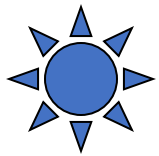
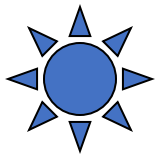


Best First and Uniform Search

Zahoor Tanoli (PhD)

CUI Attock

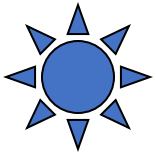
Heuristic



1. Rule of thumb (what, why, when) guarantee to find good but not optimal solution
2. Technique to solve a problem quickly
3. Problems are complex both in terms of time and memory
    1. 8 puzzle $O(b^d) = 3^{20}$
    2. 15 puzzle will be 10^{13}
    3. 24 puzzle will 10^{24}
    4. Task is to convert NP to polynomial problem
4. Techniques for heuristic can be:
    1. Euclidean distance
    2. Manhattan distance

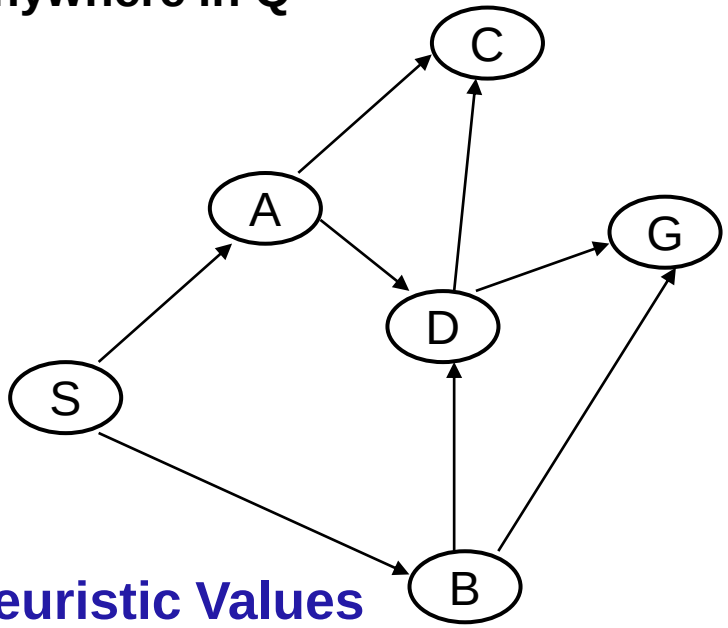
    Select branch with minimum number of misplaced items

Best First Search



Pick "best" (by heuristic value) element of Q;
Add path extensions anywhere in Q

	Q	Visited
1	(10 S)	S
2		
3		
4		
5		



Heuristic Values

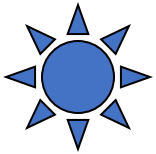
A=2 C=1 S=10

B=3 D=4 G=0

Added paths in blue; heuristic value of node's state is in front

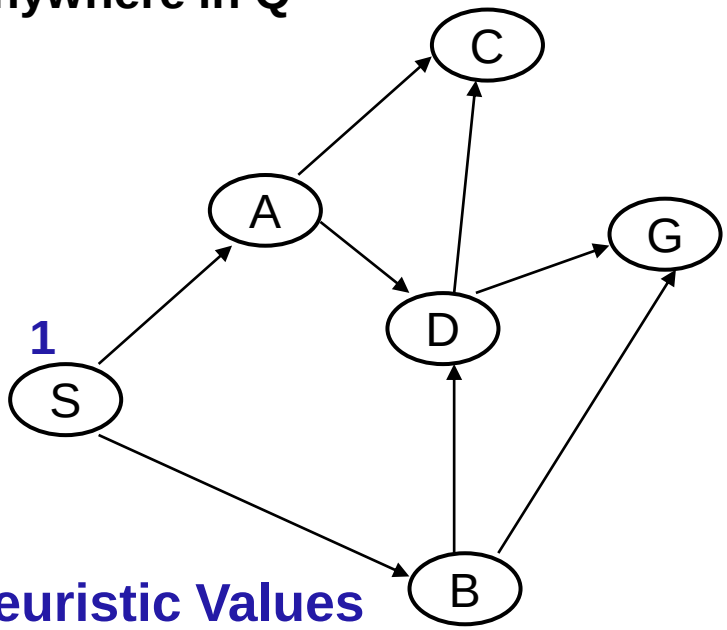
The paths are shown in reversed order; the node's state is the first entry

