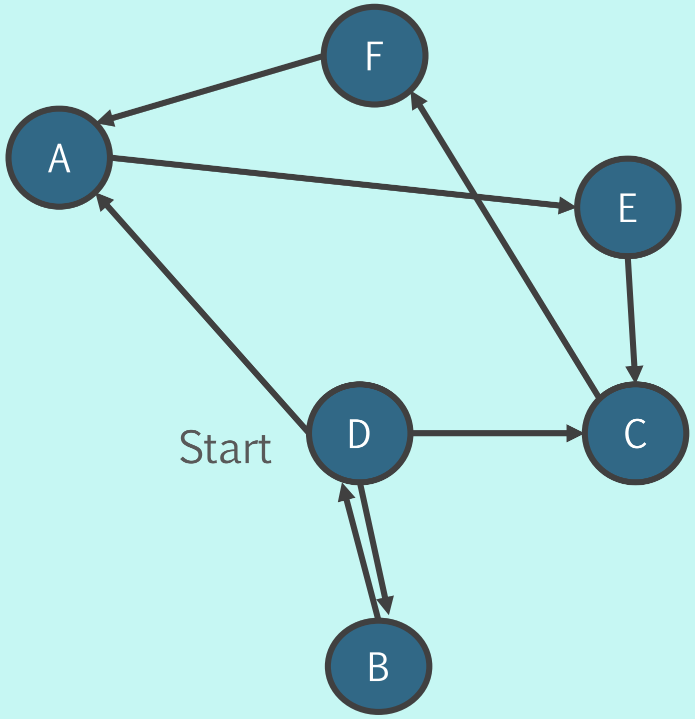
```
GraphSearch(start, goal)
  **Set** visited
  **Queue** s
  s.add(start)
  **while** (s **not** empty)
    curr = s.remove()
    **if** (curr **is** visited)
      **continue**
    visited.add(curr)
    evaluate(curr) // check if goal
    **for** Vertex u **in** neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue:

Curr: A



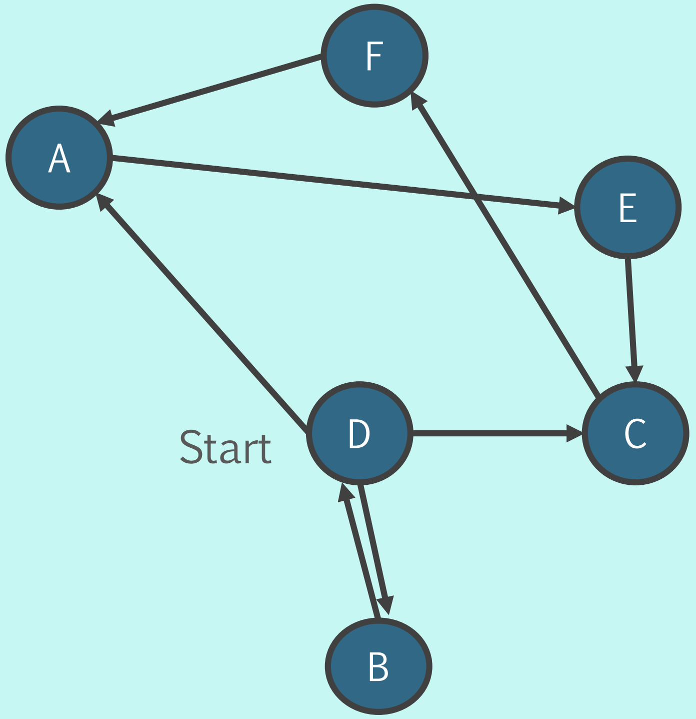
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue:

Curr: A



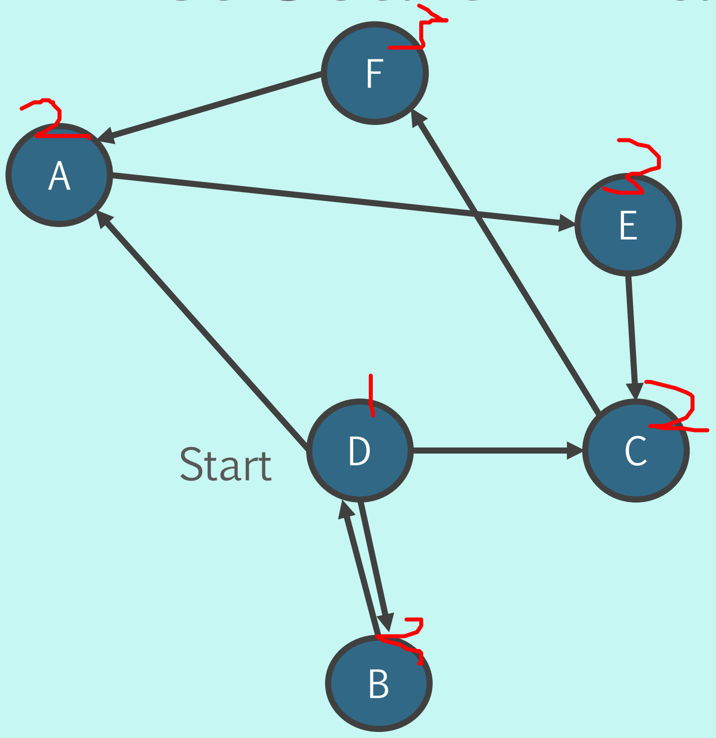
```
GraphSearch(start, goal)
   Set visited
   Queue s
   s.add(start)
   while (s not empty)
      curr = s.remove()
      if (curr is visited)
         continue
      visited.add(curr)
      evaluate(curr) // check if goal
      for Vertex u in neighbors(curr)
         s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue:

