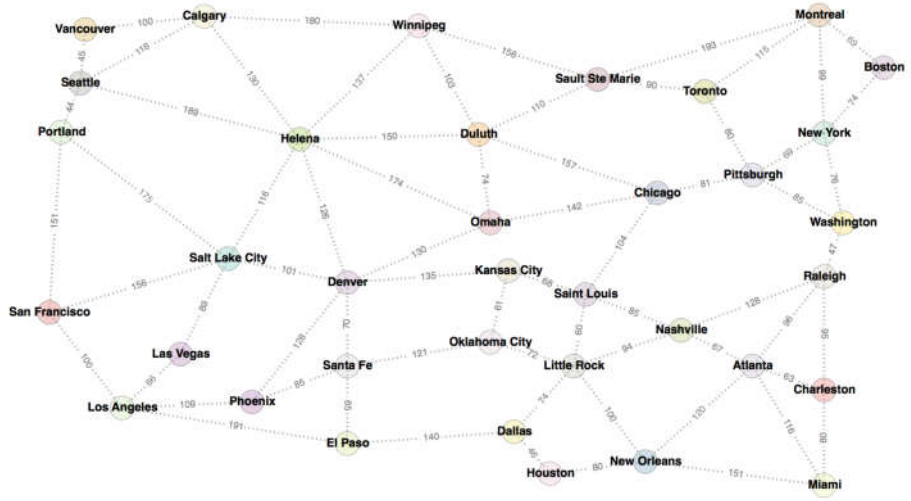
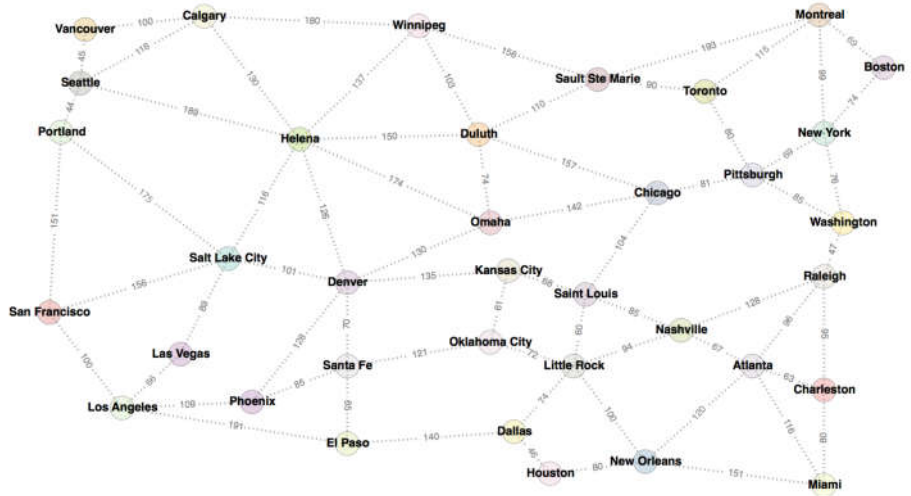
Uniform-cost search



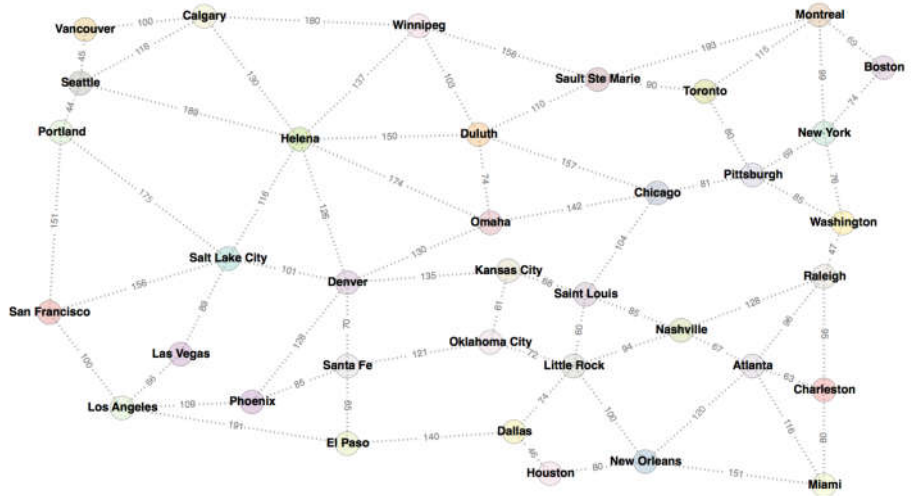
- The arcs in the search graph may have weights (different cost attached). How to leverage this information?

Uniform-cost search



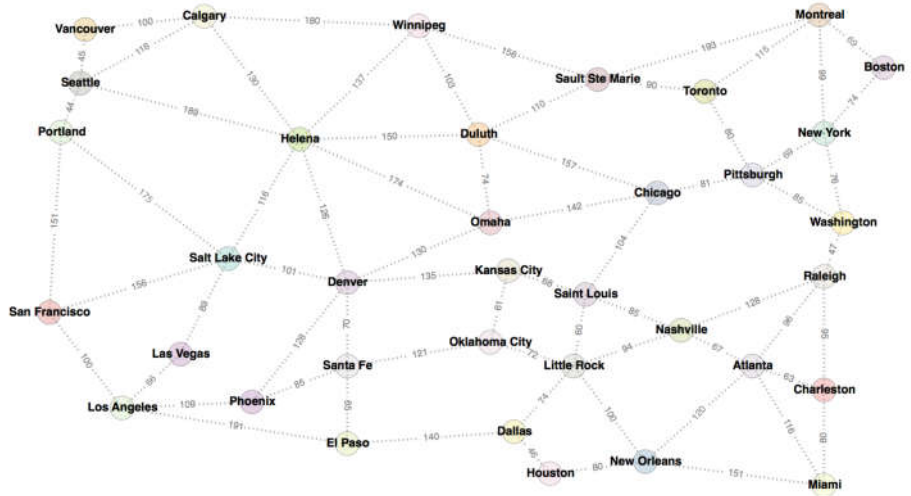
- The arcs in the search graph may have weights (different cost attached). How to leverage this information?
- BFS will find the shortest path which may be costly.
- We want the cheapest not shallowest solution.

Uniform-cost search



- The arcs in the search graph may have weights (different cost attached). How to leverage this information?
- BFS will find the shortest path which may be costly.
- We want the cheapest not shallowest solution.
- Modify BFS: Prioritize by cost not depth $\rightarrow$ **Expand node n with the lowest path cost $g(n)$**

Uniform-cost search

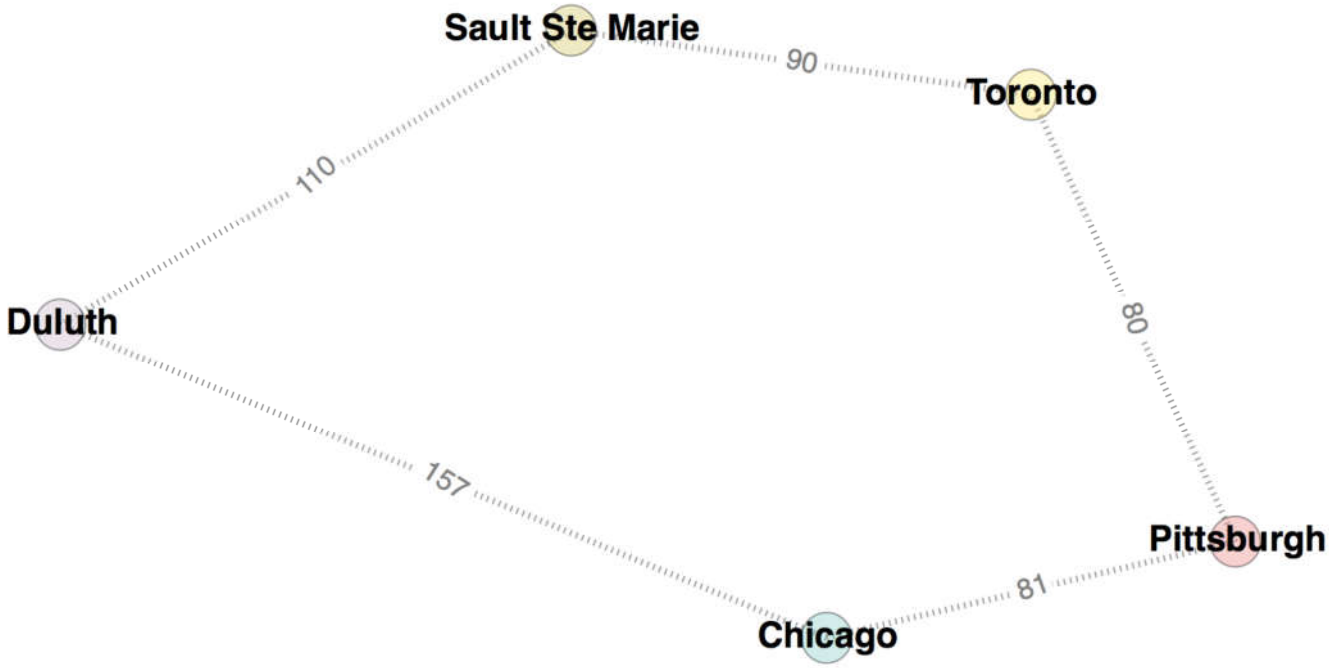


- The arcs in the search graph may have weights (different cost attached). How to leverage this information?
- BFS will find the shortest path which may be costly.
- We want the cheapest not shallowest solution.
- Modify BFS: Prioritize by cost not depth $\rightarrow$ **Expand node n with the lowest path cost $g(n)$**
- Explores increasing costs.