Best First Search



Pick "best" (by heuristic value) element of Q;
Add path extensions anywhere in Q

	Q	Visited
1	(10 S)	S
2	(2 A S) (3 B S)	A, B, S
3		
4		
5		



Heuristic Values

A=2 C=1 S=10

B=3 D=4 G=0

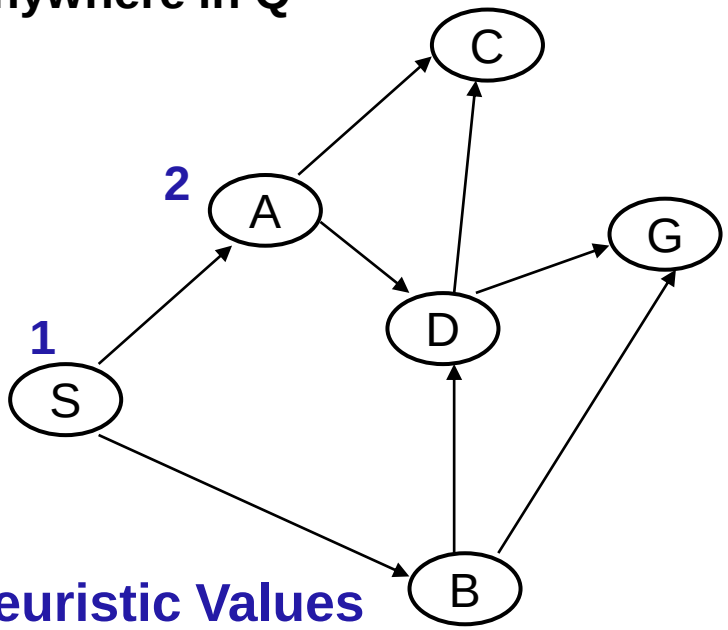
Added paths in blue; heuristic value of node's state is in front

The paths are shown in reversed order; the node's state is the first entry

Best First Search

Pick "best" (by heuristic value) element of Q;
Add path extensions anywhere in Q

	Q	Visited
1	(10 S)	S
2	(2 A S) (3 B S)	A, B, S
3	(1 C A S) (3 B S) (4 D A S)	C,D,B,A,S
4		
5		



Heuristic Values

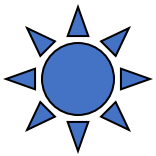
A=2 C=1 S=10

B=3 D=4 G=0

Added paths in blue; heuristic value of node's state is in front

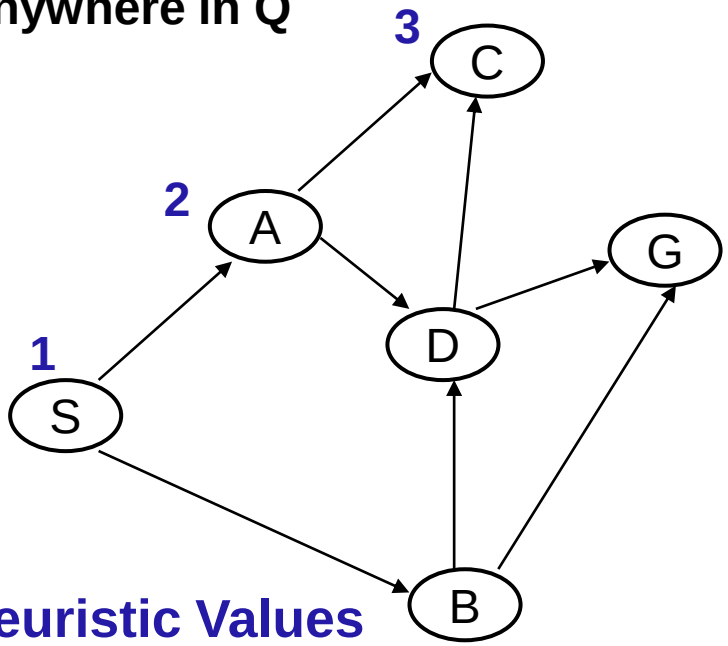
The paths are shown in reversed order; the node's state is the first entry