Curr: A



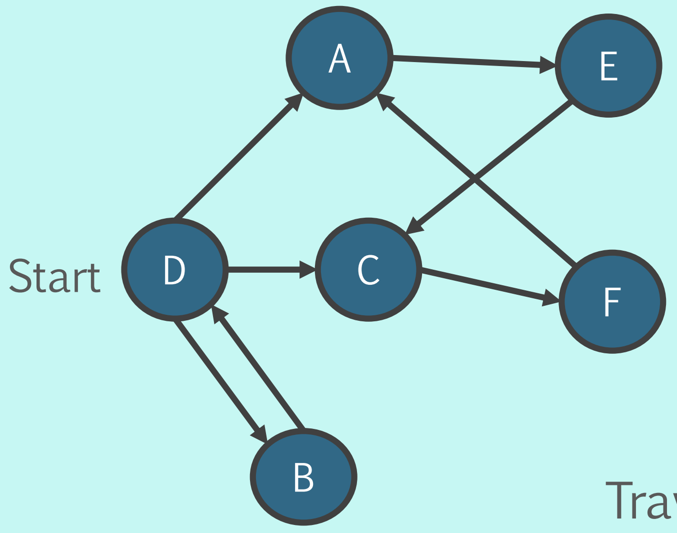
```
GraphSearch(start, goal)
  Set visited
  Queue s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue:

Curr: A



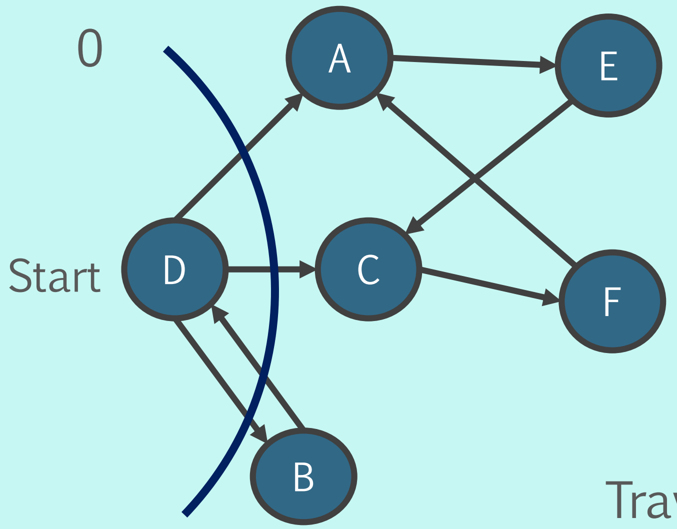
Traverses by edge distance from start

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue:

Curr: A



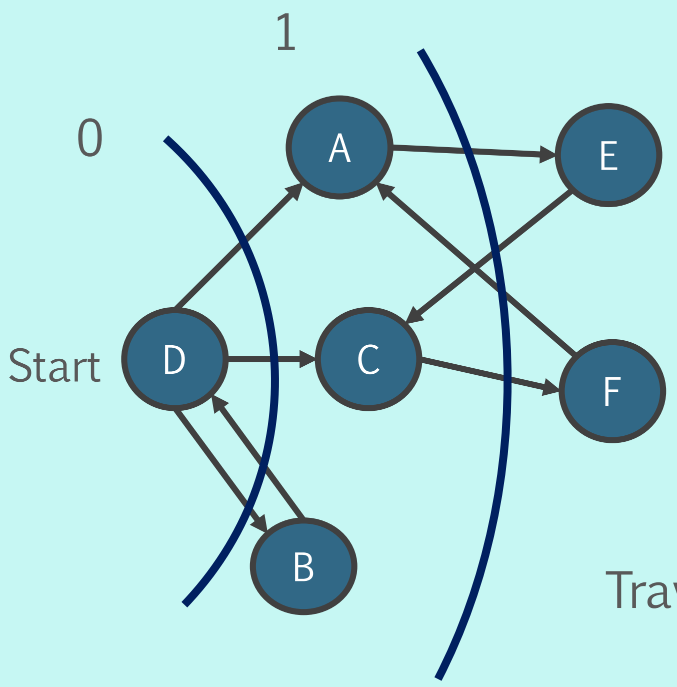
Traverses by edge distance from start

Breadth First Search Example

Visited: D, A, B, C, E, F

Queue:

Curr: A



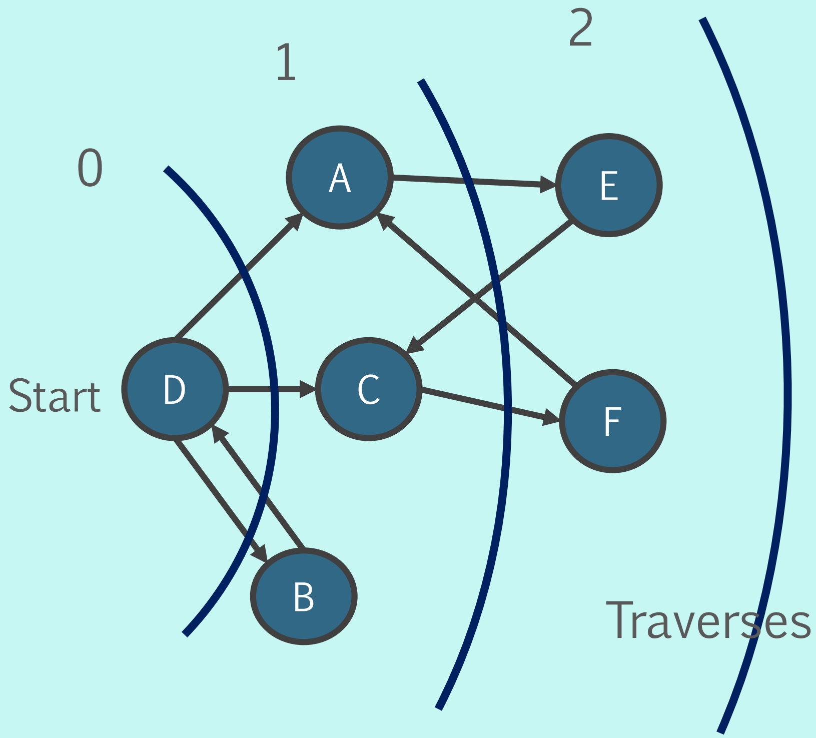
Traverses by edge distance from start

Breadth First Search Example

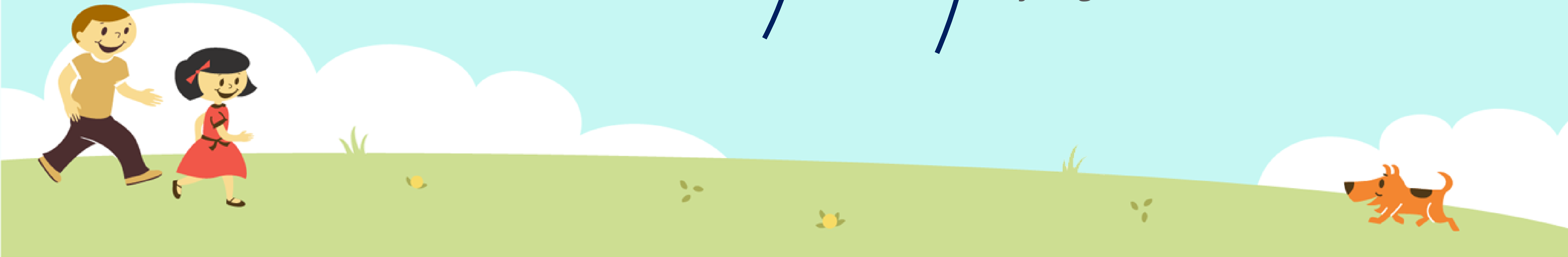
Visited: D, A, B, C, E, F

Queue:

Curr: A

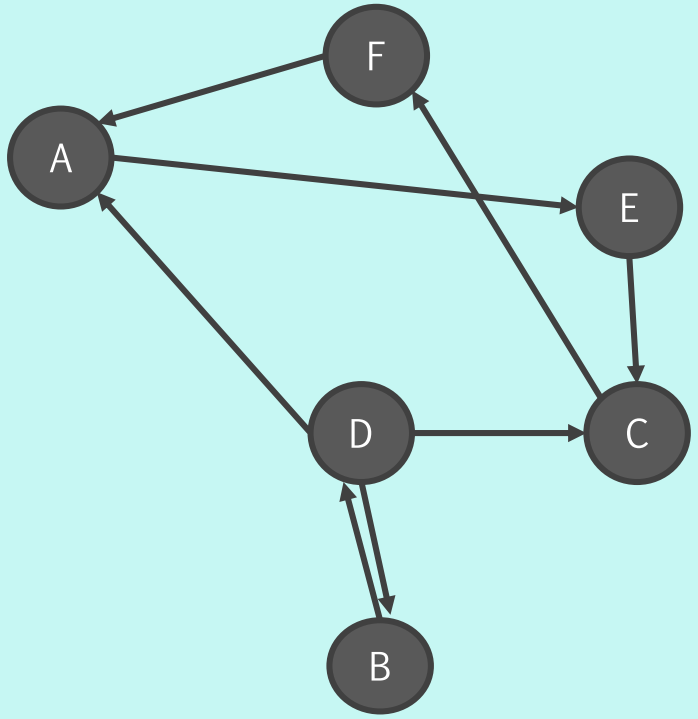


Traverses by edge distance from start



Depth First Search

- Using the General Graph Search Algorithm, **Structure s = Stack**
- Starting at a vertex v, the search will go down one path as far as it can until it hits a dead end.   It then backtracks till it can go down another path.

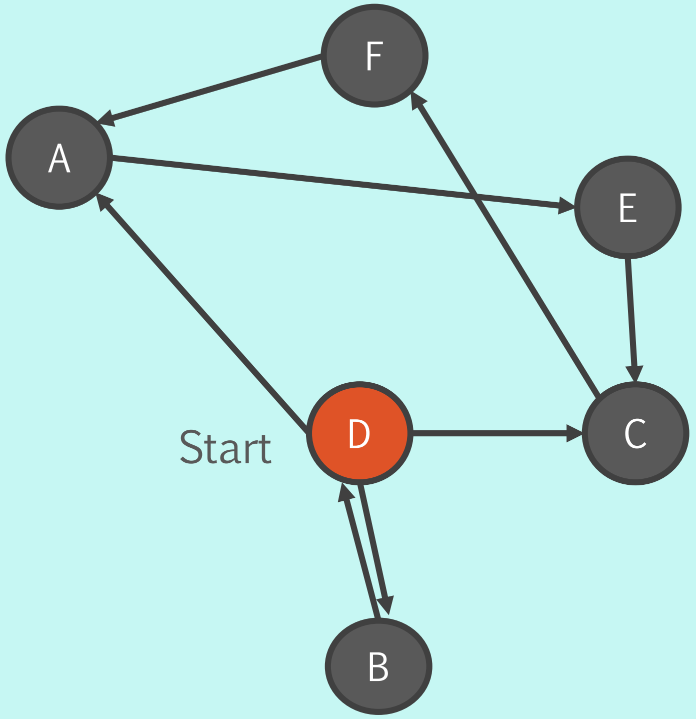


Depth First Search Example

Visited:

Stack: D

Curr:



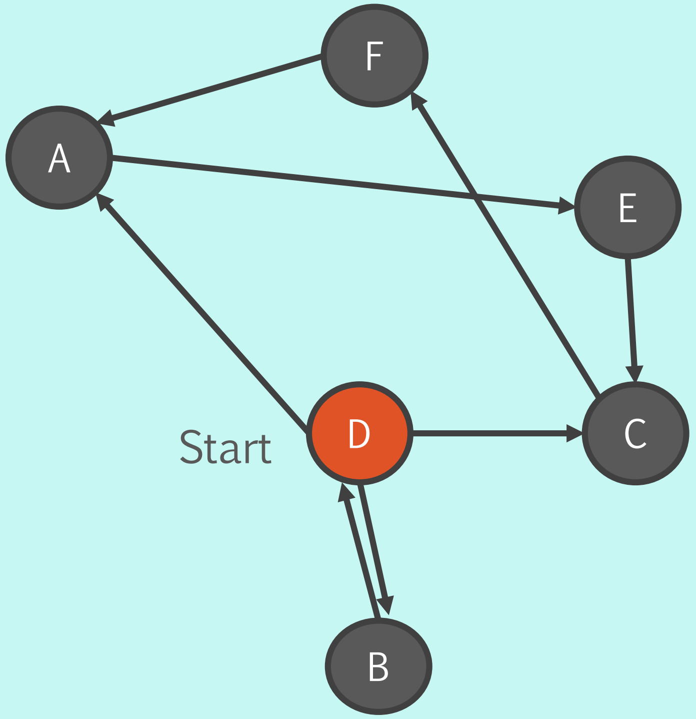
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited:

Stack: D

Curr:



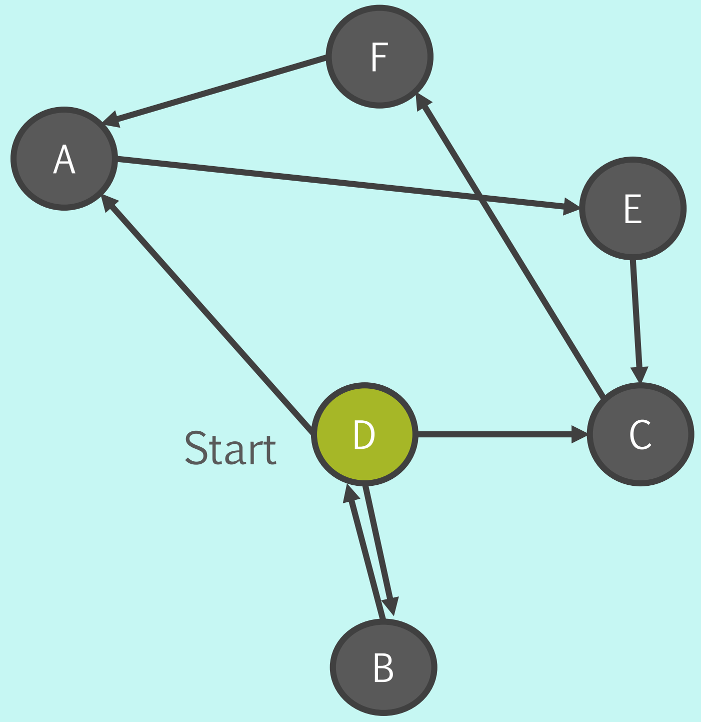
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited:

Stack:

Curr: D



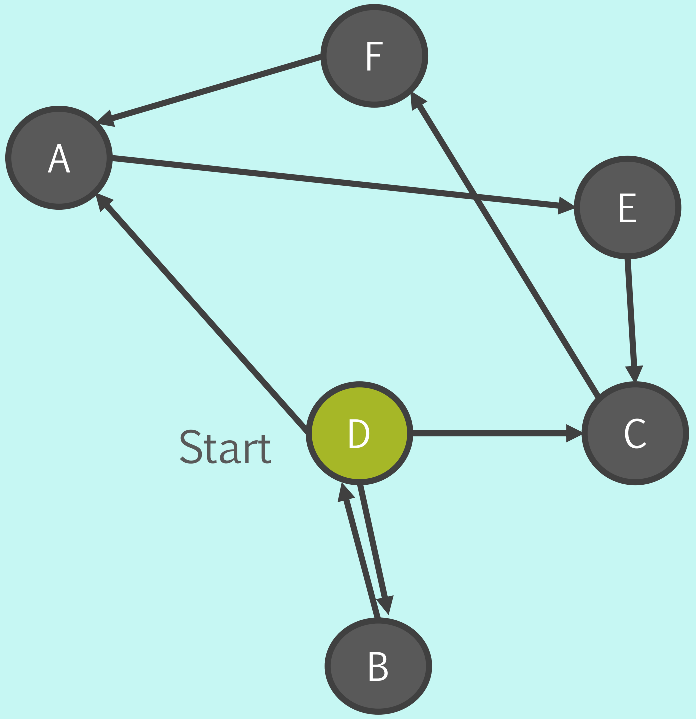
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited:

Stack:

Curr: D



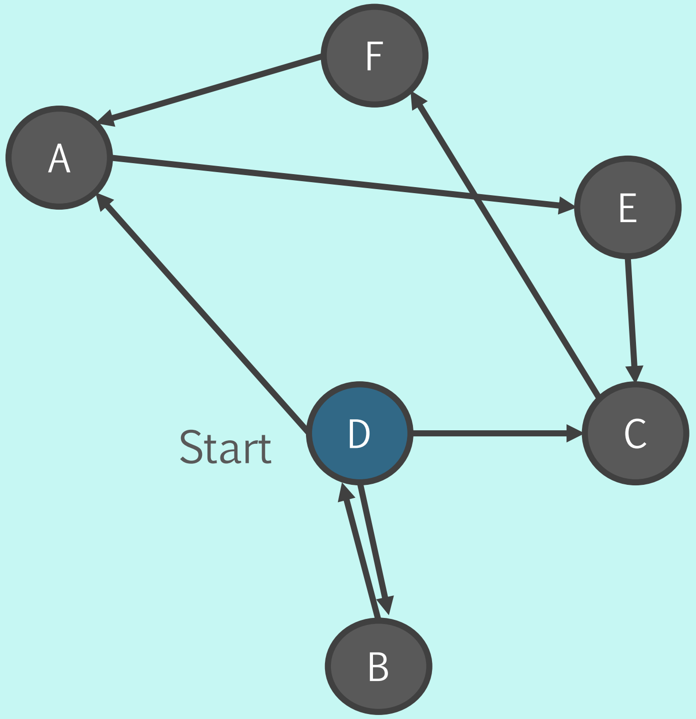