UCS algorithm

```
function UNIFORM-COST-SEARCH(initialState, goalTest)
    returns SUCCESS or FAILURE :  /* Cost f(n) = g(n) */

    frontier = Heap.new(initialState)
    explored = Set.new()

    while not frontier.isEmpty():
        state = frontier.deleteMin()
        explored.add(state)

        if goalTest(state):
            return SUCCESS(state)

        for neighbor in state.neighbors():
            if neighbor not in frontier ∪ explored:
                frontier.insert(neighbor)
            else if neighbor in frontier:
                frontier.decreaseKey(neighbor)

    return FAILURE
```

Uniform-cost search



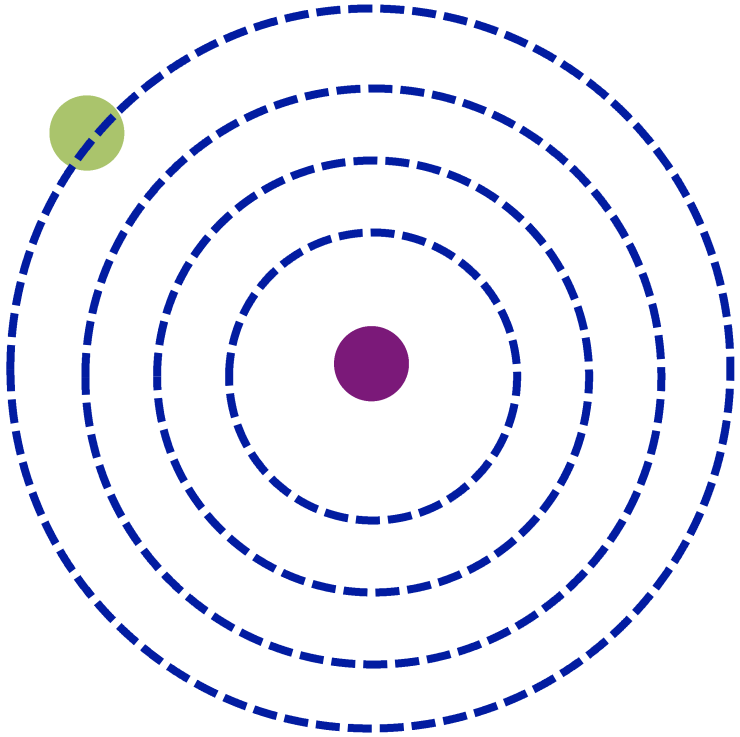
Go from Chicago to Sault Ste Marie. Using BFS, we would find Chicago-Duluth-Sault Ste Marie. However, using UCS, we would find Chicago-Pittsburgh-Toronto-Sault Ste Marie, which is actually the shortest path!

Uniform-cost search

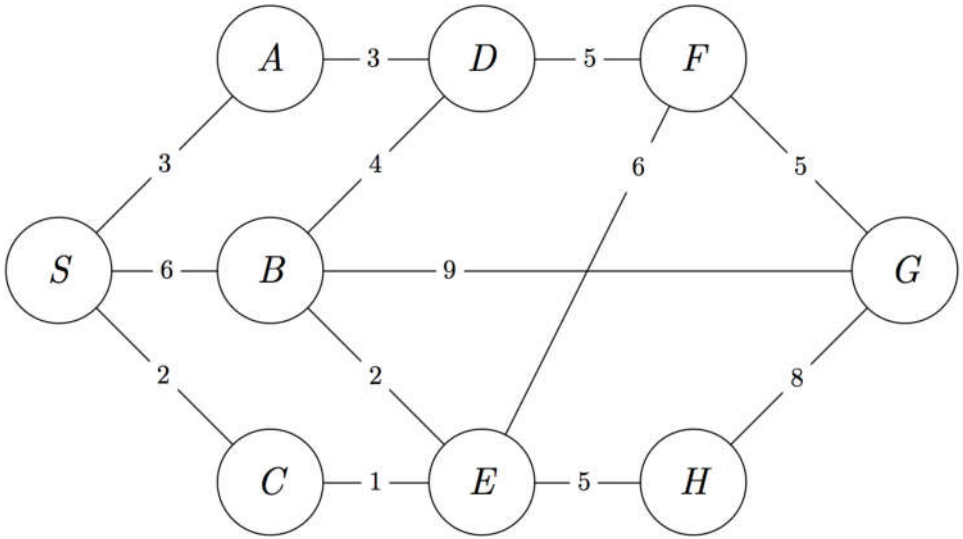
- **Complete** Yes, if solution has a finite cost.
- **Time**
  - Suppose C^*: cost of the optimal solution
  - Every action costs at least ϵ (bound on the cost)
  - The effective depth is roughly C^*/ϵ (how deep the *cheapest* solution could be).
  - $O(b^{C^*/\epsilon})$
- **Space** # of nodes with $g \leq$ cost of optimal solution, $O(b^{C^*/\epsilon})$
- **Optimal** Yes
- **Implementation**: fringe = queue ordered by path cost $g(n)$, lowest first = Heap!

Uniform-cost search

While complete and optimal, UCS explores the space in every direction because no information is provided about the goal!

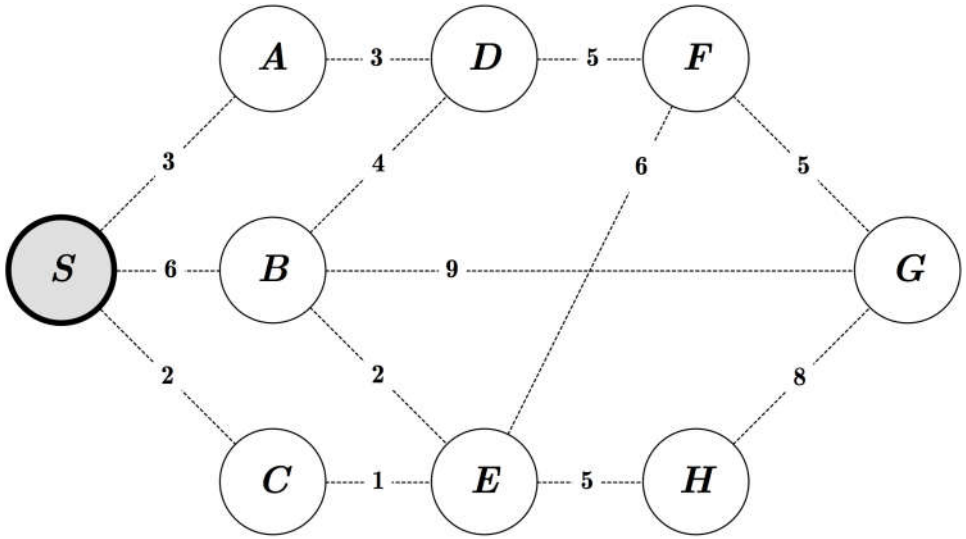


Exercise



Question: What is the **order of visits of the nodes** and the **path** returned by BFS, DFS and UCS?

Exercise: BFS

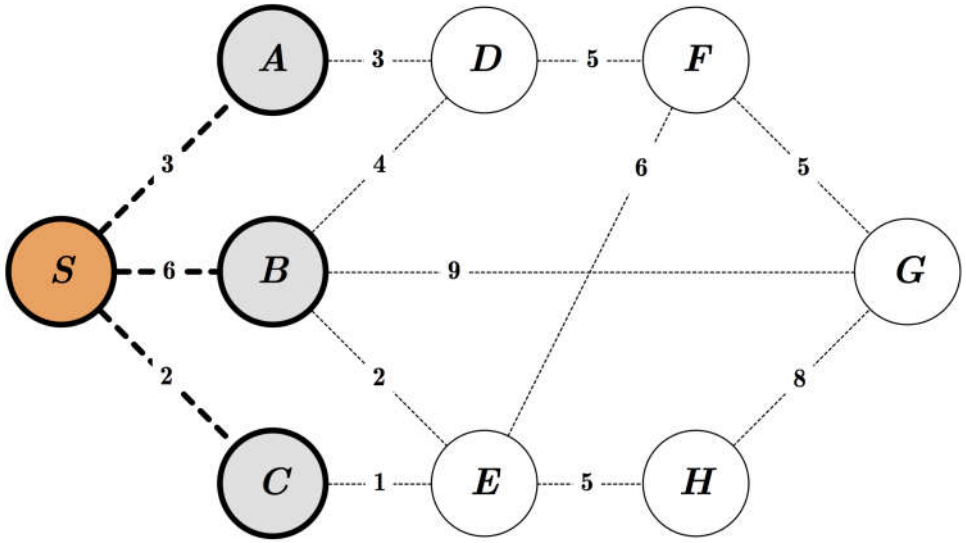


Queue:

S

Order of Visit:

Exercise: BFS



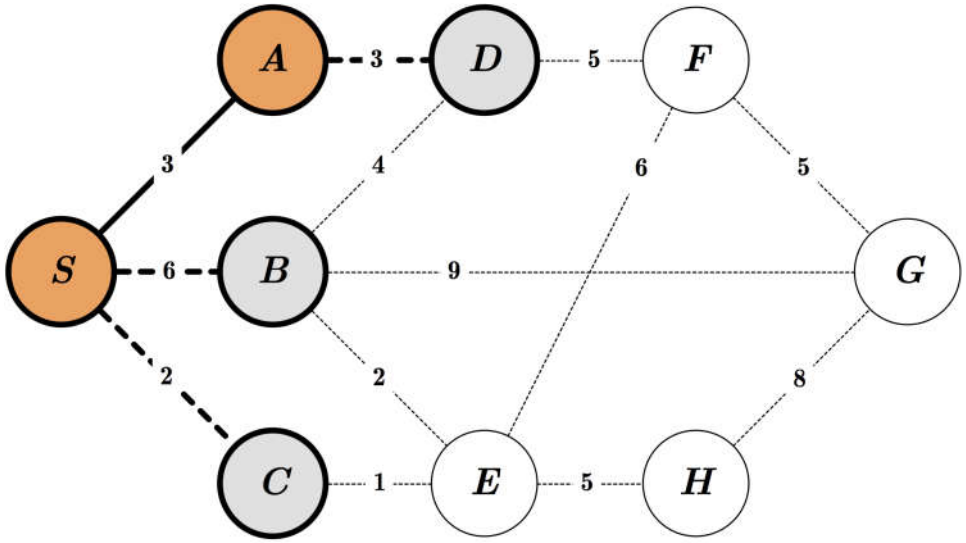
Queue:

S	*A*	*B*	*C*

Order of Visit:

S

Exercise: BFS



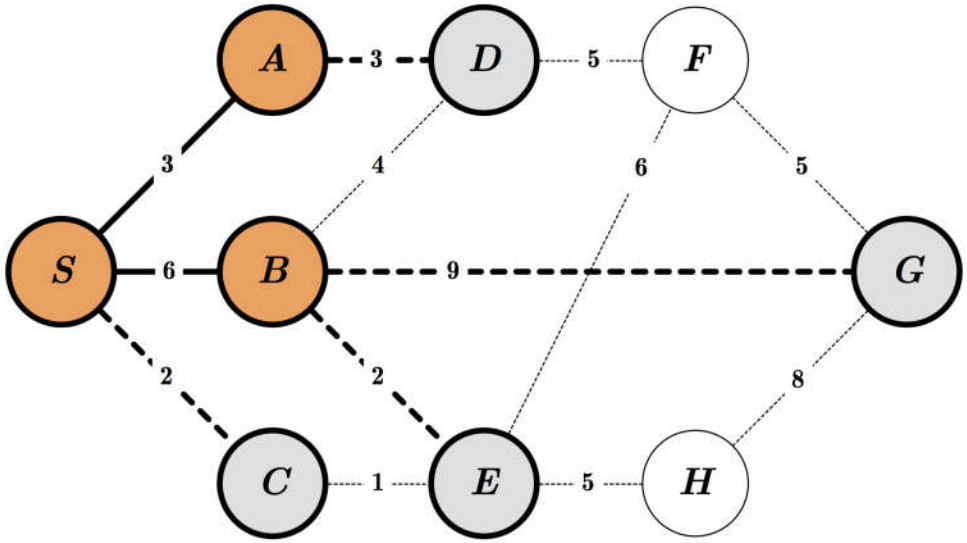
Queue:

S	*A*	*B*	*C*	*D*

Order of Visit:

S *A*

Exercise: BFS



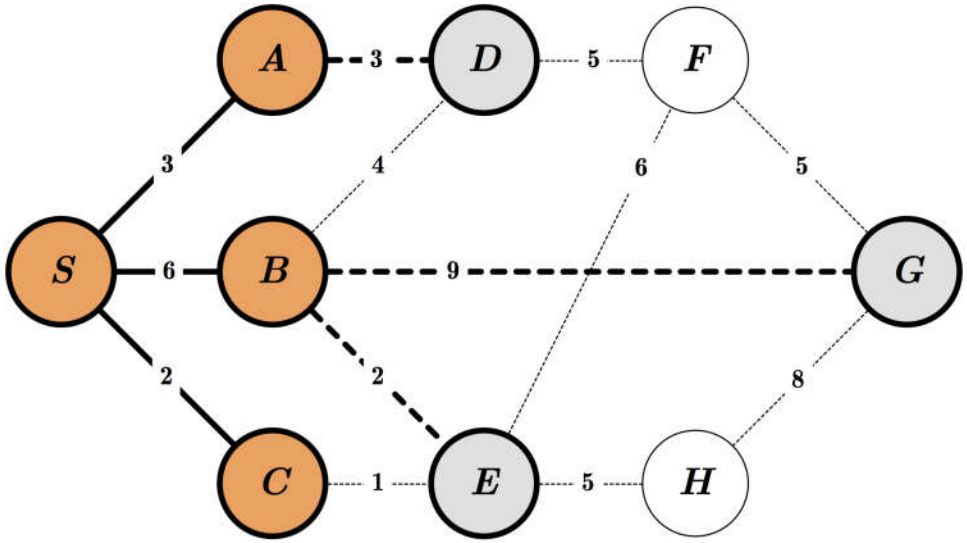
Queue:

S	*A*	*B*	*C*	*D*	*E*	*G*

Order of Visit:

S *A* *B*

Exercise: BFS



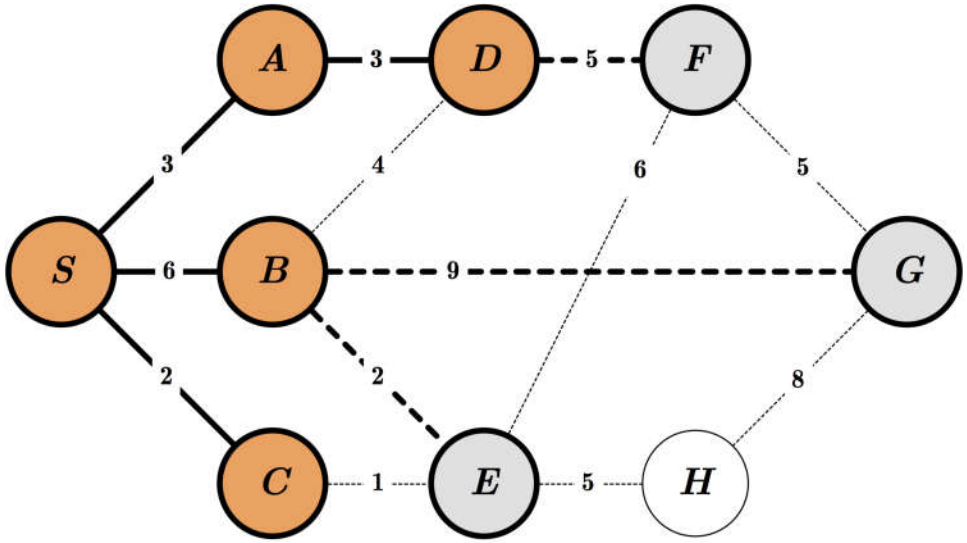
Queue:

S	*A*	*B*	*C*	*D*	*E*	*G*

Order of Visit:

S *A* *B* *C*

Exercise: BFS



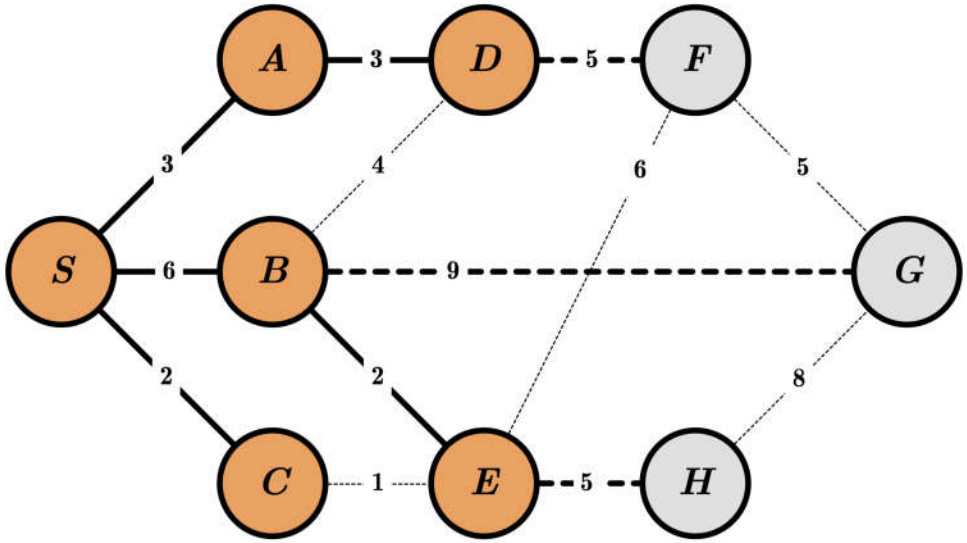
Queue:

S	*A*	*B*	*C*	*D*	*E*	*G*	*F*

Order of Visit:

S *A* *B* *C* *D*

Exercise: BFS



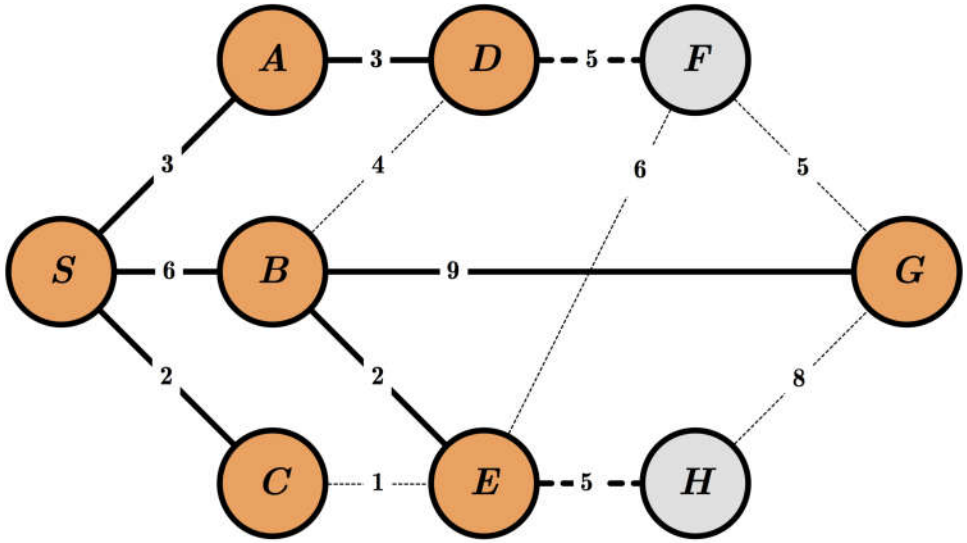
Queue:

S	*A*	*B*	*C*	*D*	*E*	*G*	*F*	*H*

Order of Visit:

S *A* *B* *C* *D* *E*

Exercise: BFS



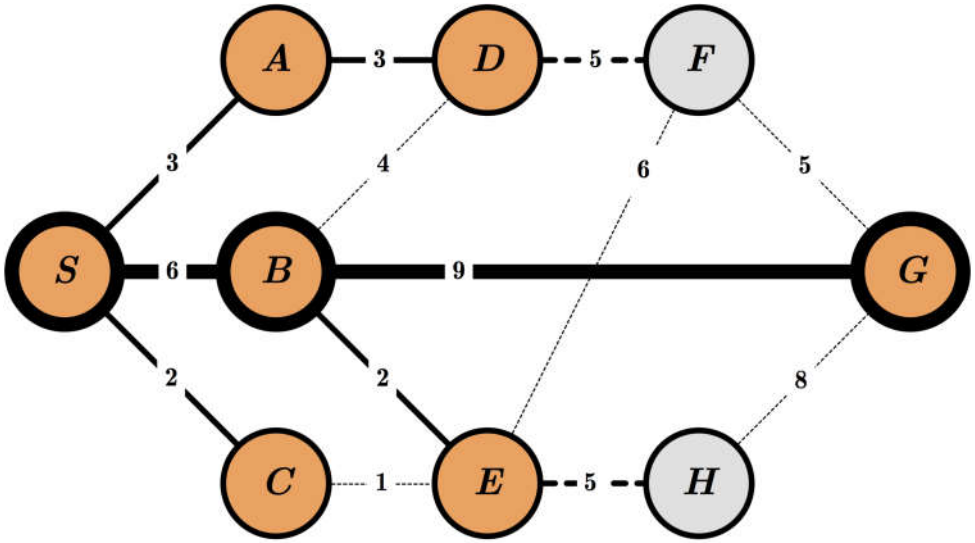
Queue:

S	*A*	*B*	*C*	*D*	*E*	*G*	***F***	***H***

Order of Visit:

S ***A*** ***B*** ***C*** ***D*** ***E*** ***G***

Exercise: BFS



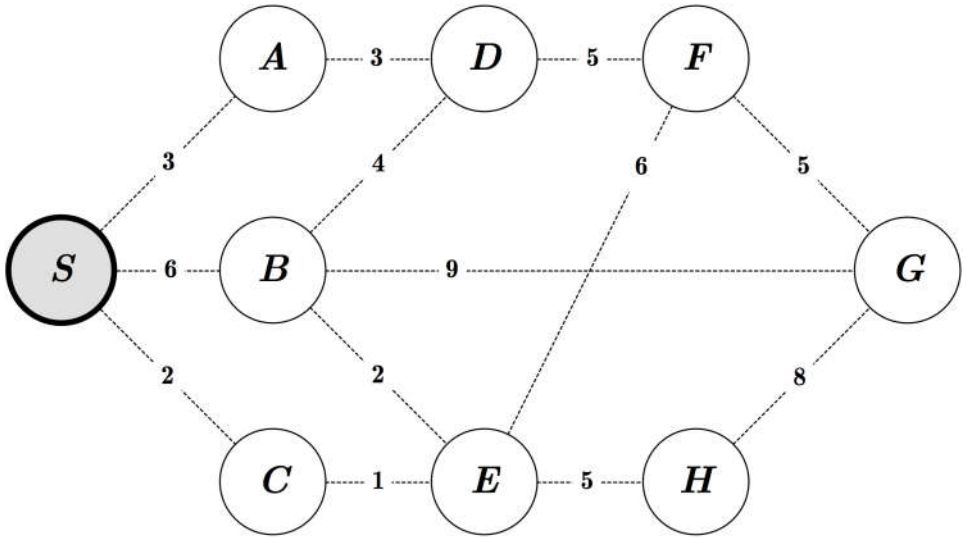
Queue:

S	*A*	*B*	*C*	*D*	*E*	*G*	***F***	***H***

Order of Visit:

S ***A*** ***B*** ***C*** ***D*** ***E*** ***G***

Exercise: DFS

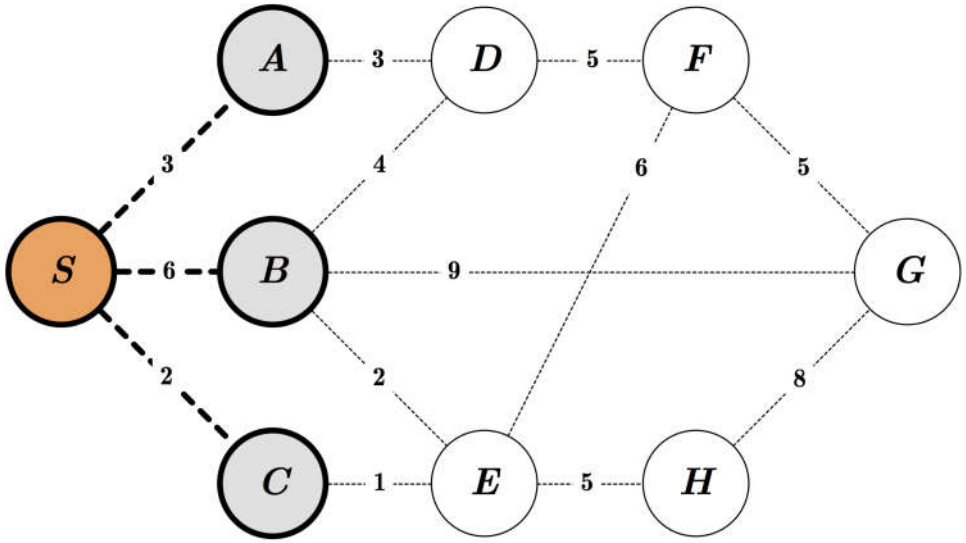


Stack:

S

Order of Visit:

Exercise: DFS



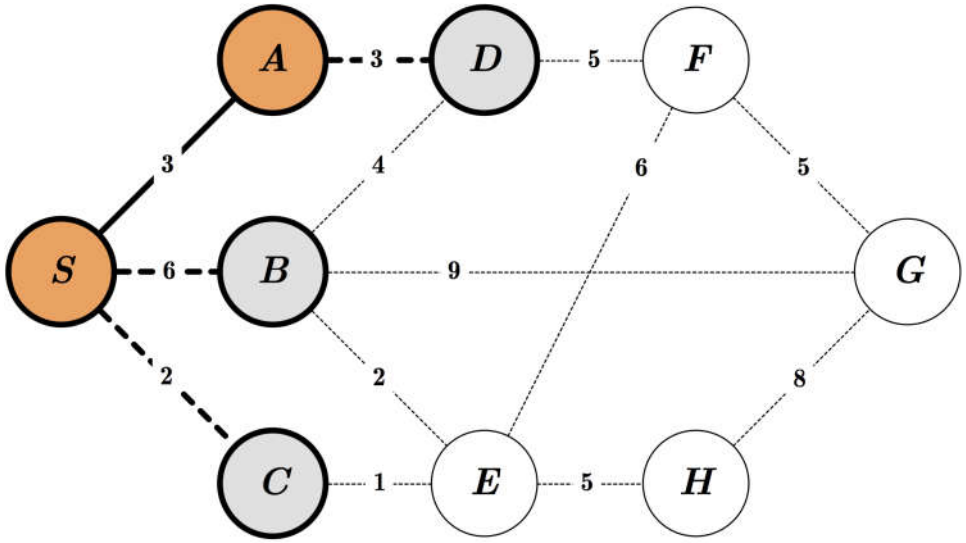
Stack:

S	*C*	*B*	*A*

Order of Visit:

S

Exercise: DFS



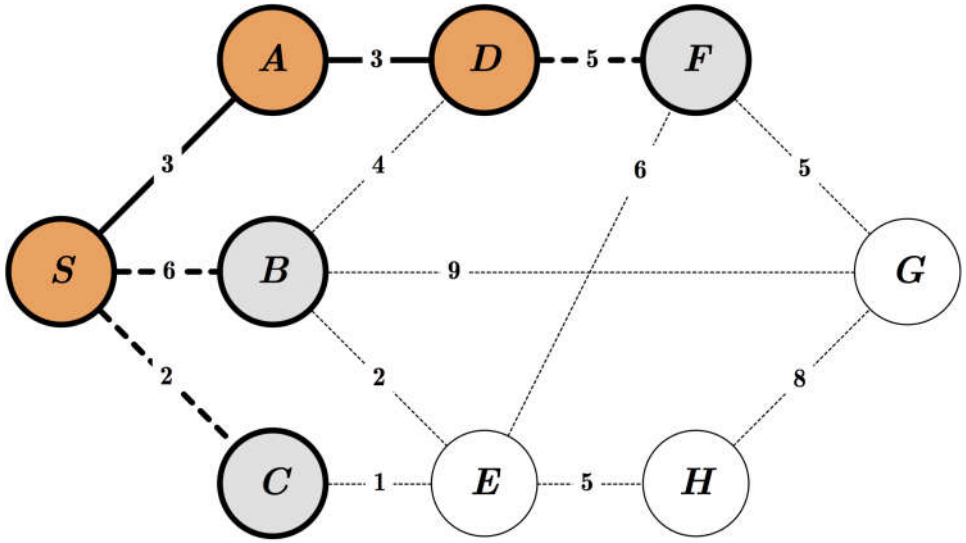
Stack:

S	*C*	*B*	*A*	*D*

Order of Visit:

S *A*

Exercise: DFS



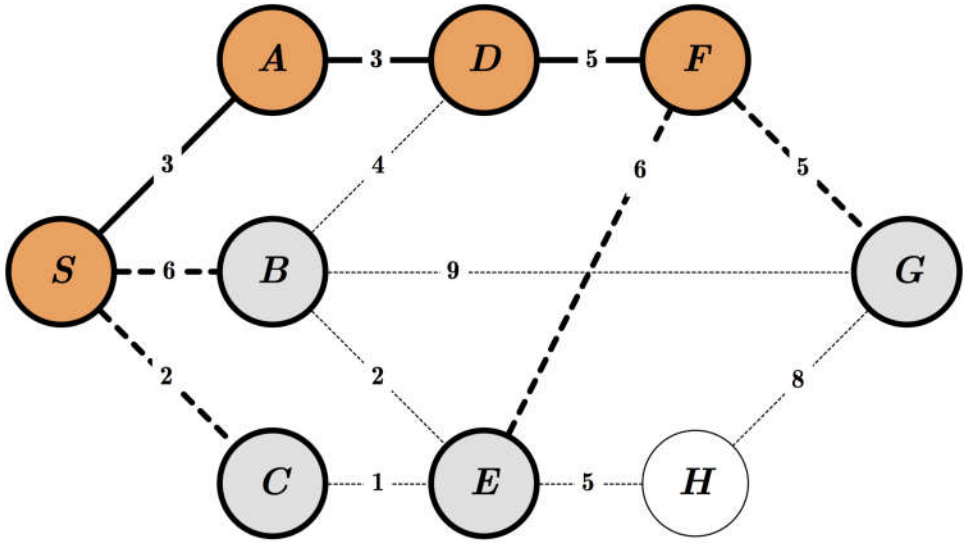
Stack:

S	*C*	*B*	*A*	*D*	*F*

Order of Visit:

S *A* *D*

Exercise: DFS



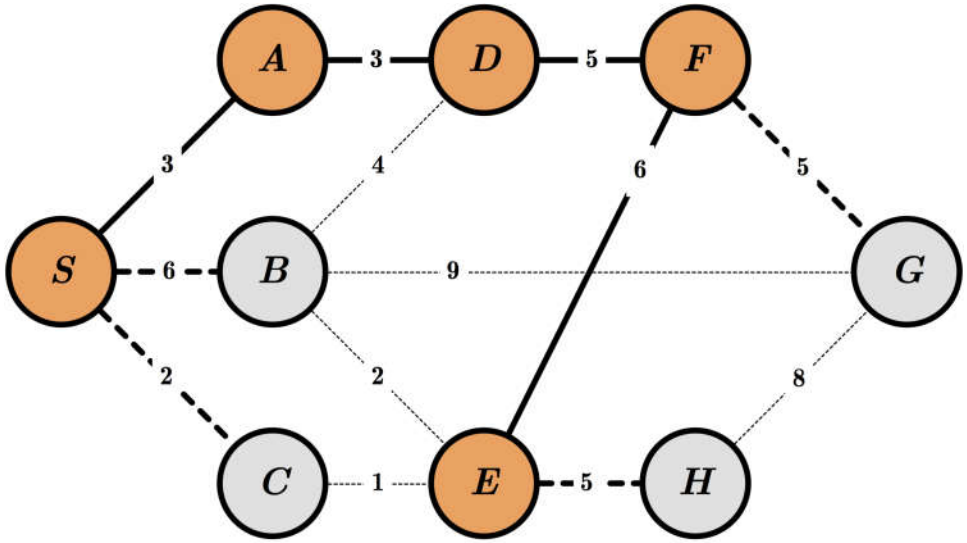
Stack:

S	*C*	*B*	*A*	*D*	*F*	*G*	*E*

Order of Visit:

S *A* *D* *F*

Exercise: DFS



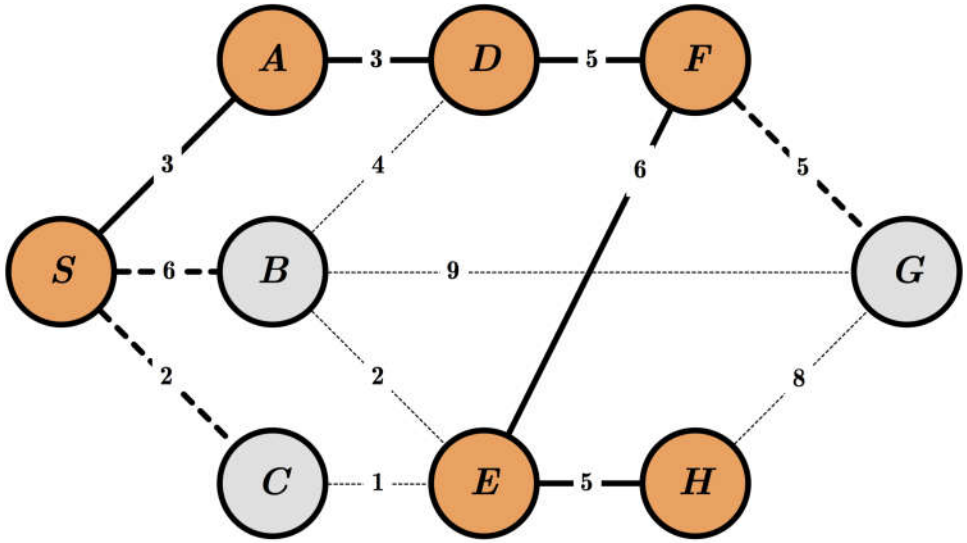
Stack:

S	*C*	*B*	*A*	*D*	*F*	*G*	*E*	*H*

Order of Visit:

S *A* *D* *F* *E*

Exercise: DFS



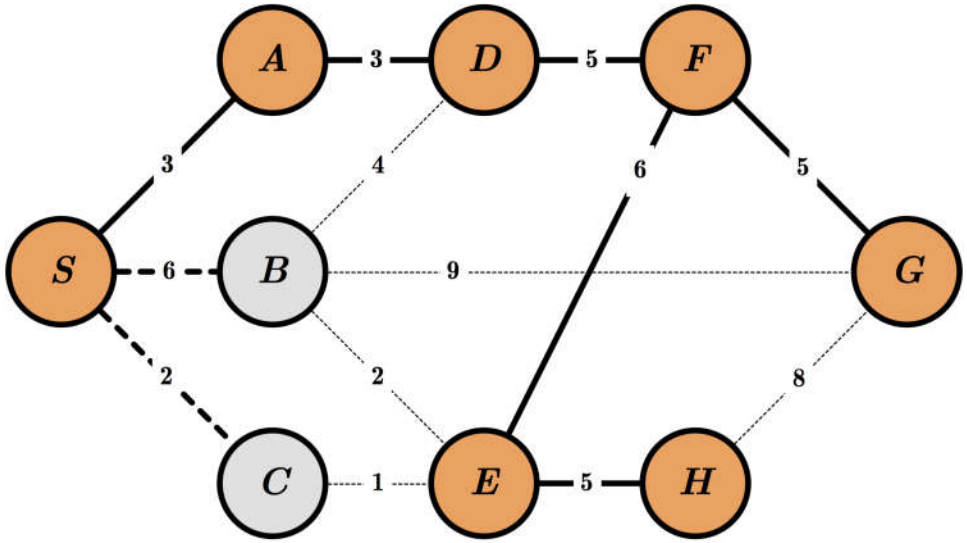
Stack:

S	*C*	*B*	*A*	*D*	*F*	*G*	*E*	*H*

Order of Visit:

S *A* *D* *F* *E* *H*

Exercise: DFS



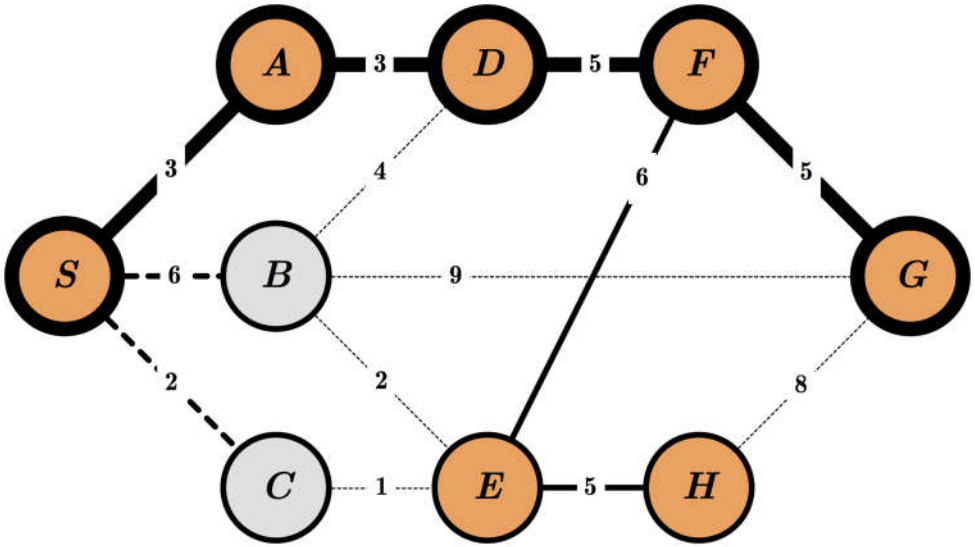
Stack:

S	*C*	*B*	*A*	*D*	*F*	*G*	*E*	*H*

Order of Visit:

S *A* *D* *F* *E* *H* *G*

Exercise: DFS



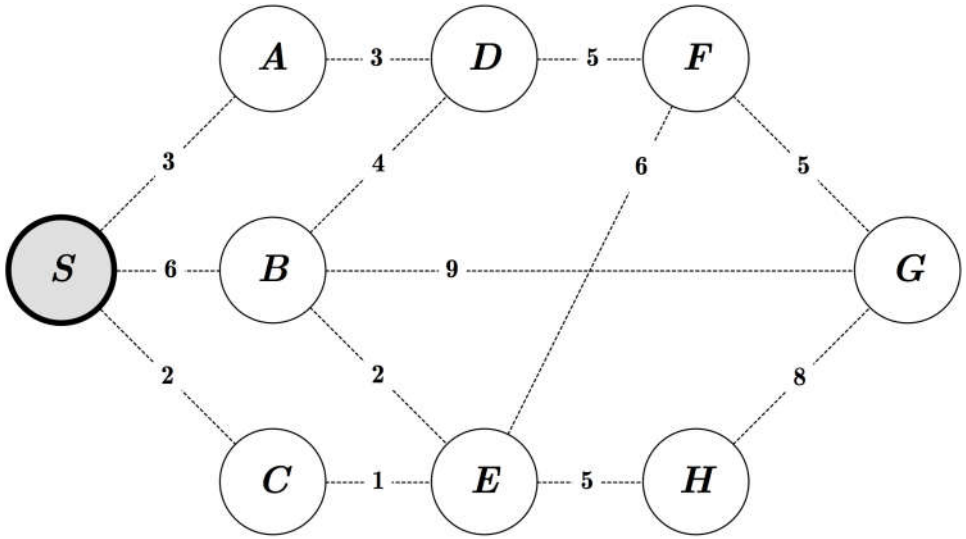
Stack:

S	*C*	*B*	*A*	*D*	*F*	*G*	*E*	*H*

Order of Visit:

S *A* *D* *F* *E* *H* *G*

Exercise: UCS

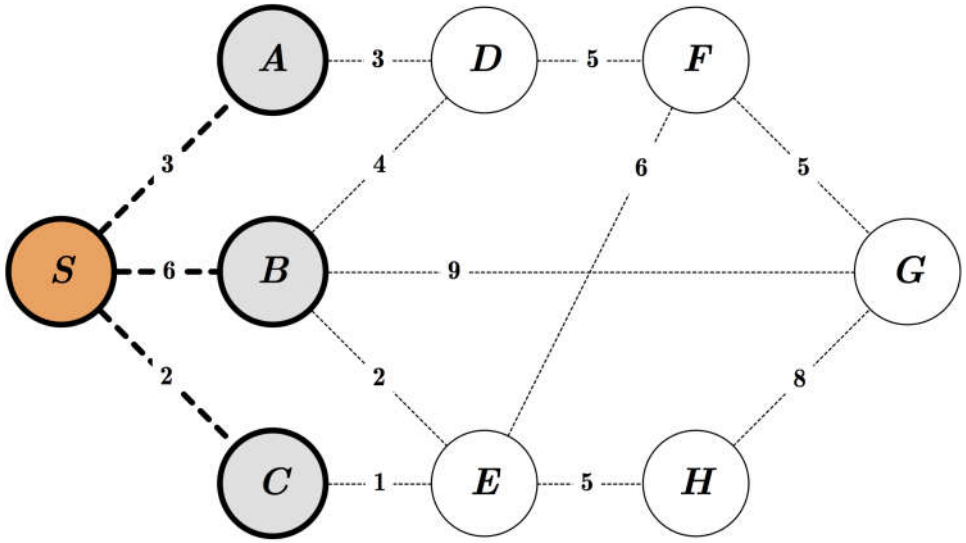


Priority Queue:

S_0

Order of Visit:

Exercise: UCS



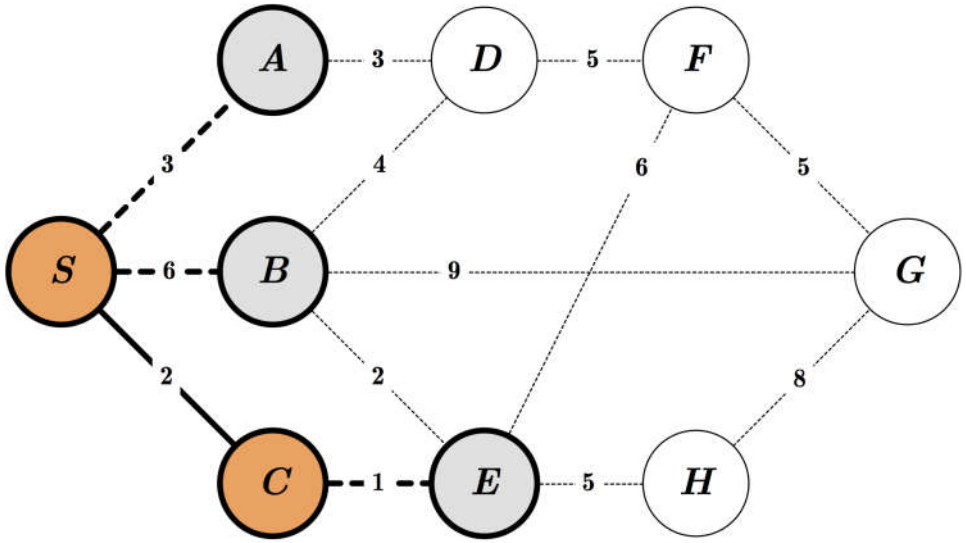
Priority Queue:

S_0	C_2	A_3	B_6

Order of Visit:

S

Exercise: UCS



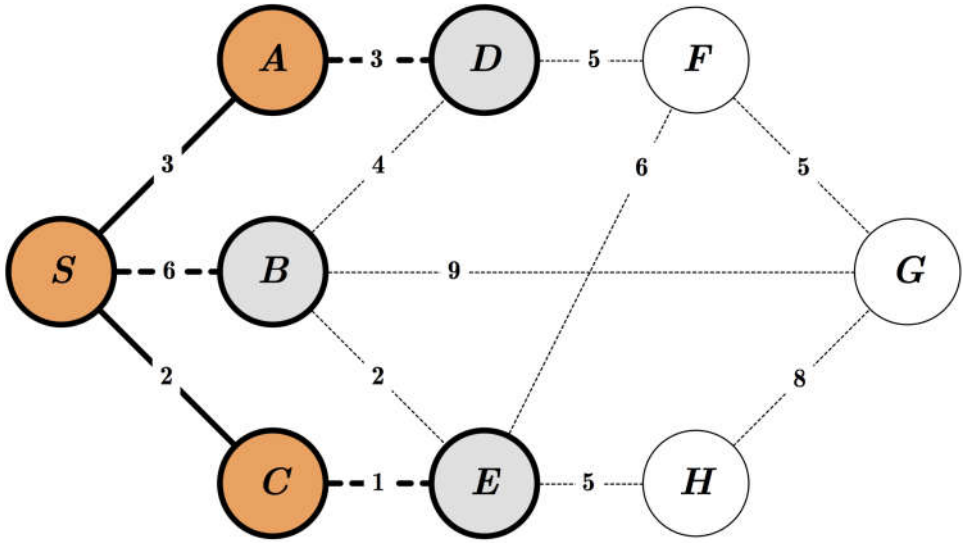
Priority Queue:

S_0	C_2	A_3	E_3	B_6

Order of Visit:

S C

Exercise: UCS



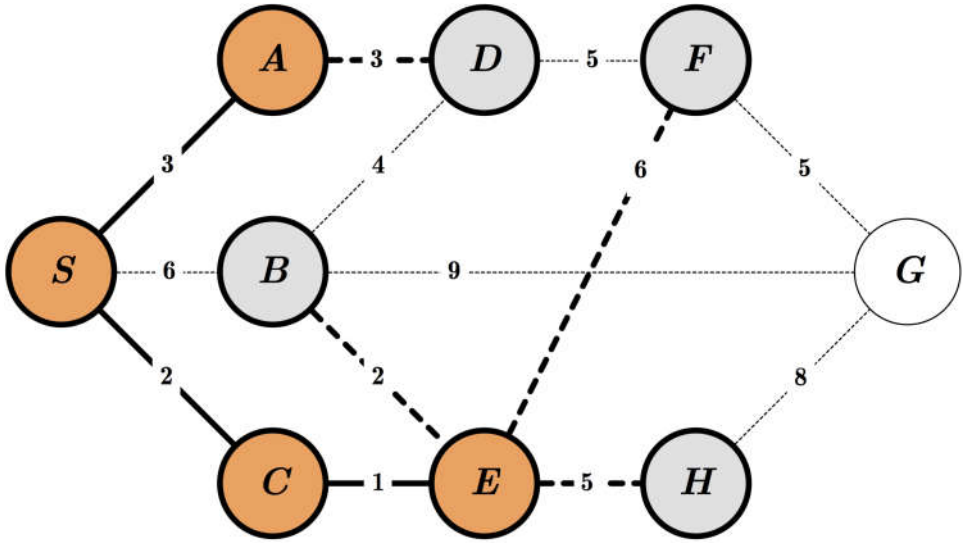
Priority Queue:

S_0	C_2	A_3	E_3	B_6	D_6

Order of Visit:

S C A

Exercise: UCS



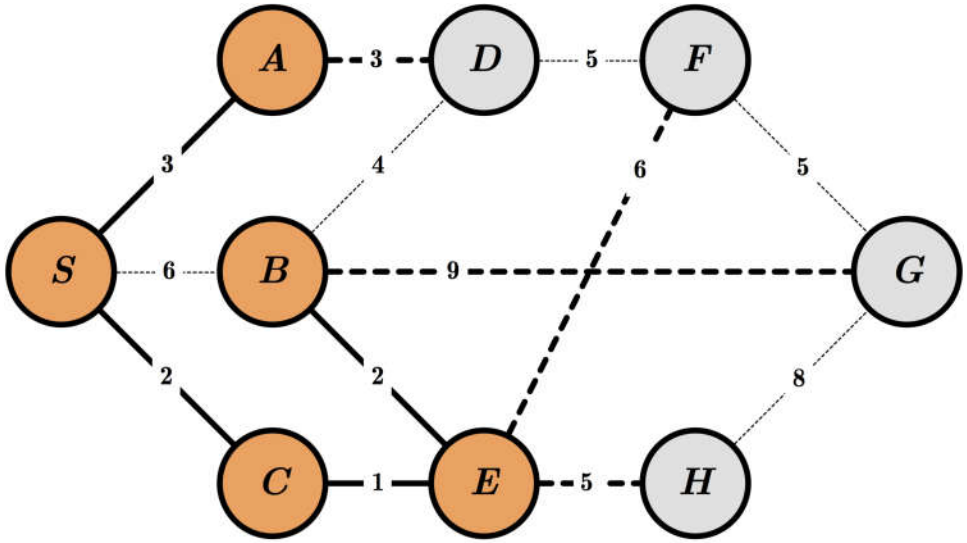
Priority Queue:

S_0	C_2	A_3	E_3	B_5	D_6	H_8	F_9

Order of Visit:

S C A E

Exercise: UCS



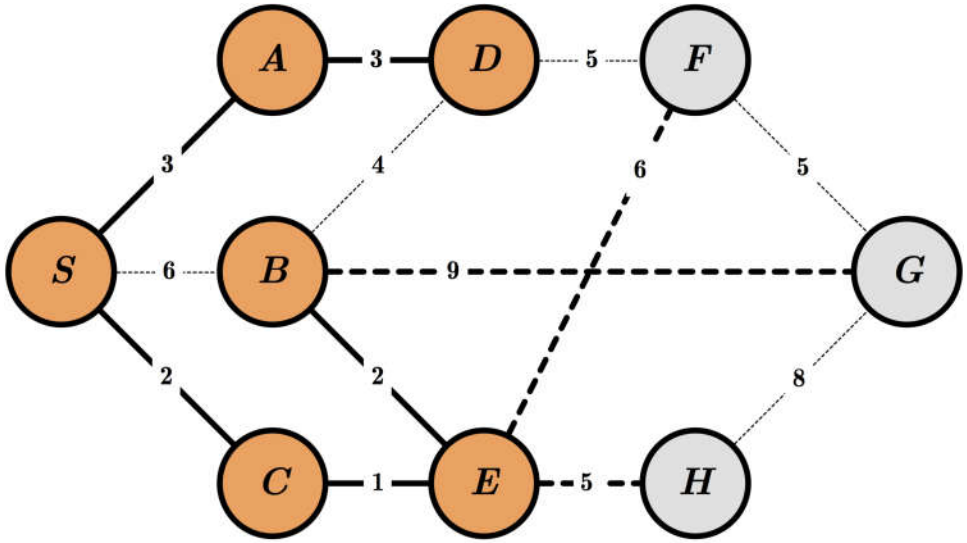
Priority Queue:

S_0	C_2	A_3	E_3	B_5	D_6	H_8	F_9	G_{14}

Order of Visit:

S C A E B

Exercise: UCS



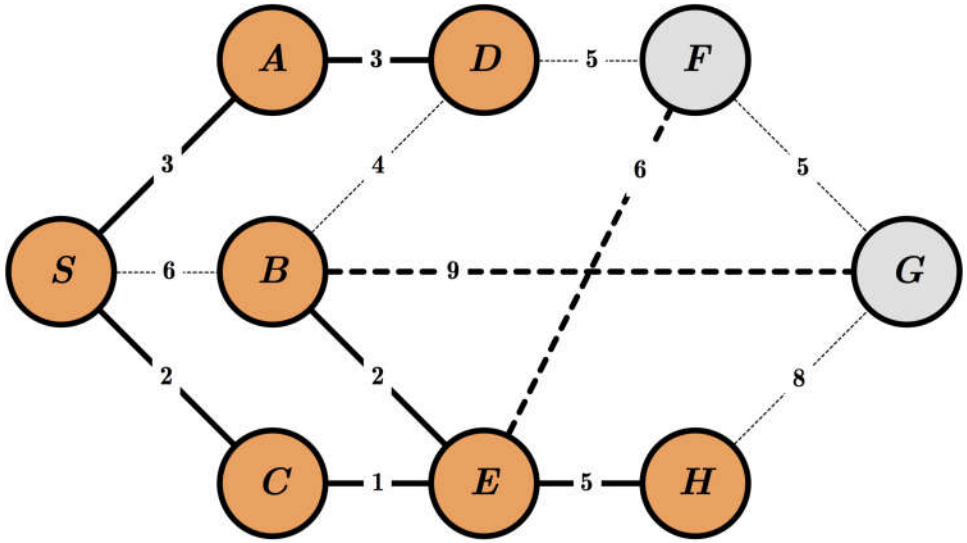
Priority Queue:

S_0	C_2	A_3	E_3	B_5	D_6	H_8	F_9	G_{14}

Order of Visit:

S C A E B D

Exercise: UCS



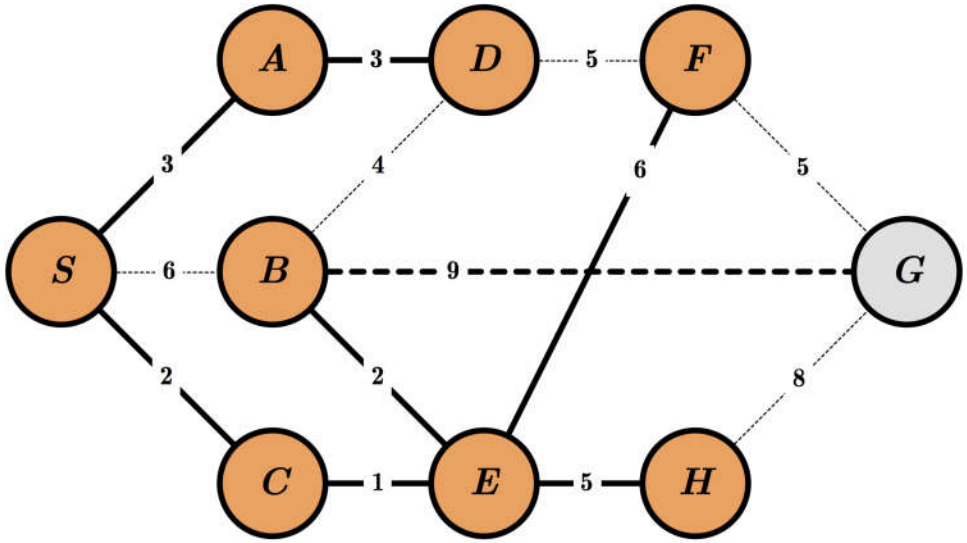
Priority Queue:

S_0	C_2	A_3	E_3	B_5	D_6	H_8	F_9	G_{14}

Order of Visit:

S C A E B D H

Exercise: UCS



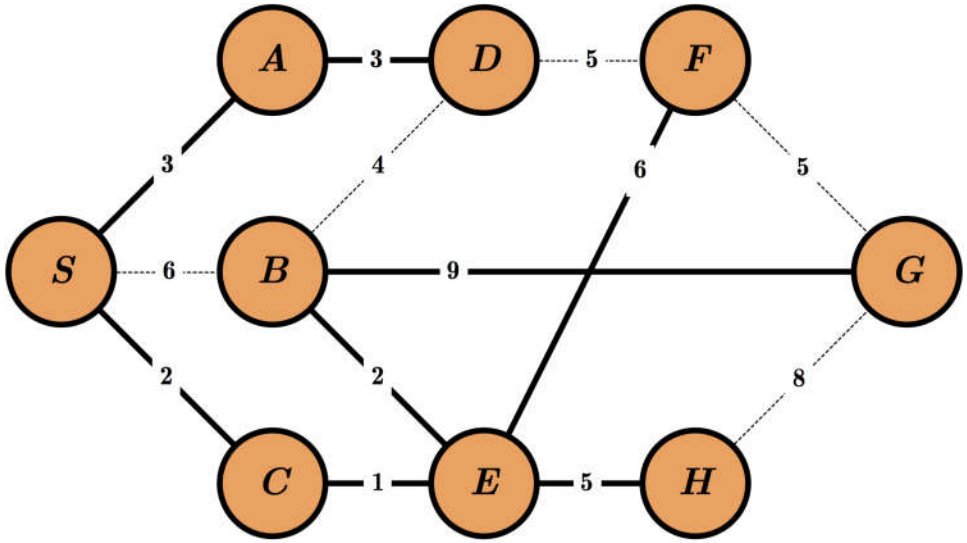
Priority Queue:

S_0	C_2	A_3	E_3	B_5	D_6	H_8	F_9	G_{14}

Order of Visit:

S C A E B D H F

Exercise: UCS



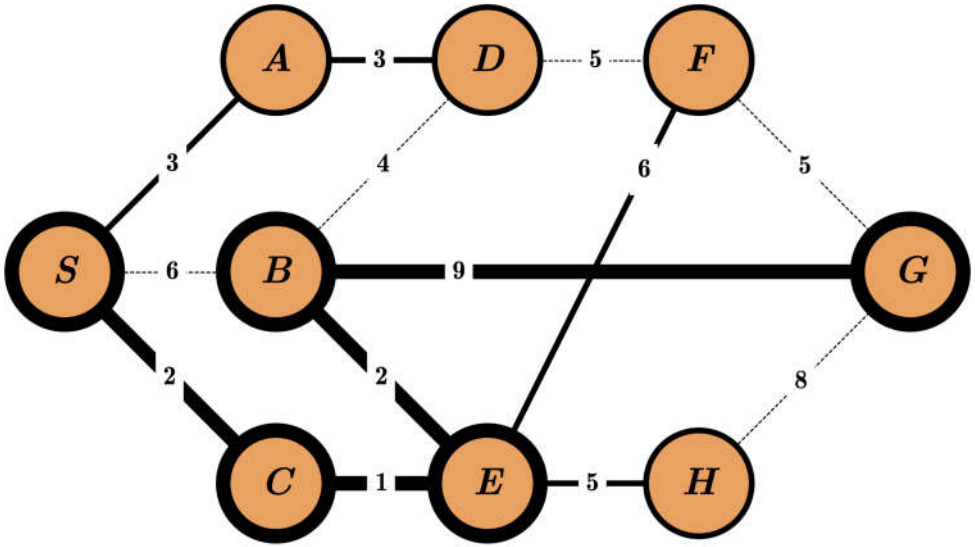
Priority Queue:

S_0	C_2	A_3	E_3	B_5	D_6	H_8	F_9	G_{14}

Order of Visit:

S	*C*	*A*	*E*	*B*	*D*	*H*	*F*	*G*

Exercise: UCS



Priority Queue:

S_0	C_2	A_3	E_3	B_5	D_6	H_8	F_9	G_{14}

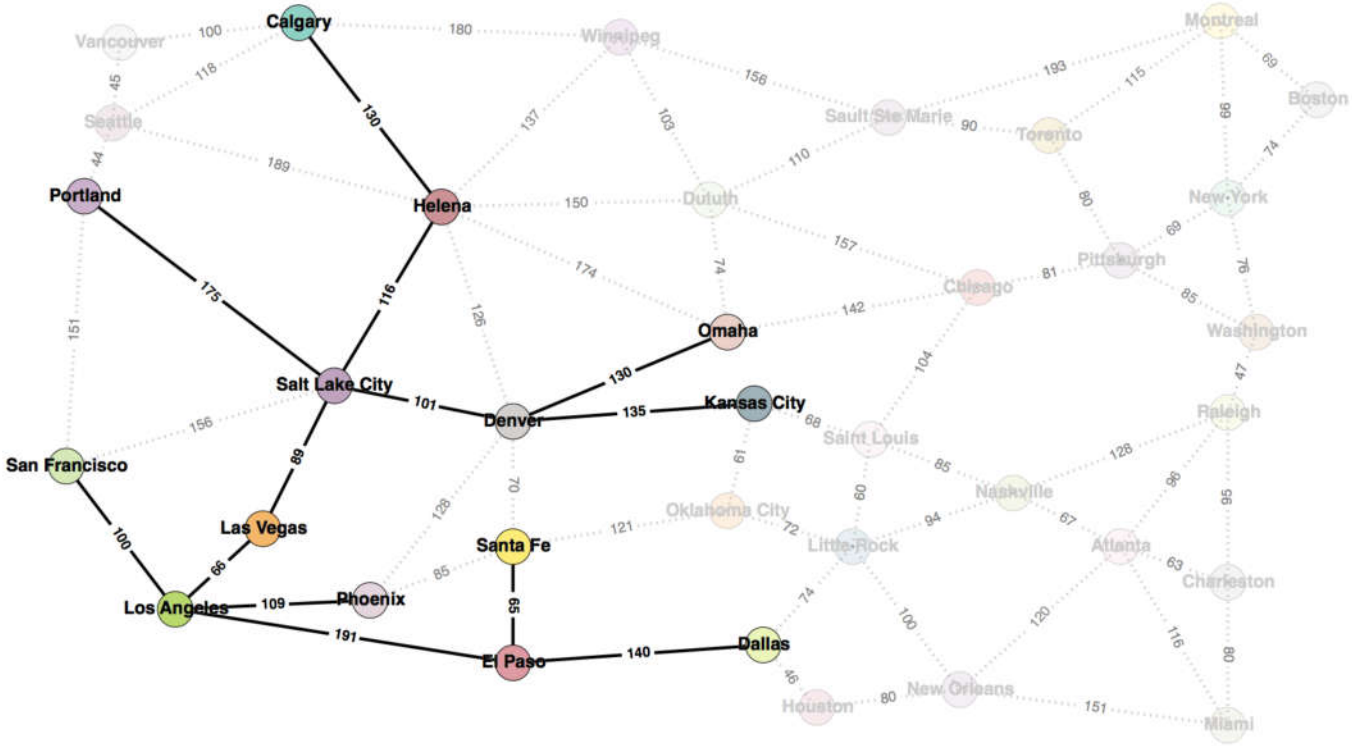
Order of Visit:

S *C* *A* *E* *B* *D* *H* *F* *G*

Examples using the map

Start: Las Vegas

Goal: Calgary



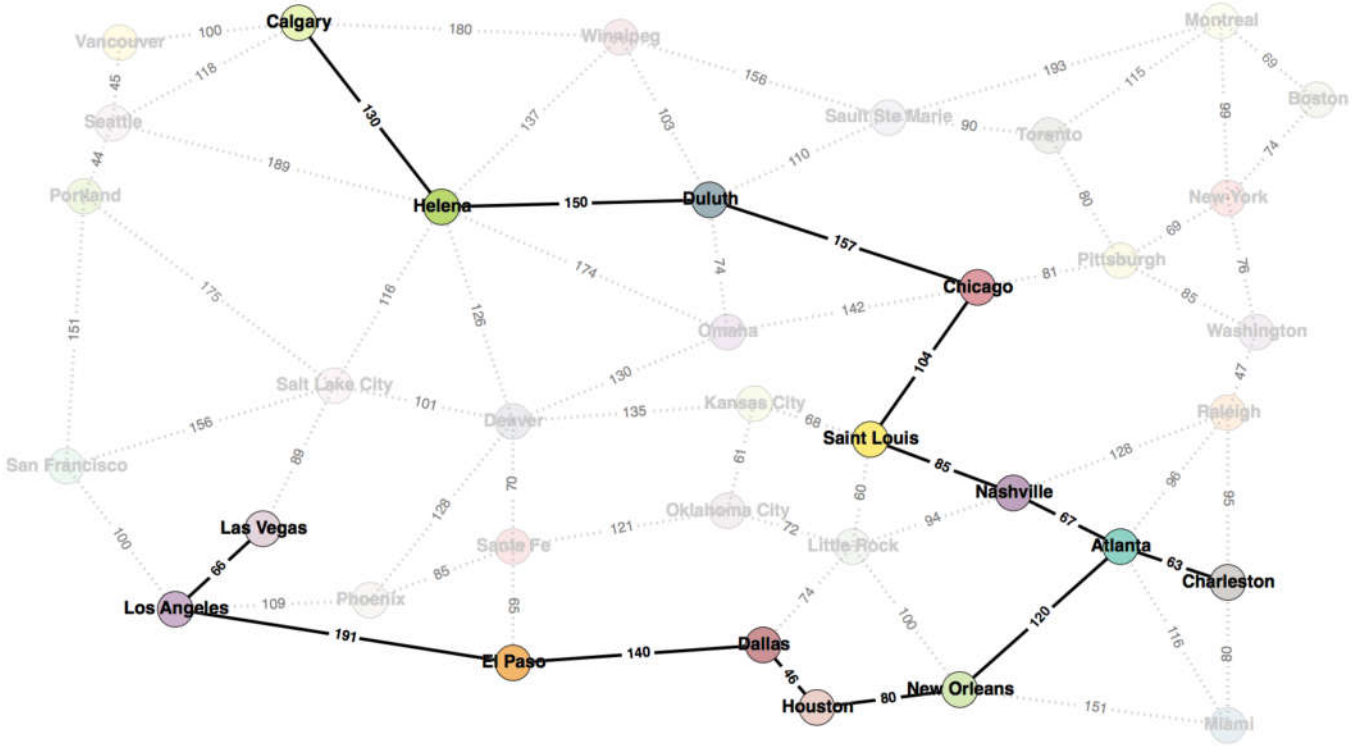
BFS

Order of Visit: Las Vegas, Los Angeles, Salt Lake City, El Paso, Phoenix, San Francisco, Denver, Helena, Portland, Dallas, Santa Fe, Kansas City, Omaha, Calgary.

Examples using the map

Start: Las Vegas

Goal: Calgary



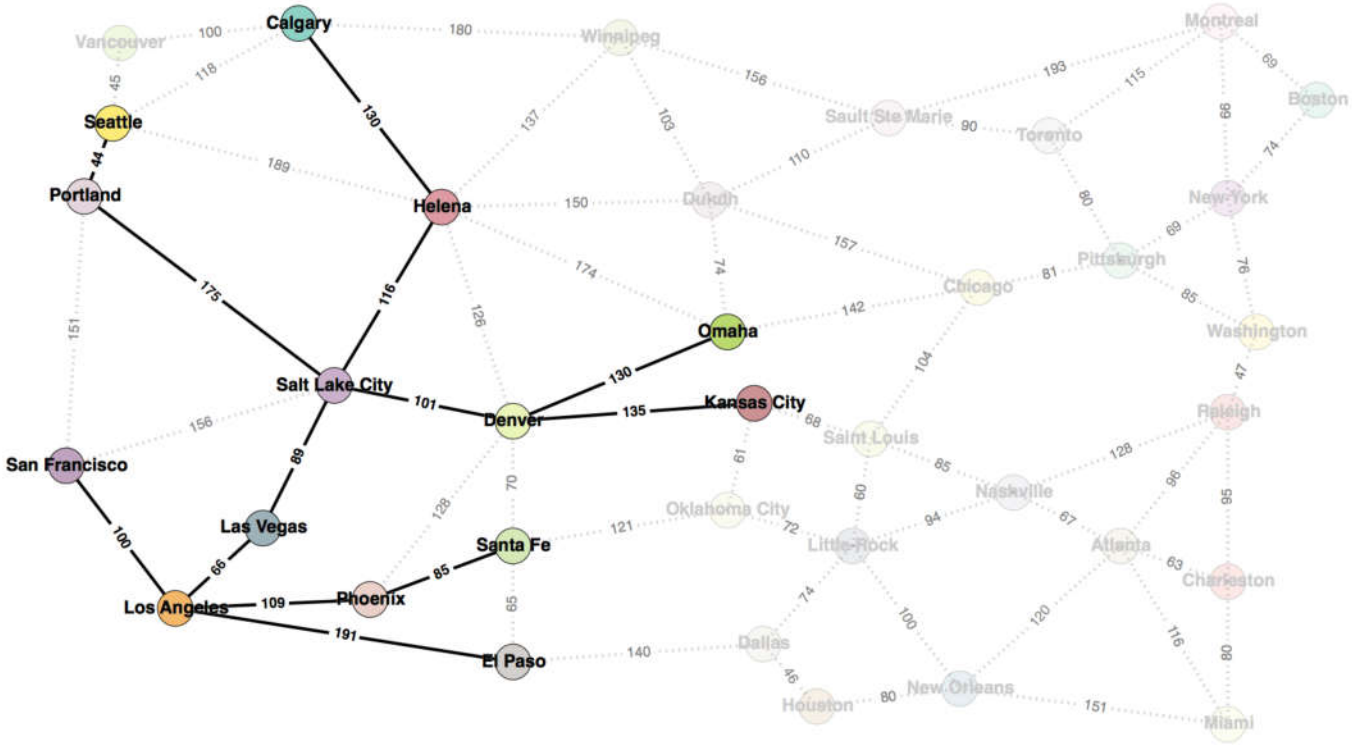
DFS

Order of Visit: Las Vegas, Los Angeles, El Paso, Dallas, Houston, New Orleans, Atlanta, Charleston, Nashville, Saint Louis, Chicago, Duluth, Helena, Calgary.

Examples using the map

Start: Las Vegas

Goal: Calgary



UCS

Order of Visit: Las Vegas, Los Angeles, Salt Lake City, San Francisco, Phoenix, Denver, Helena, El Paso, Santa Fe, Portland, Seattle, Omaha, Kansas City, Calgary.

Credit

- Artificial Intelligence, A Modern Approach. Stuart Russell and Peter Norvig. Third Edition. Pearson Education.

http://aima.cs.berkeley.edu/