Best First Search



Pick "best" (by heuristic value) element of Q;
Add path extensions anywhere in Q

	Q	Visited
1	(10 S)	S
2	(2 A S) (3 B S)	A, B, S
3	(1 C A S) (3 B S) (4 D A S)	C,D,B,A,S
4	(3 B S) (4 D A S)	C,D,B,A,S
5		



Heuristic Values

A=2 C=1 S=10

B=3 D=4 G=0

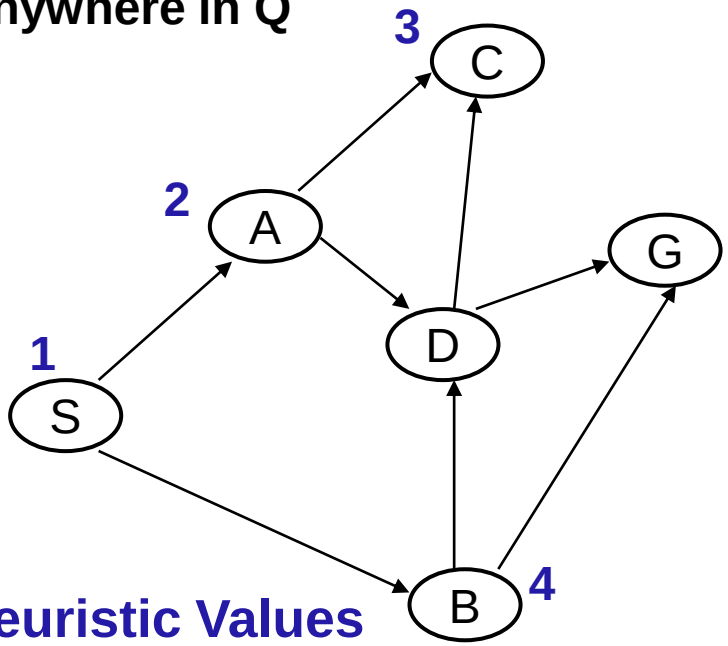
Added paths in blue; heuristic value of node's state is in front

The paths are shown in reversed order; the node's state is the first entry

Best First Search

Pick "best" (by heuristic value) element of Q;
Add path extensions anywhere in Q

	Q	Visited
1	(10 S)	S
2	(2 A S) (3 B S)	A, B, S
3	(1 C A S) (3 B S) (4 D A S)	C,D,B,A,S
4	(3 B S) (4 D A S)	C,D,B,A,S
5	(0 G B S) (4 D A S)	G,C,D,B,A,S



Heuristic Values

A=2 C=1 S=10

B=3 D=4 G=0

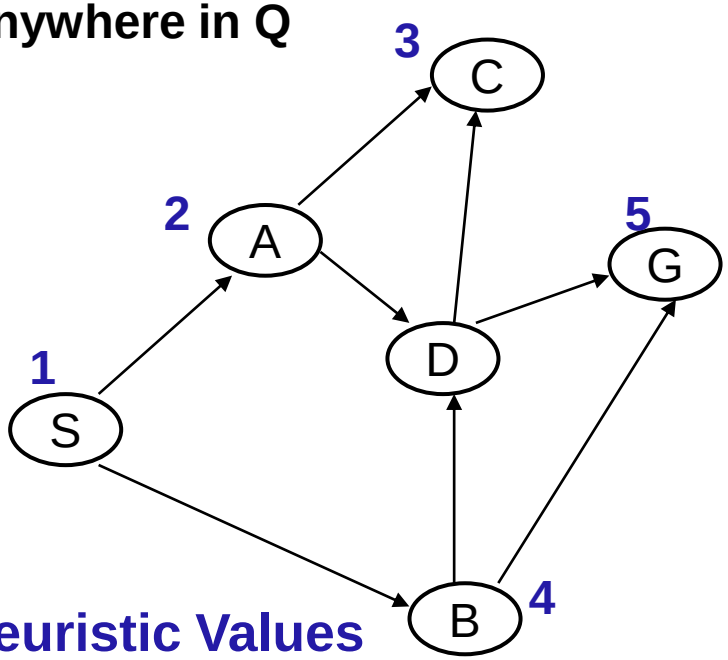
Added paths in blue; heuristic value of node's state is in front