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D

Stack:

Curr: D



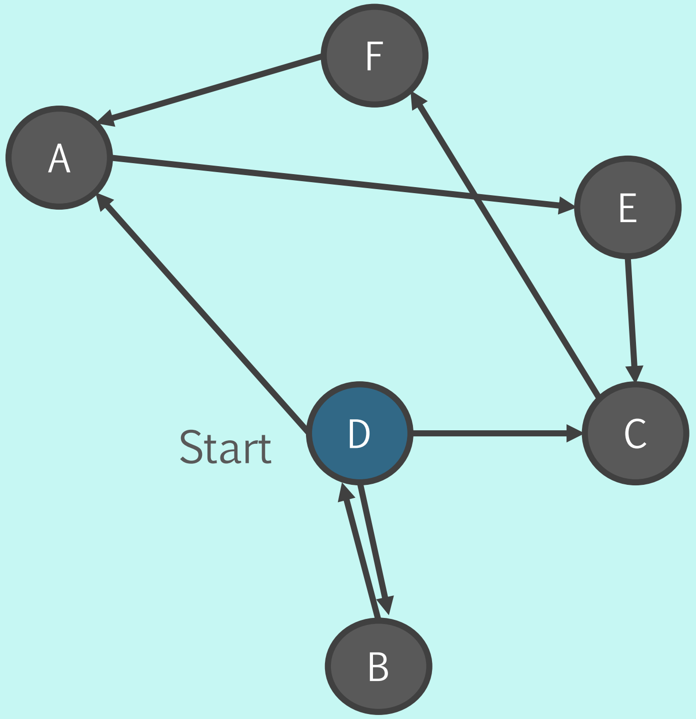
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D

Stack:

Curr: D



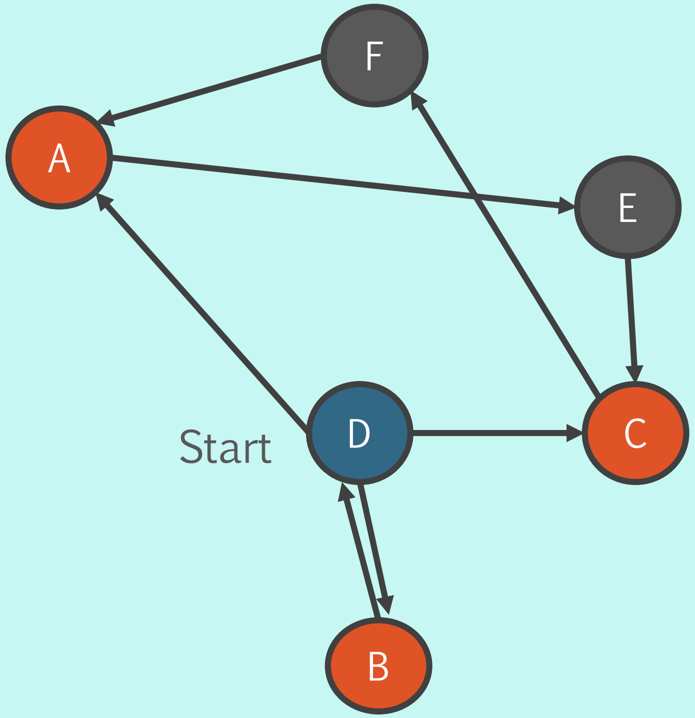
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D

Stack: A, B, C

Curr: D



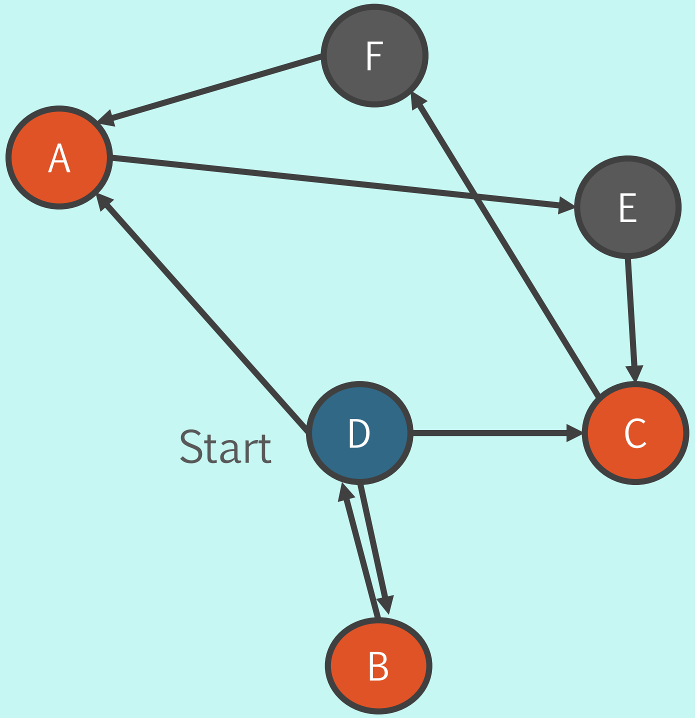
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D

Stack: A, B, C

Curr: D



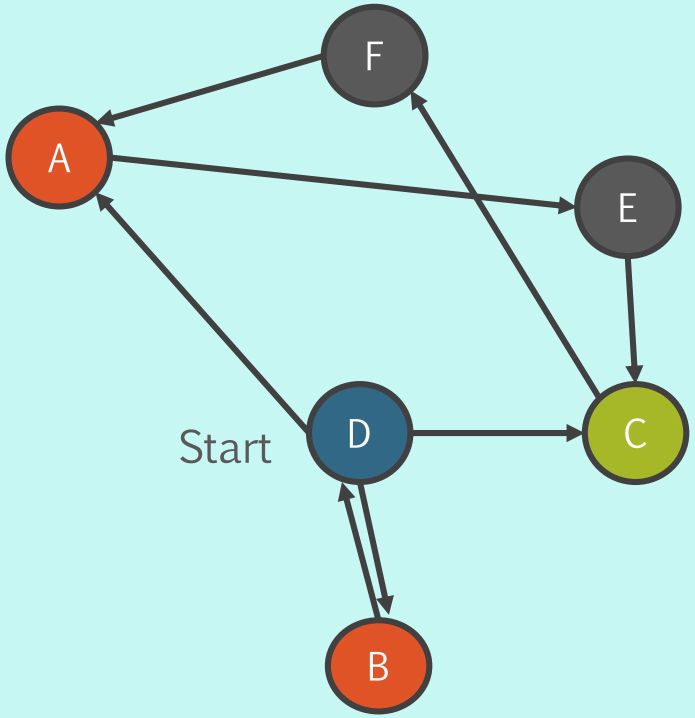
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D

Stack: A, B,

Curr: C



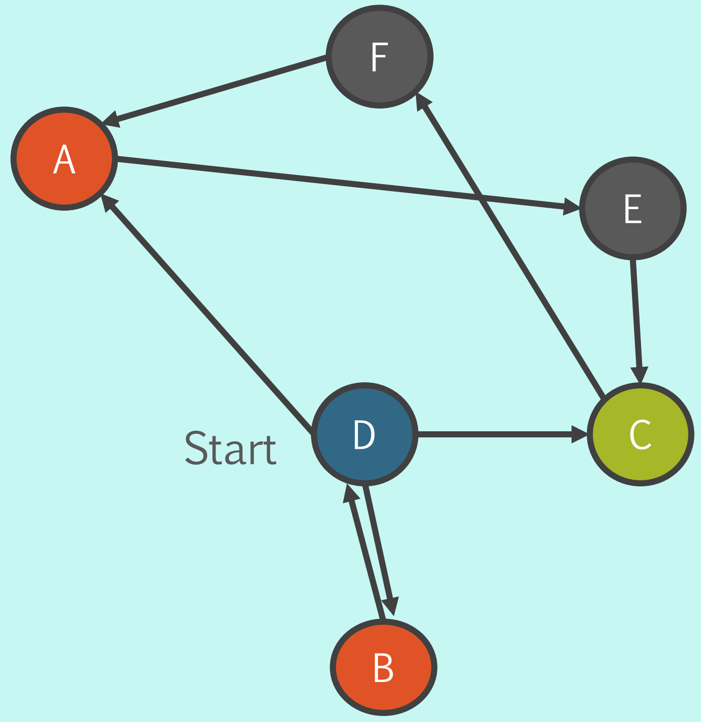
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D

Stack: A, B

Curr: C



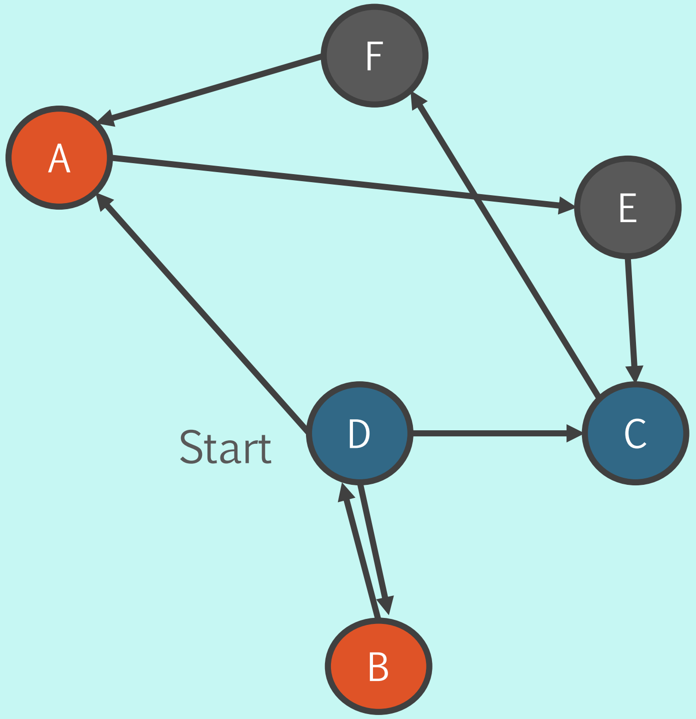
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C

Stack: A, B

Curr: C



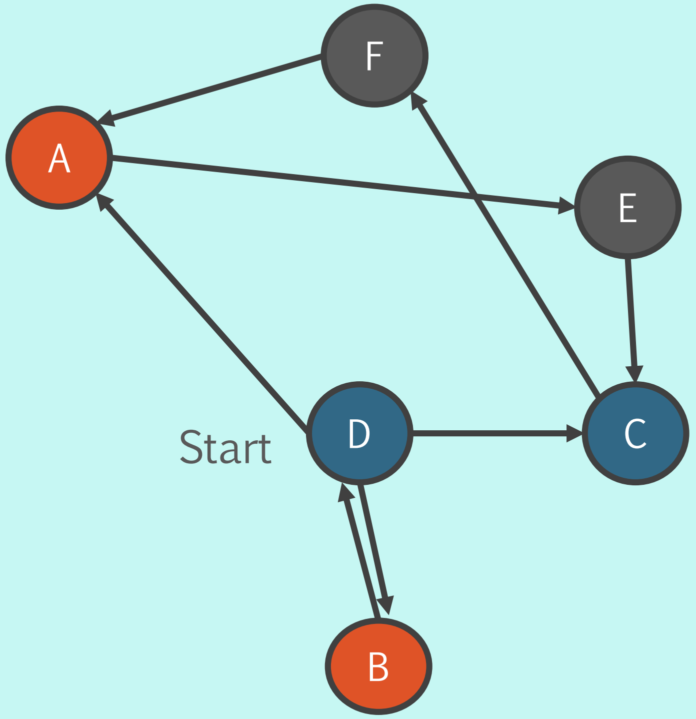
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C

Stack: A, B

Curr: C



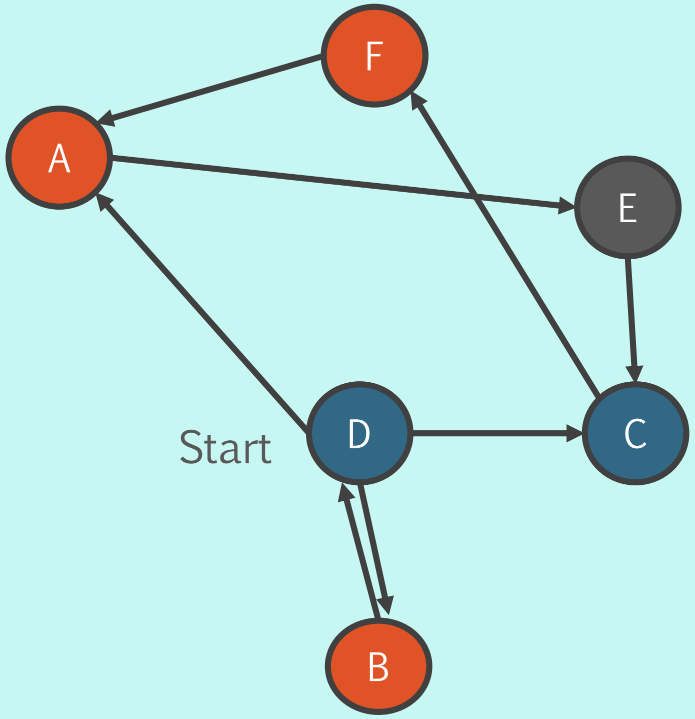
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C

Stack: A, B, F

Curr: C



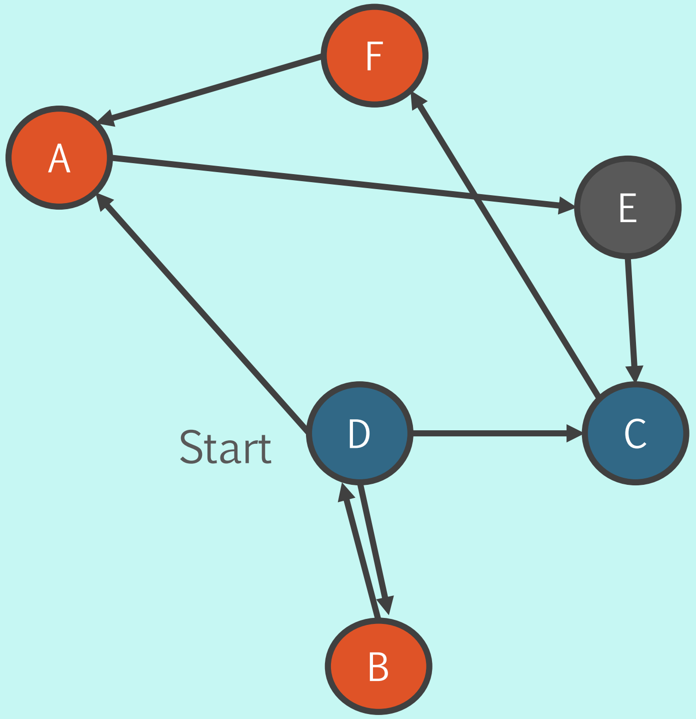
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C

Stack: A, B, F

Curr: C



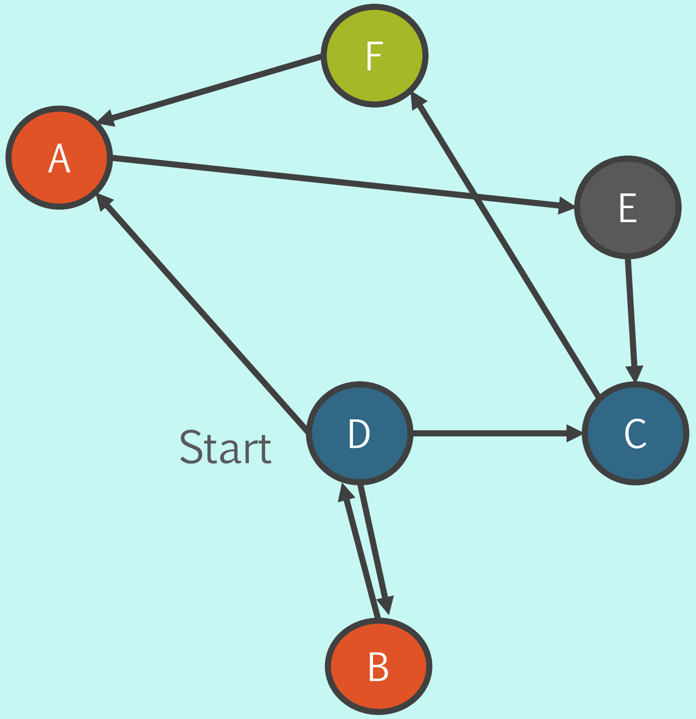
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C

Stack: A, B,

Curr: F



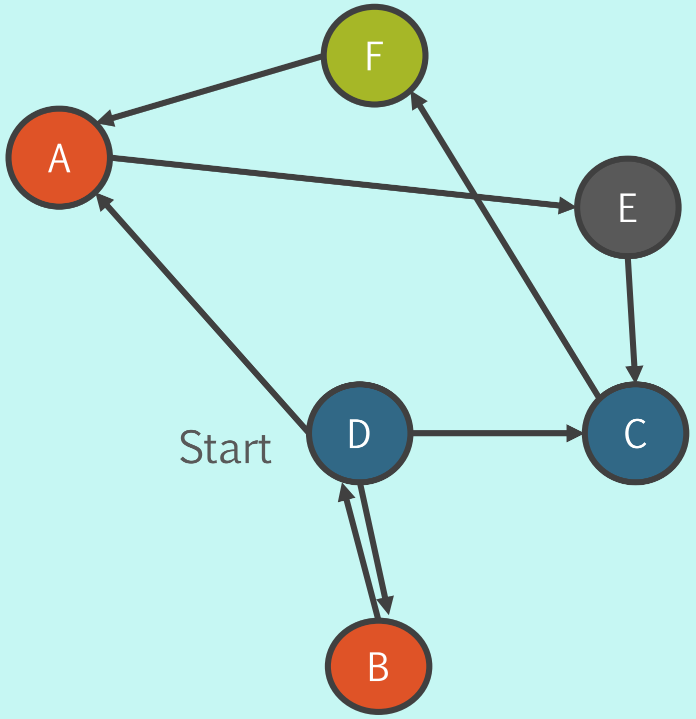
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C

Stack: A, B,

Curr: F



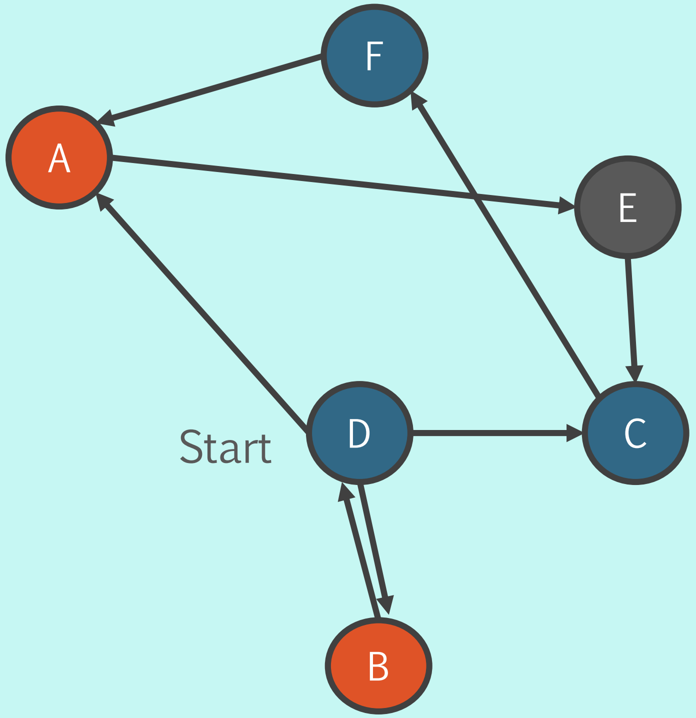
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F

Stack: A, B,