The paths are shown in reversed order; the node's state is the first entry

Best First Search

Pick "best" (by heuristic value) element of Q;
Add path extensions anywhere in Q

	Q	Visited
1	(10 S)	S
2	(2 A S) (3 B S)	A, B, S
3	(1 C A S) (3 B S) (4 D A S)	C,D,B,A,S
4	(3 B S) (4 D A S)	C,D,B,A,S
5	(0 G B S) (4 D A S)	G,C,D,B,A,S



Heuristic Values

A=2 C=1 S=10

B=3 D=4 G=0

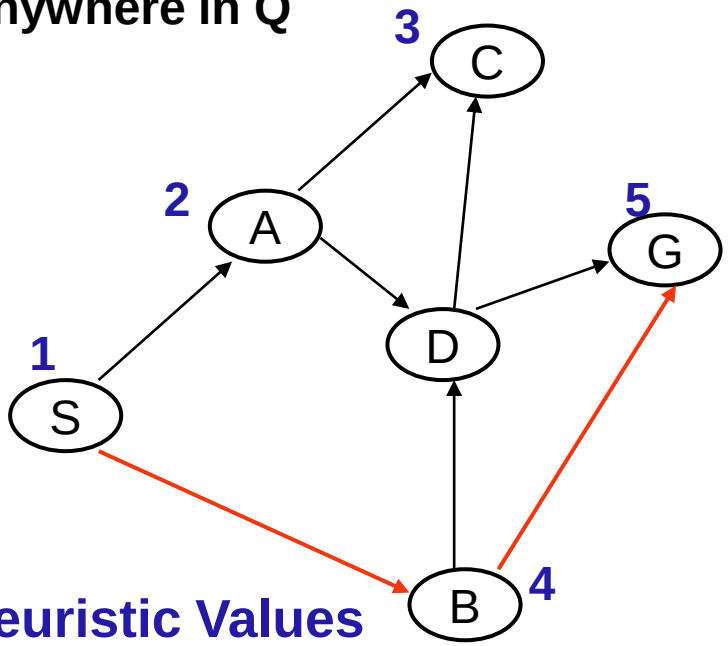
Added paths in blue; heuristic value of node's state is in front

The paths are shown in reversed order; the node's state is the first entry

Best First Search

Pick "best" (by heuristic value) element of Q;
Add path extensions anywhere in Q

	Q	Visited
1	(10 S)	S
2	(2 A S) (3 B S)	A, B, S
3	(1 C A S) (3 B S) (4 D A S)	C,D,B,A,S
4	(3 B S) (4 D A S)	C,D,B,A,S
5	(0 G B S) (4 D A S)	G,C,D,B,A,S



Heuristic Values

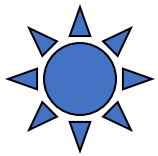
A=2 C=1 S=10

B=3 D=4 G=0

Added paths in blue; heuristic value of node's state is in front

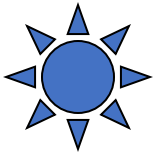
The paths are shown in reversed order; the node's state is the first entry

Classes of Search



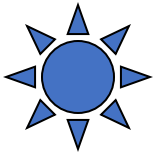
Class	Name	Operation
Any Path Uninformed	Depth-First Breadth-First	Systematic exploration of whole tree until a goal node is found
Any Path Informed	Best First	Uses heuristic measure of goodness of a state, e.g. estimated distance to goal

Classes of Search



Class	Name	Operation
Any Path Uninformed	Depth-First Breadth-First	Systematic exploration of whole tree until a goal node is found
Any Path Informed	Best First	Uses heuristic measure of goodness of a state, e.g. estimated distance to goal
Optimal Uninformed	Uniform Cost	Uses path “length” measure. Finds “shortest” path.

Simple Search Algorithm



A search node is a path from some state X to the start state, e.g., (XBAS)

The state of a search node is the most recent state of the path, e.g. X.

Let Q be a list of search nodes, e.g. ((XBAS) (CBAS)...)

Let S be the start state

1. **Initialize Q with search node (S) as only entry; ~~set Visited = (S)~~**
2. **If Q is empty, fail. Else, pick some search node N from Q**
3. **If state (N) is a goal, return N (we've reached the goal)**
4. **(otherwise) Remove N from Q**
5. **Find all the descendants of state(N) ~~not in Visited and~~ create all the one-step extensions of N to each descendant.**
6. **Add the extended paths to Q; a~~dd children of state(N) to Visited~~**
7. **Go to step 2.**

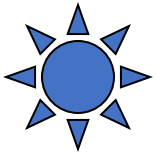
Don't use Visited for Optimal Search

Critical Decisions:

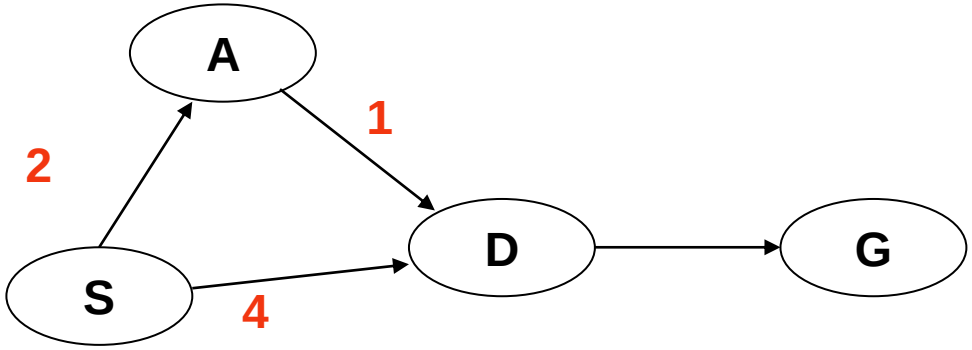
Step 2: Pick N from Q

Step 6: adding extension of N to Q

Why not a Visited list?

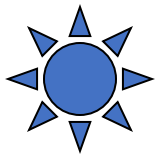


- For the any-path algorithms, the Visited list would not cause us to fail to find a path when one existed, since the path to a state did not matter
- However, the Visited list in connection with UC can cause us to miss the best path



- The shorted path from S to G is (S A D G)
- But, on extending (S), A and D would be added to Visited list and so (S A) would not be extended to (S A D)

Implementing Optimal Search Strategies

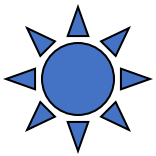


Uniform Cost:

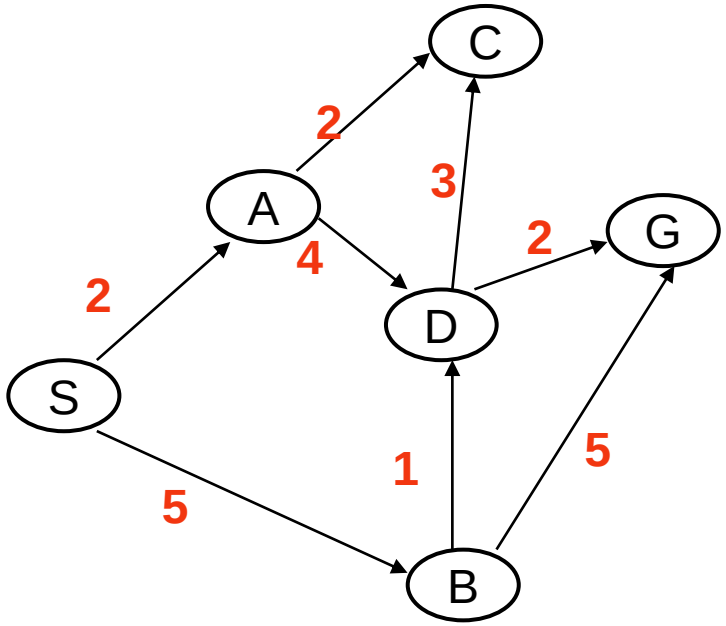
Pick best (measured by path length) element of Q

Add path extensions anywhere in Q.