Curr: F



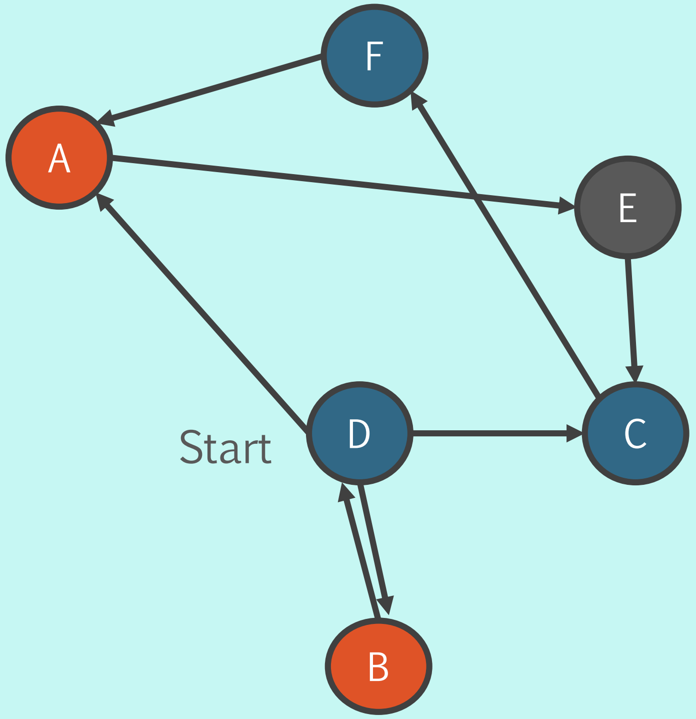
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F

Stack: A, B,

Curr: F



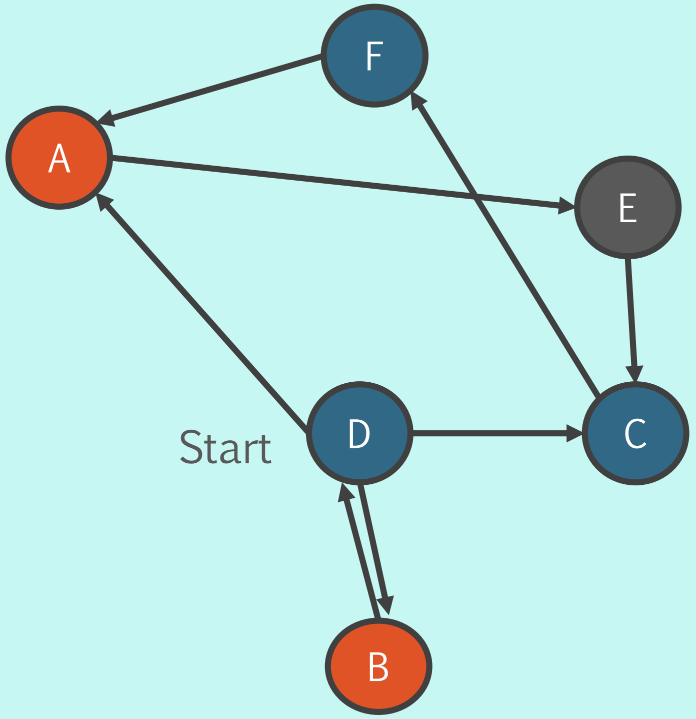
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F

Stack: A, B, A

Curr: F



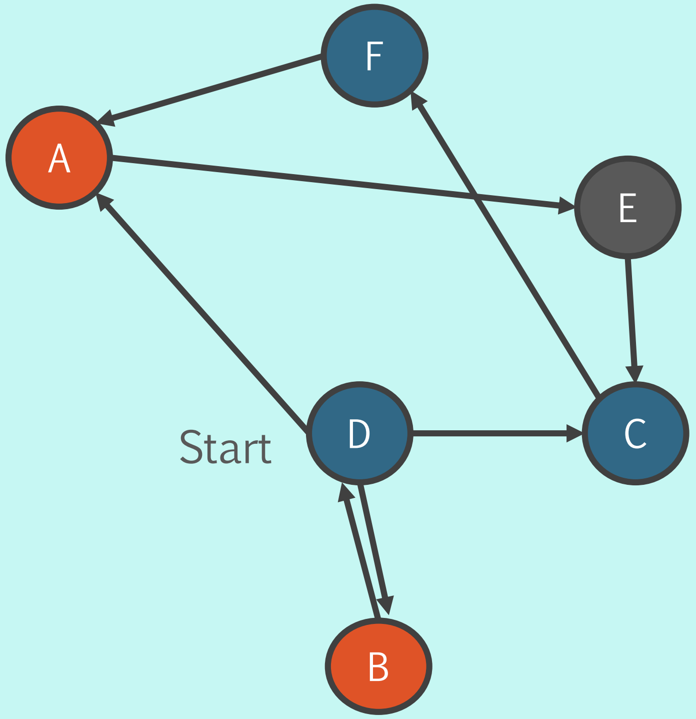
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F

Stack: A, B, A

Curr: F



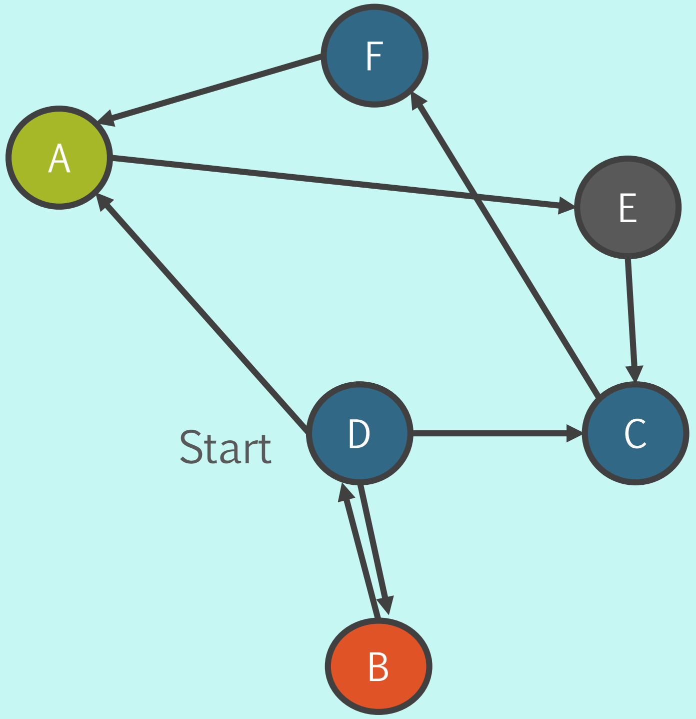
```
GraphSearch(start, goal)
   Set visited
   Stack s
   s.add(start)
   while (s not empty)
      curr = s.remove()
      if (curr is visited)
         continue
      visited.add(curr)
      evaluate(curr) // check if goal
      for Vertex u in neighbors(curr)
         s.add(u)
```

Depth First Search Example

Visited: D, C ,F

Stack: A, B

Curr: A



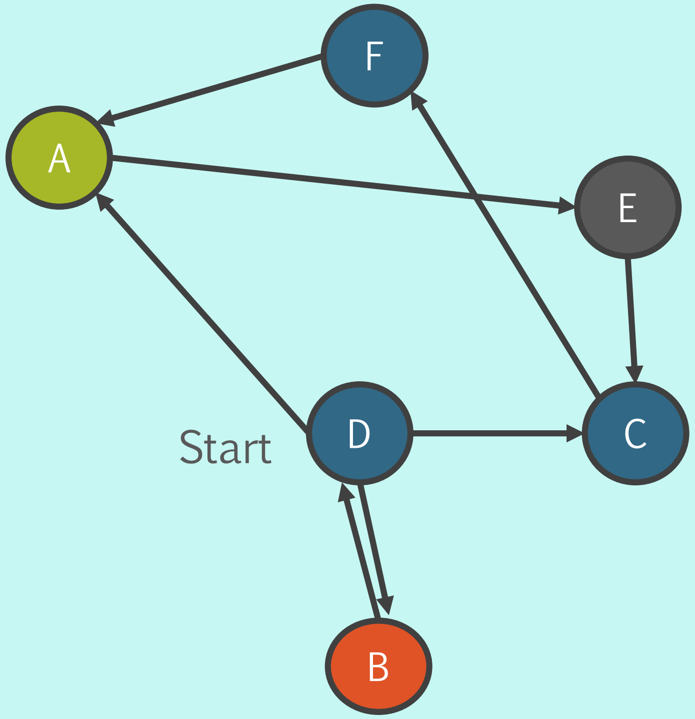
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F

Stack: A, B

Curr: A



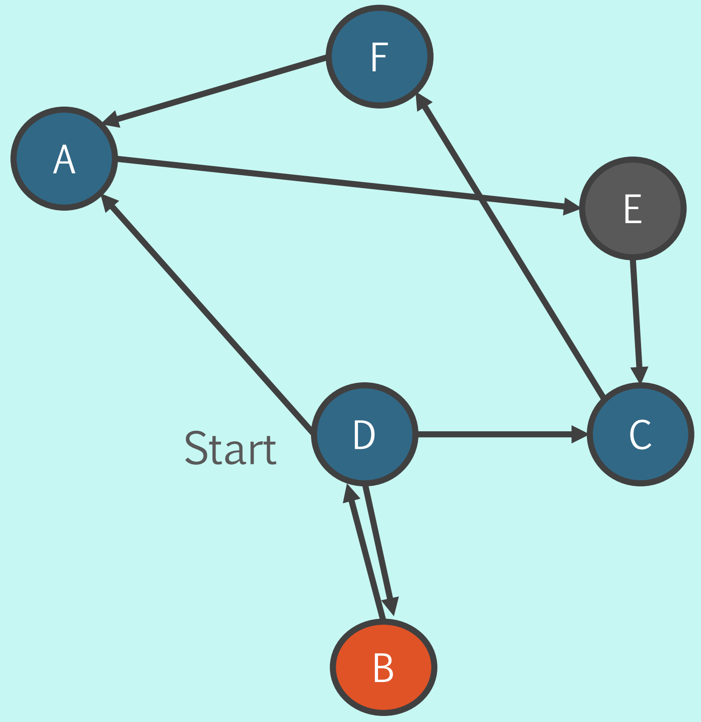
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A

Stack: A, B

Curr: A



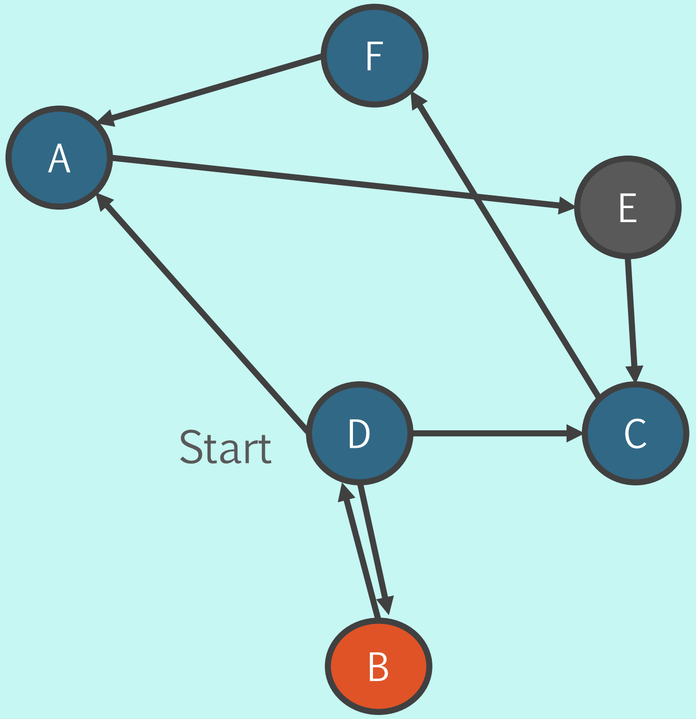
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A

Stack: A, B

Curr: A



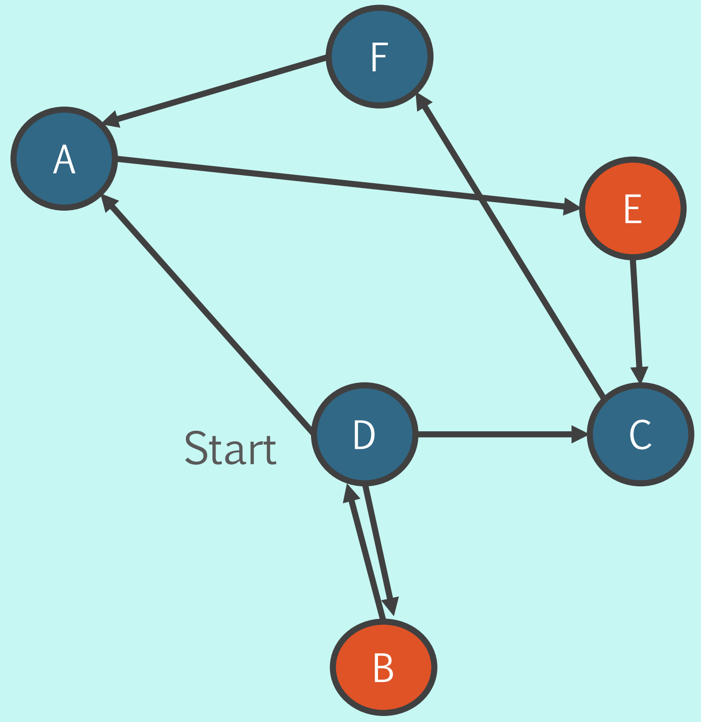
```
GraphSearch(start, goal)
   Set visited
   Stack s
   s.add(start)
   while (s not empty)
      curr = s.remove()
      if (curr is visited)
         continue
      visited.add(curr)
      evaluate(curr) // check if goal
      for Vertex u in neighbors(curr)
         s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A