Uniform Cost (UC)



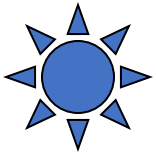
- Like best-first except that it used the "total length (cost)" of a path instead of a heuristic value for the state
- Each link has a "length" or "cost" (which is always greater than 0)
- We want "shortest" or "least cost" path



Total path cost:

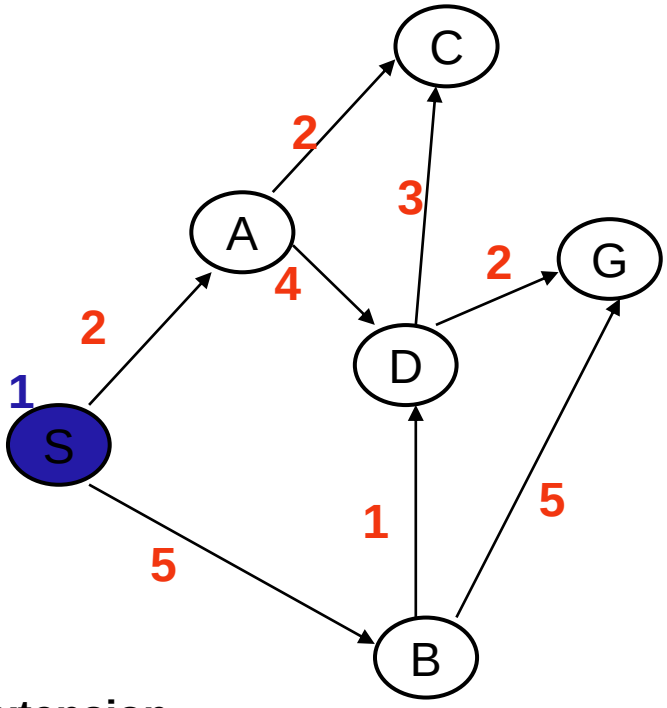
(SAC)	**4**
(SBDG)	**8**
(SADC)	**9**

Uniform Cost (UC)



- **Pick best (by path length) element of Q; Add path extensions anywhere in Q**

	Q
1	**(0 S)**



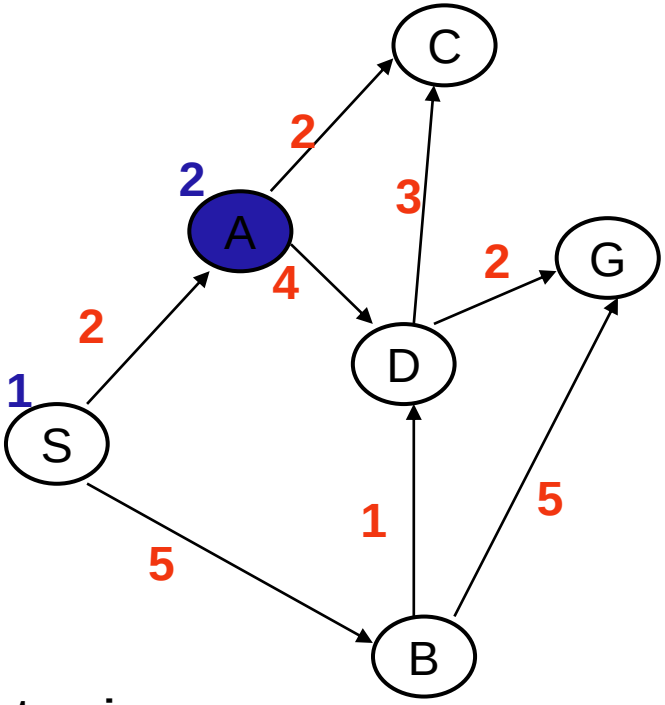
Added paths in blue; underlined paths are chosen for extension

Paths are shown in reverse order; the node's state is the first entry

Uniform Cost (UC)