Stack: A, B, E

Curr: A



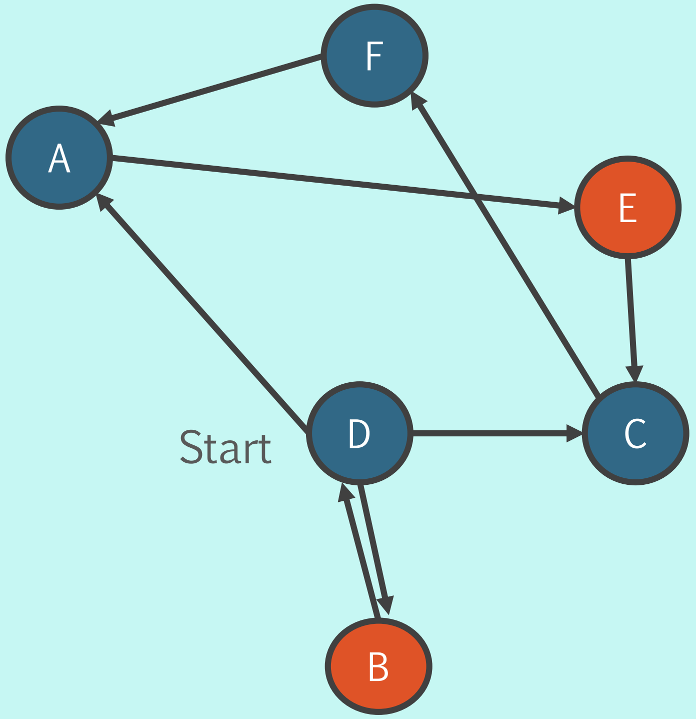
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A

Stack: A, B, E

Curr: A



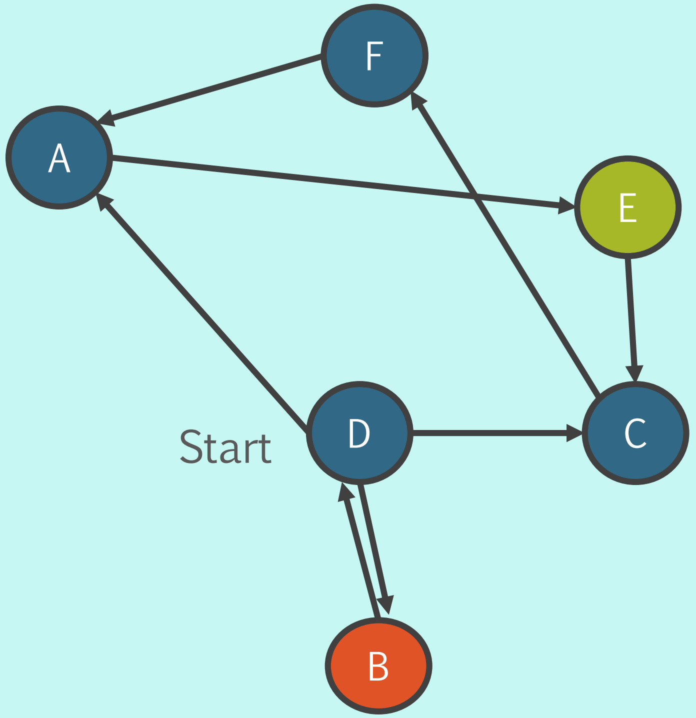
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A

Stack: A, B

Curr: E



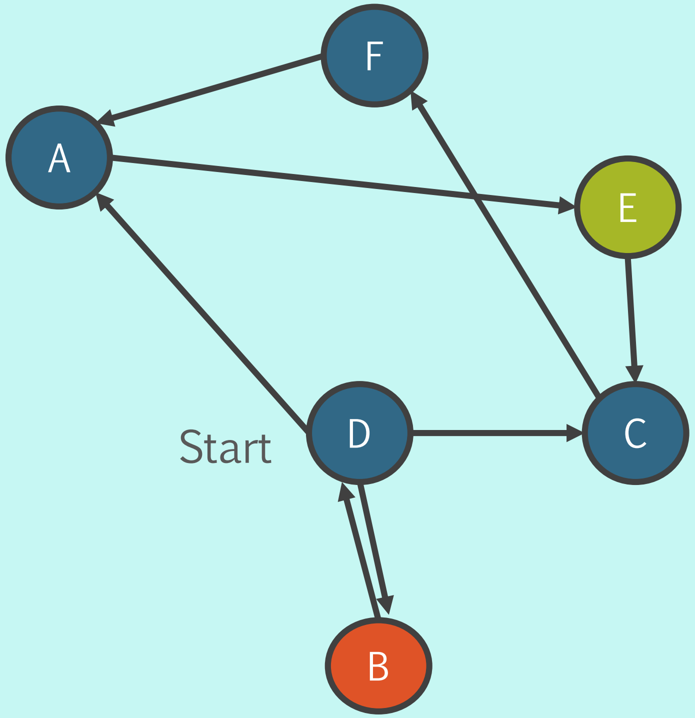
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A

Stack: A, B

Curr: E



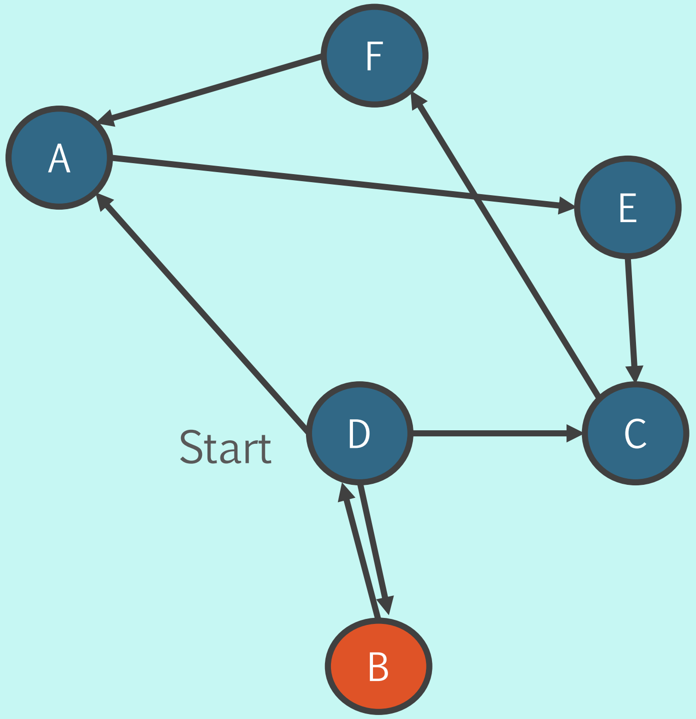
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E