- **Pick best (by path length) element of Q; Add path extensions anywhere in Q**

	Q
1	**(0 S)**
2	**(2 A S) (5 B S)**



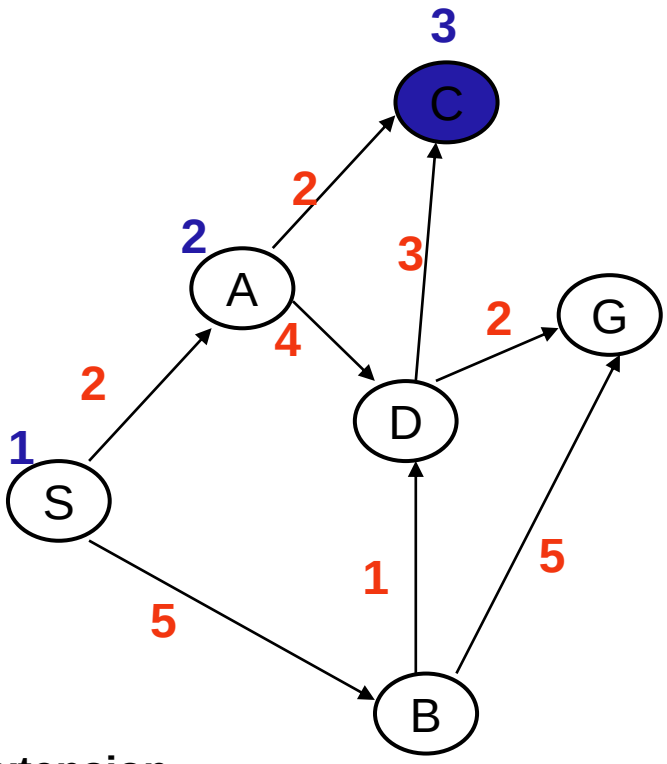
Added paths in blue; underlined paths are chosen for extension

Paths are shown in reverse order; the node's state is the first entry

Uniform Cost (UC)

- Pick best (by path length) element of Q; Add path extensions anywhere in Q

	Q
1	(0 S)
2	(2 AS) (5 BS)
3	(4 CAS) (6 DAS) (5 BS)



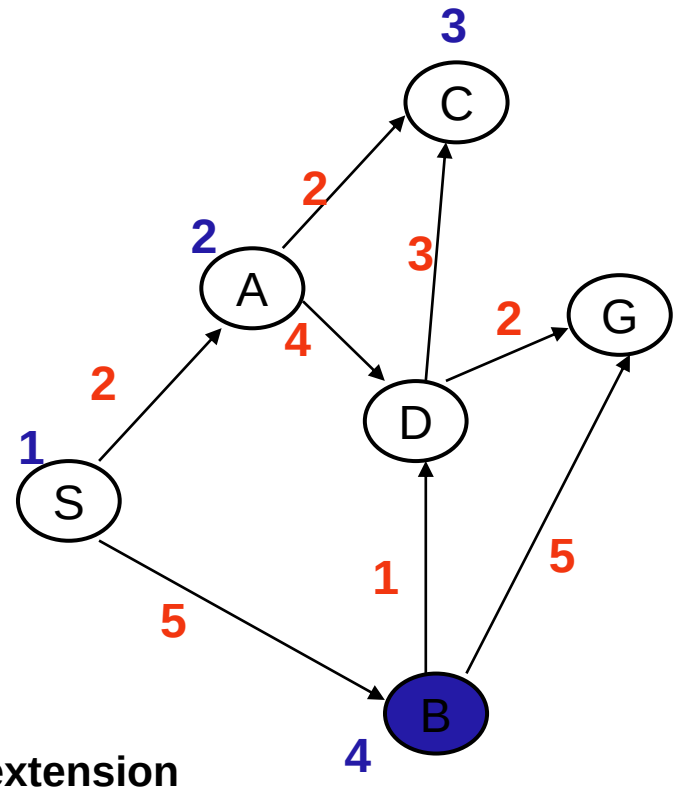
Added paths in blue; underlined paths are chosen for extension

Paths are shown in reverse order; the node's state is the first entry

Uniform Cost (UC)

- Pick best (by path length) element of Q; Add path extensions anywhere in Q

	Q
1	(0 S)
2	(2 AS) (5 BS)
3	(4 CAS) (6 DAS) (5 BS)
4	(6 DAS) (5 BS)