Stack: A, B

Curr: E



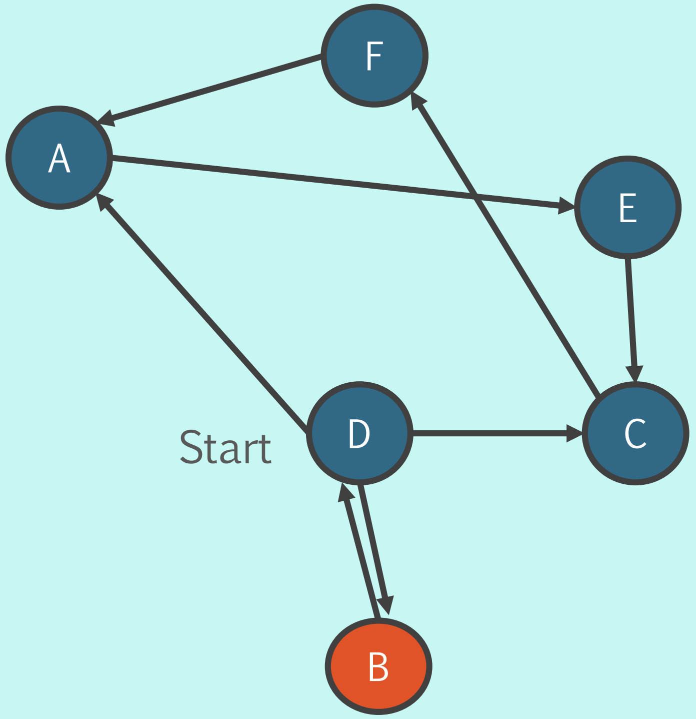
```
GraphSearch(start, goal)
   Set visited
   Stack s
   s.add(start)
   while (s not empty)
      curr = s.remove()
      if (curr is visited)
         continue
      visited.add(curr)
      evaluate(curr) // check if goal
      for Vertex u in neighbors(curr)
         s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E

Stack: A, B

Curr: E



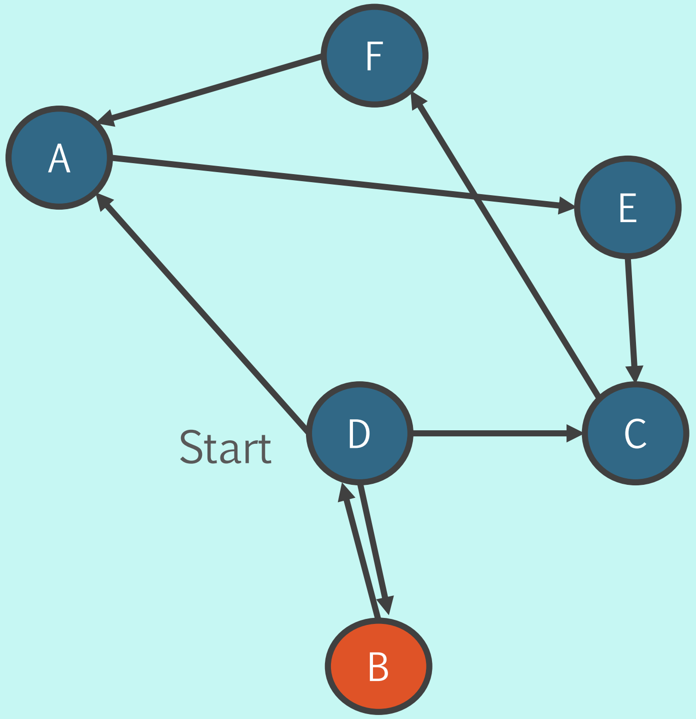
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E

Stack: A, B, C

Curr: E



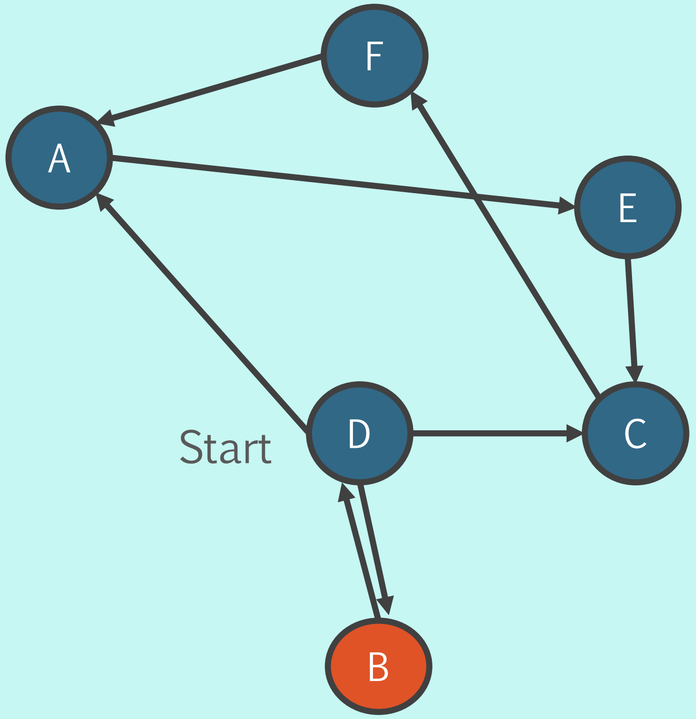
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E

Stack: A, B, C

Curr: E



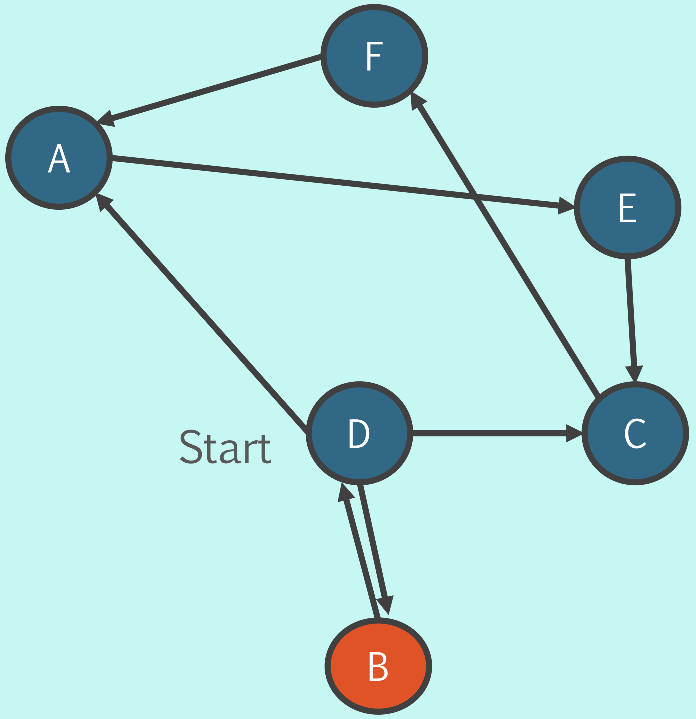
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E

Stack: A, B,

Curr: C



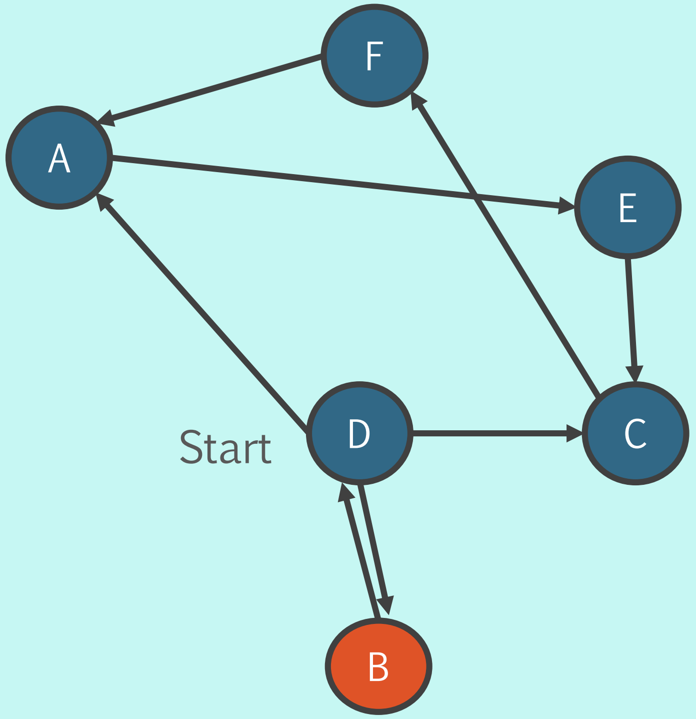
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E

Stack: A, B,

Curr: C



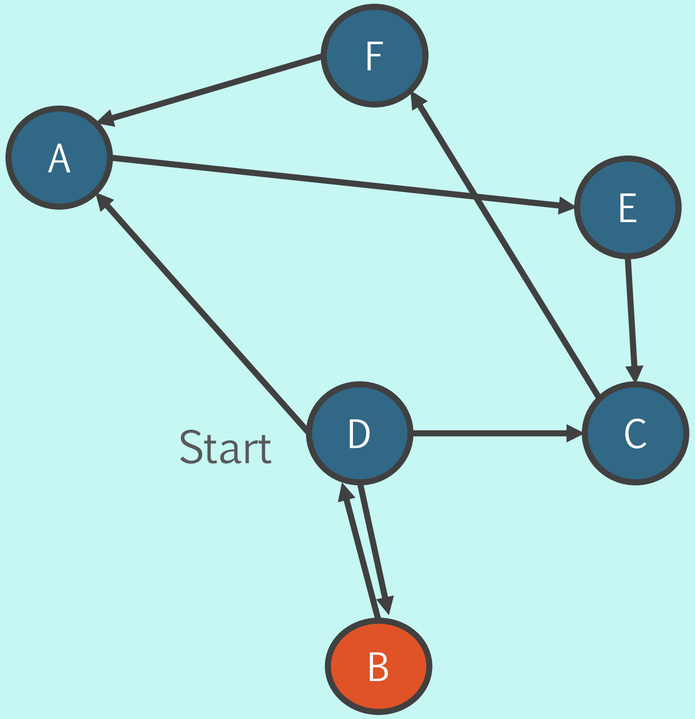
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E

Stack: A, B,

Curr: C



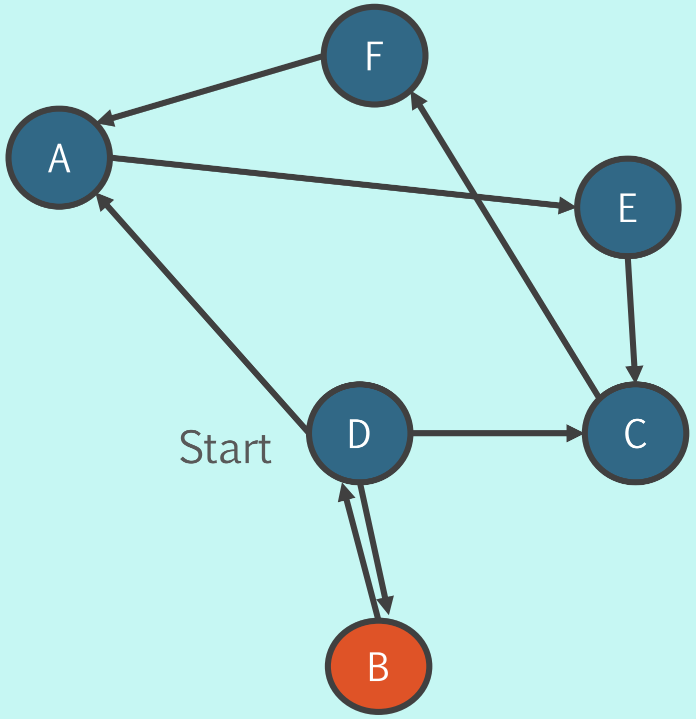
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E

Stack: A, B,

Curr: C



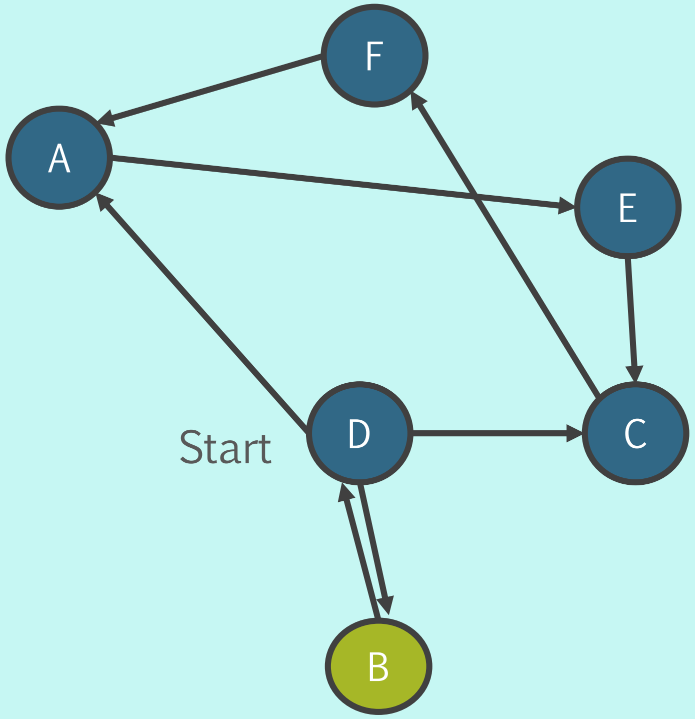
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E

Stack: A

Curr: B



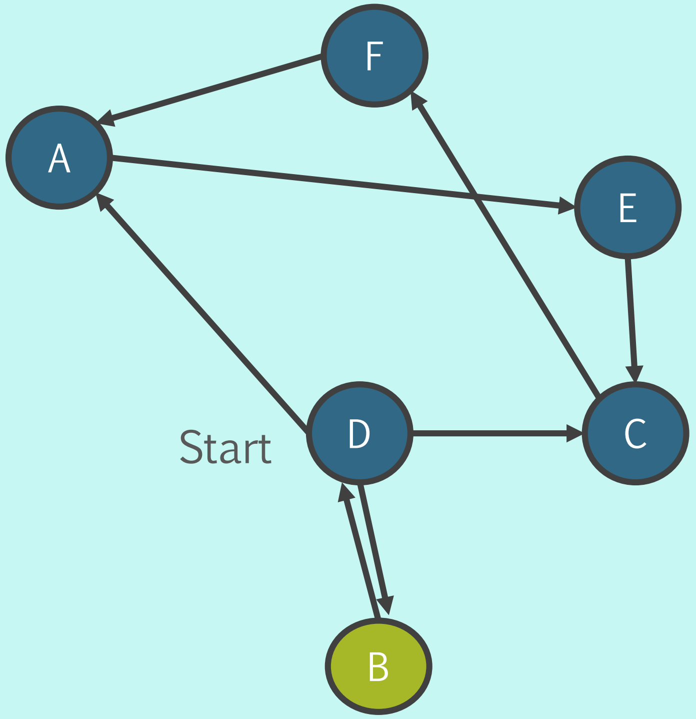
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E

Stack: A

Curr: B



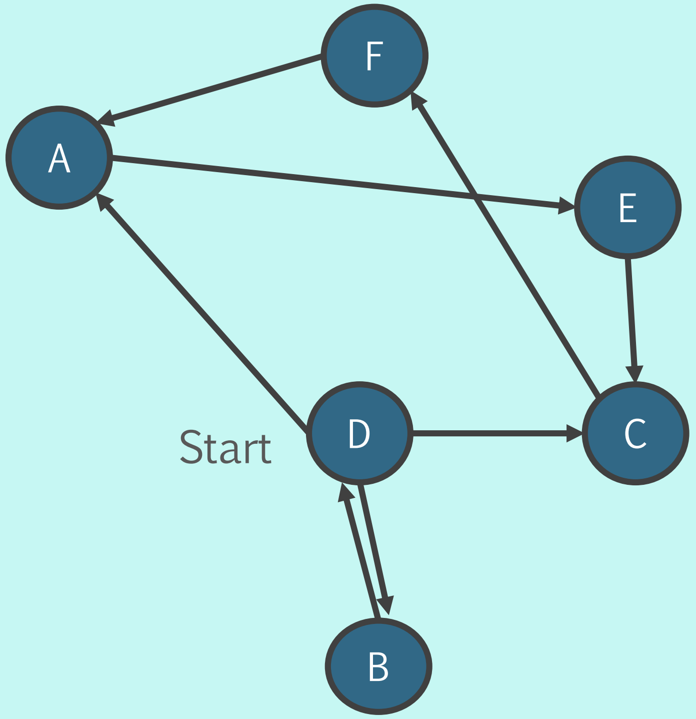
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack: A

Curr: B



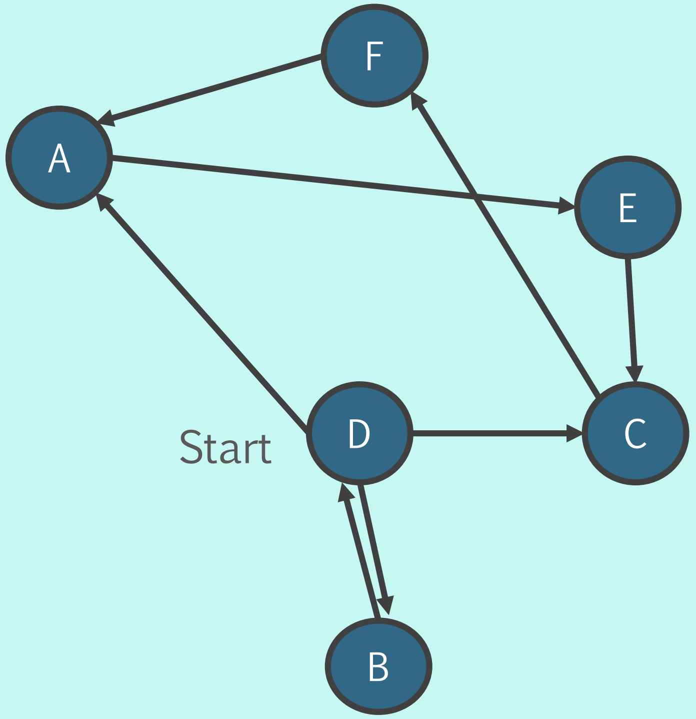
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack: A

Curr: B



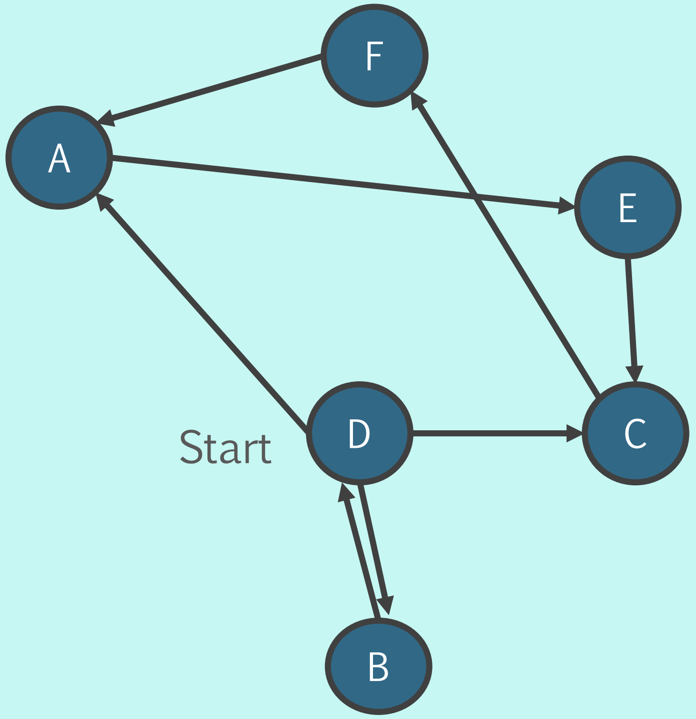
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack: A, D

Curr: B



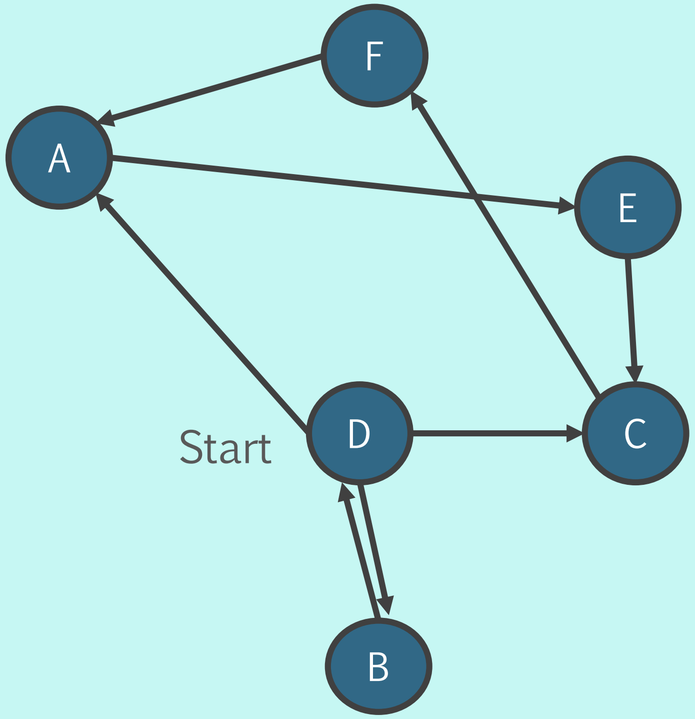
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack: A, D

Curr: B



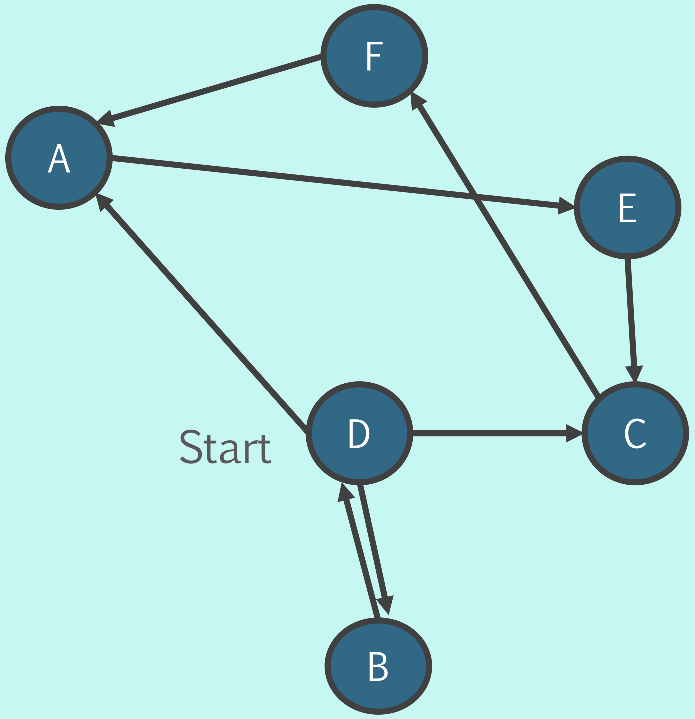
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack: A

Curr: D



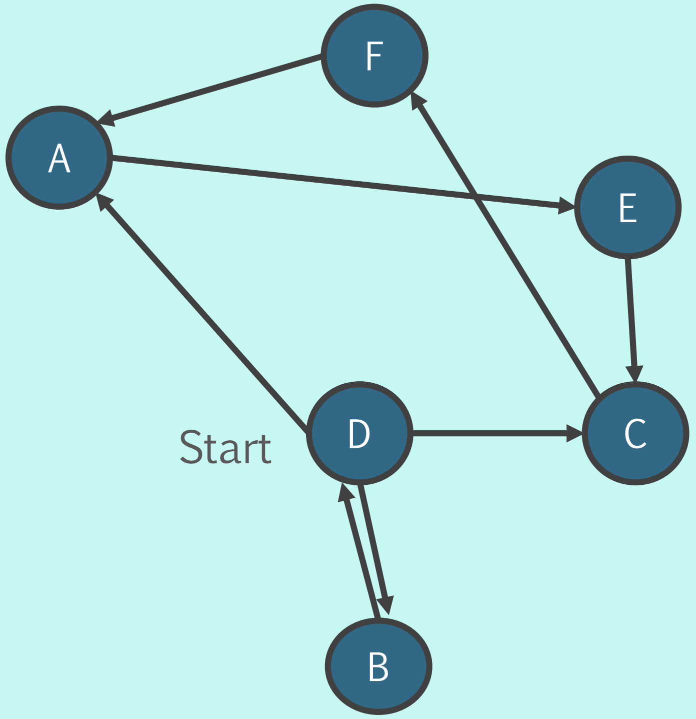
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack: A

Curr: D



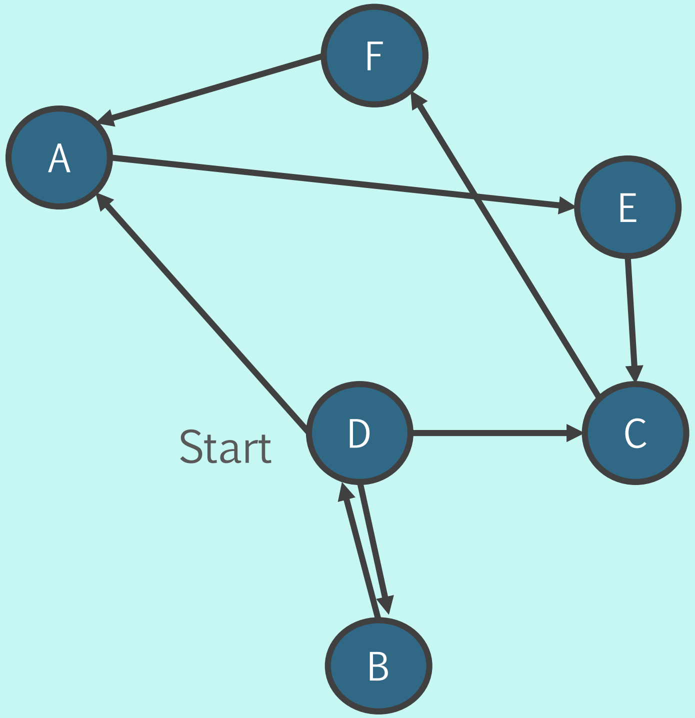
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack: A

Curr: D



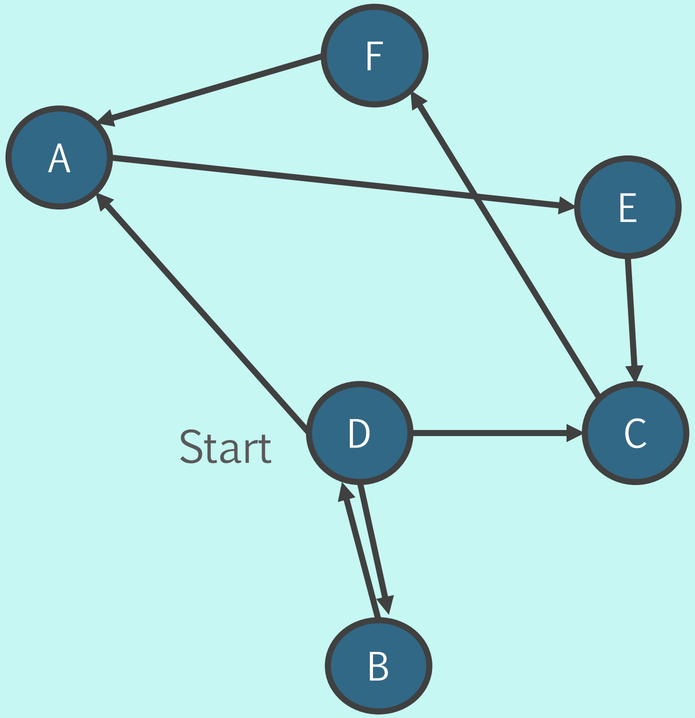
```
GraphSearch(start, goal)
  **Set** visited
  **Stack** s
  s.add(start)
  **while** (s **not** empty)
    curr = s.remove()
    **if** (curr **is** visited)
      **continue**
    visited.add(curr)
    evaluate(curr) // check if goal
    **for** Vertex u **in** neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack: A

Curr: D



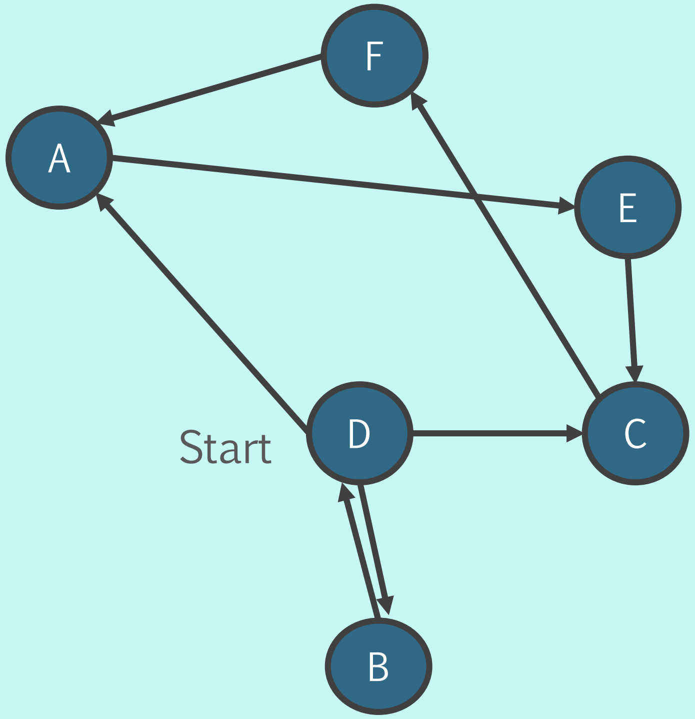
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack

Curr: A



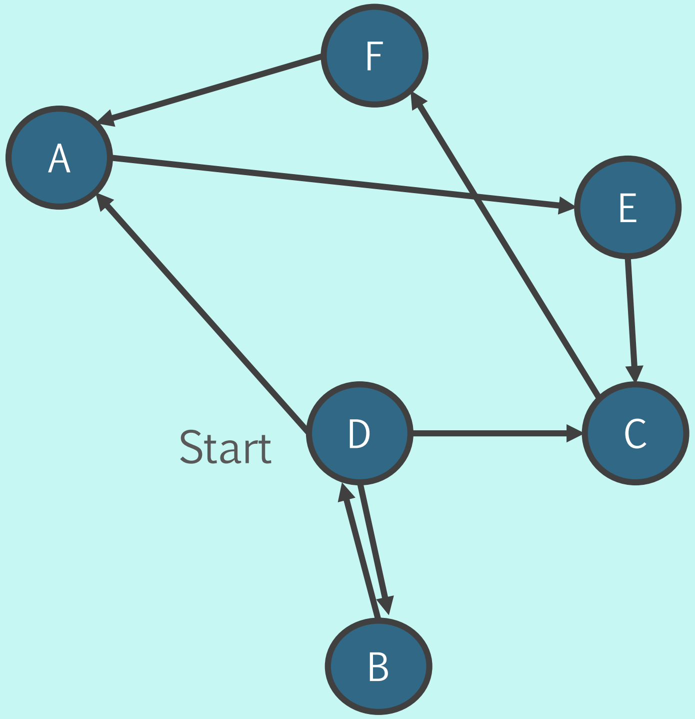
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack

Curr: A



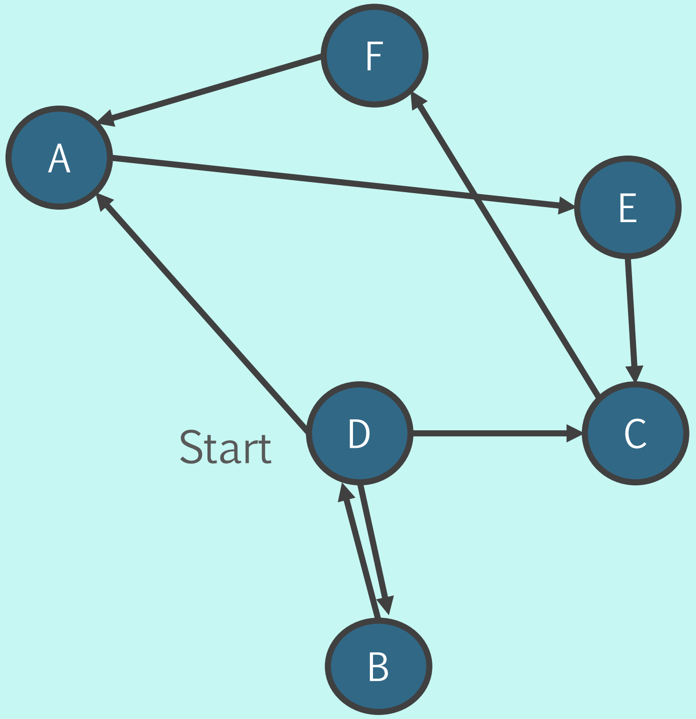
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack

Curr: A



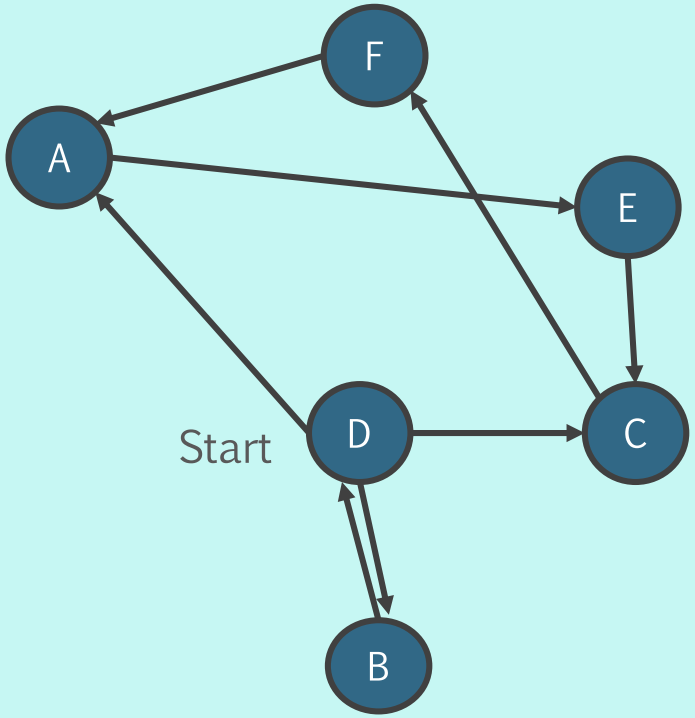
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack

Curr: A



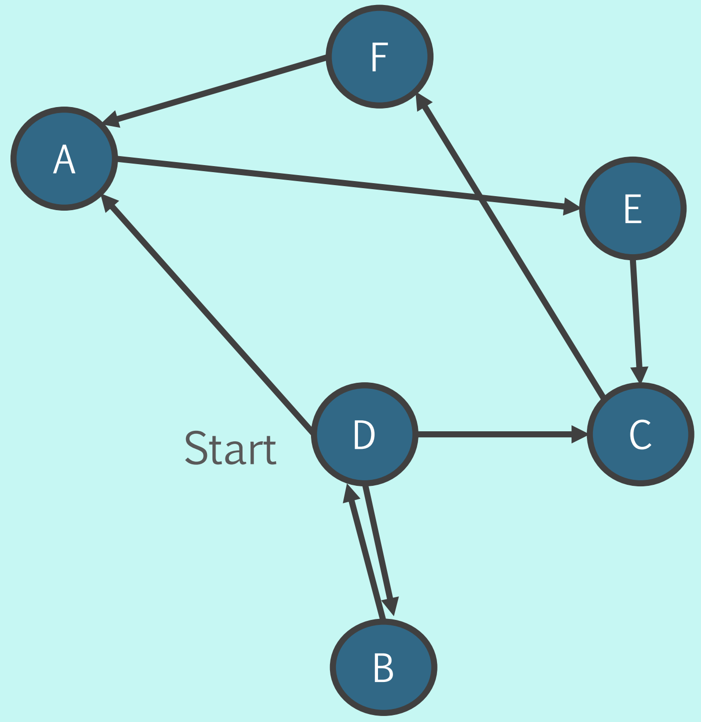
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack

Curr: A



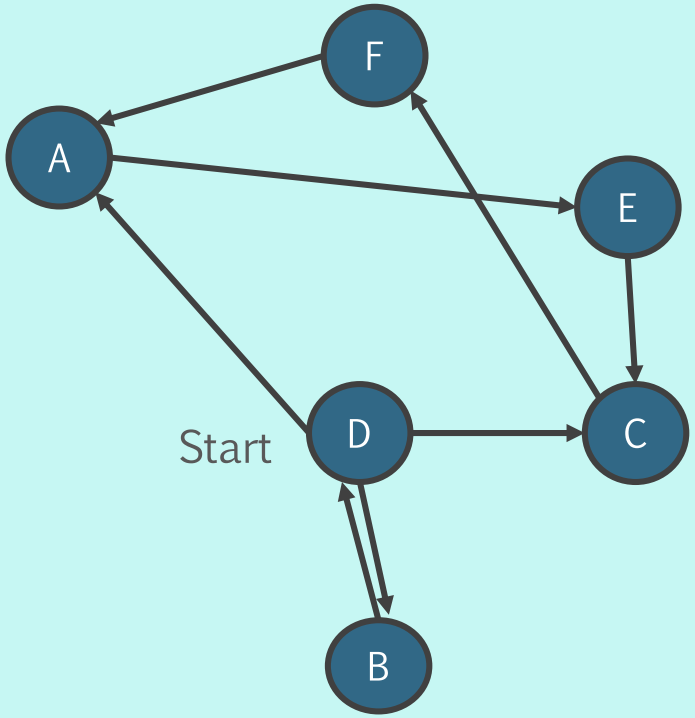
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

Visited: D, C ,F, A, E, B

Stack

Curr: A



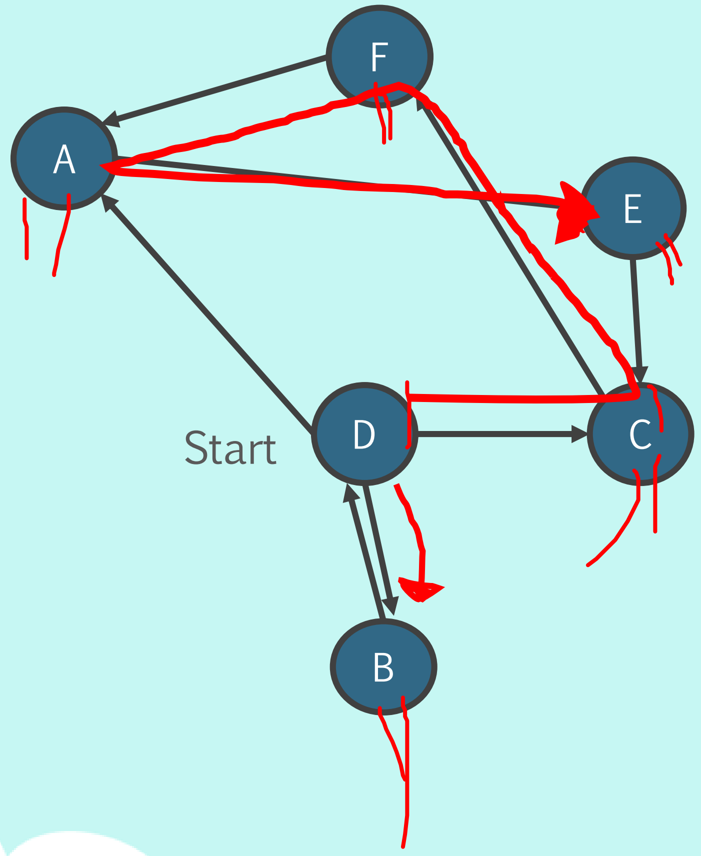
```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```

Depth First Search Example

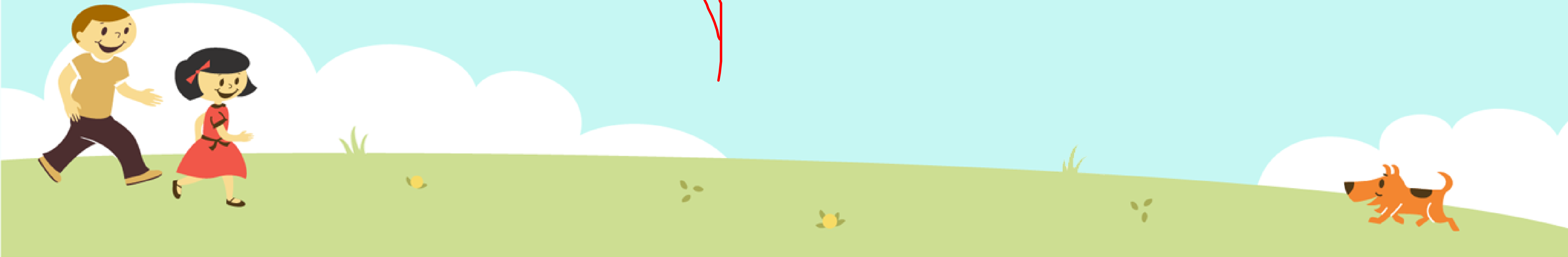
Visited: D, C ,F, A, E, B

Stack

Curr: A

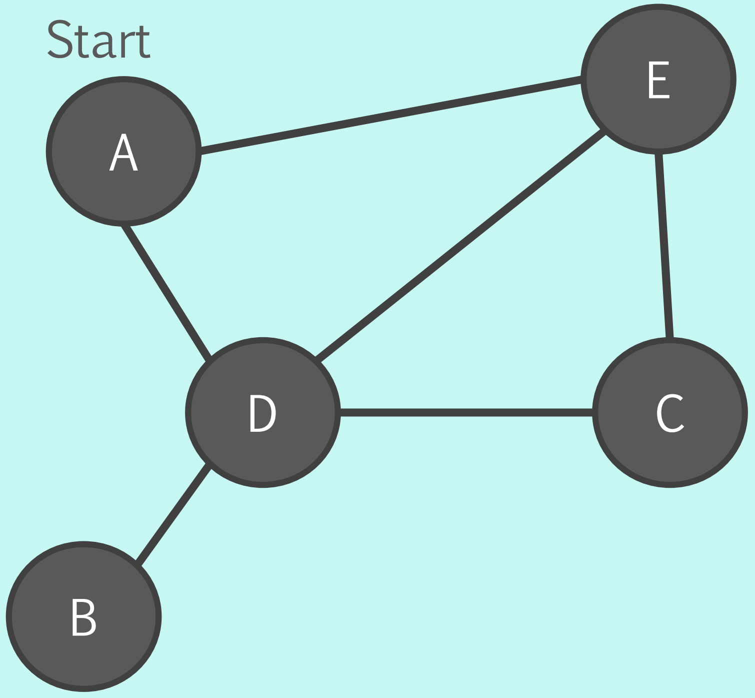


```
GraphSearch(start, goal)
  Set visited
  Stack s
  s.add(start)
  while (s not empty)
    curr = s.remove()
    if (curr is visited)
      continue
    visited.add(curr)
    evaluate(curr) // check if goal
    for Vertex u in neighbors(curr)
      s.add(u)
```



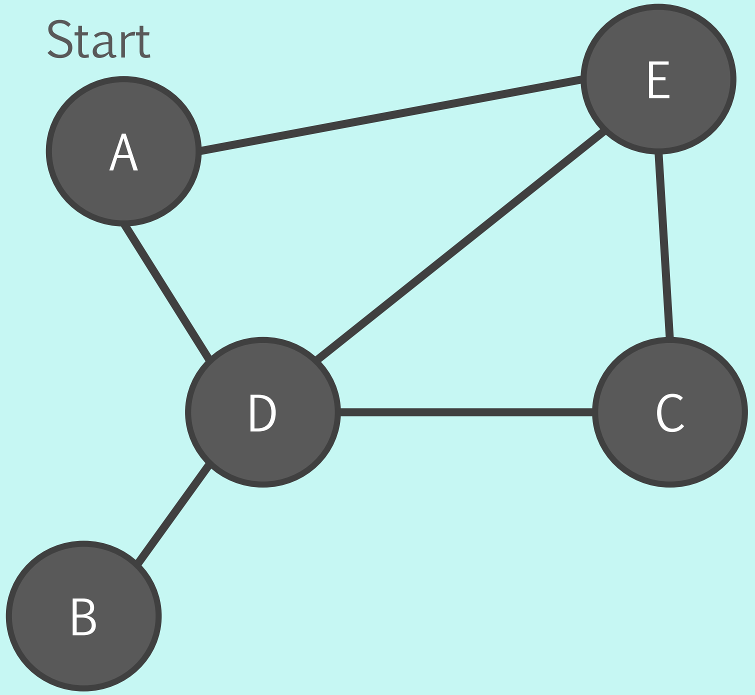
Practice

- Perform BFS and DFS on the Graph. Write the final visiting order.
- If a vertex has multiple neighbors, add them to the Structure in alphabetical order.
- Start with Vertex A
- BFS:
- DFS:



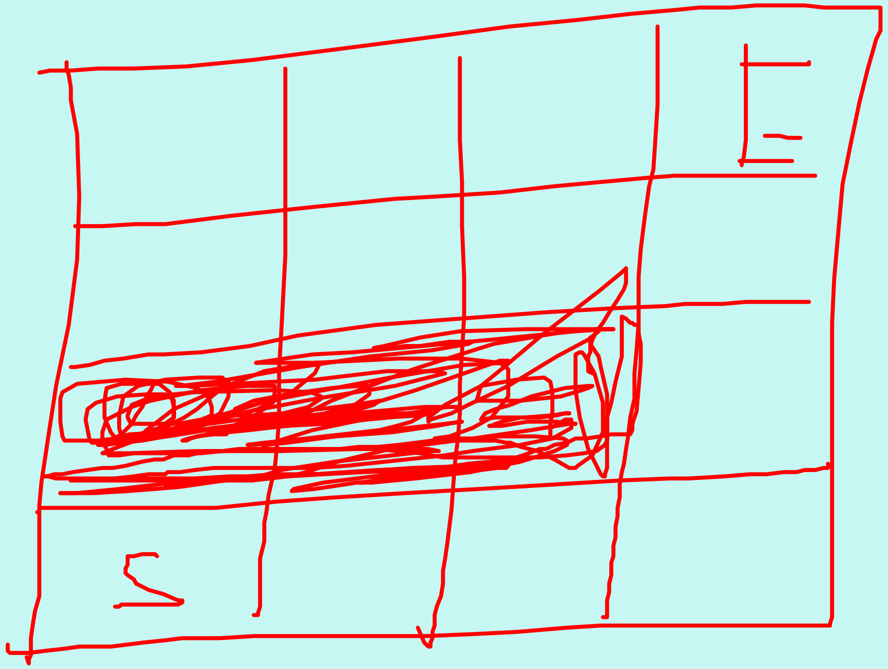
Practice

- Perform BFS and DFS on the Graph. Write the final visiting order.
- If a vertex has multiple neighbors, add them to the Structure in alphabetical order.
- Start with Vertex A
- BFS: A, D, E, B, C
- DFS: A, E, D, C, B



Graph Traversal Usage

- Maze Traversal / Path Finding = Basically AI
  - (States, State Transitions)
- BFS
  - Web Crawler
    - (Webpages, links)
  - GPS Navigation
    - (Locations, roads)
- DFS
  - Cycle Detection
  - Puzzle Solving
    - (Puzzle states, Puzzle moves)



To End

- On your paper write:
  - What’s the most important thing you learned today?
  - What do you have questions with?