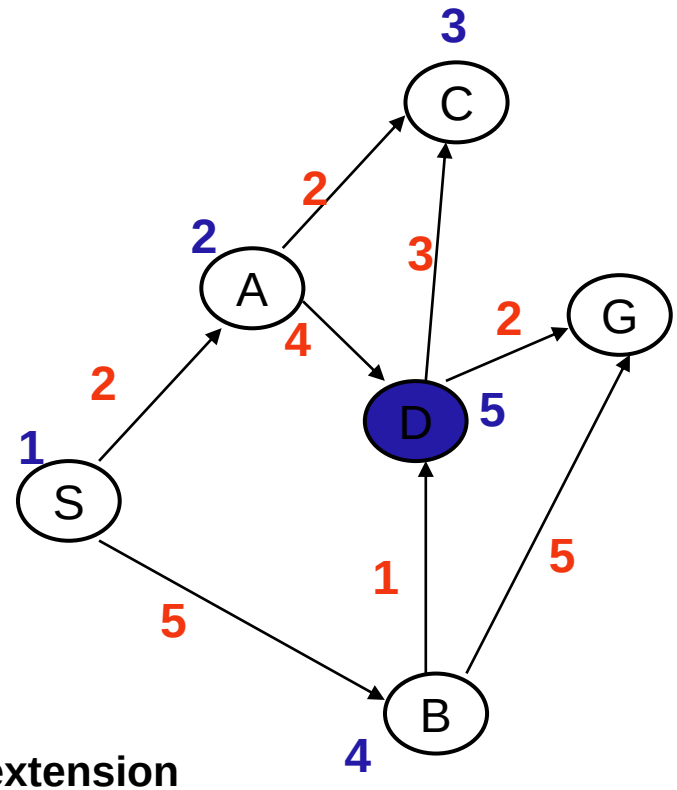
Added paths in blue; underlined paths are chosen for extension

Paths are shown in reverse order; the node's state is the first entry

Uniform Cost (UC)

- Pick best (by path length) element of Q; Add path extensions anywhere in Q

	Q
1	(0 S)
2	(2 AS) (5 BS)
3	(4 CAS) (6 DAS) (5 BS)
4	(6 DAS) (5 BS)
5	(6 DBS) (10 GBS) (6 DAS)



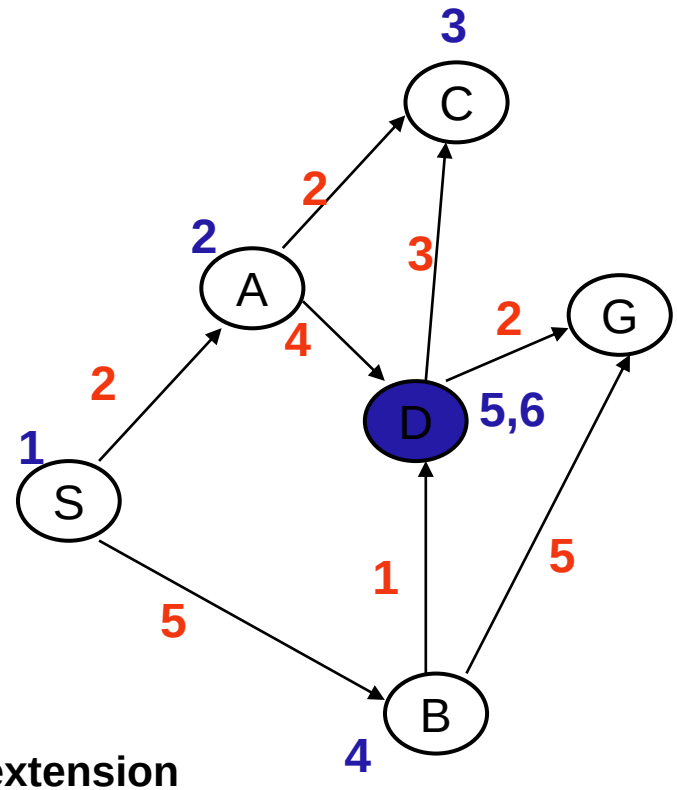
Added paths in blue; underlined paths are chosen for extension

Paths are shown in reverse order; the node's state is the first entry

Uniform Cost (UC)

- Pick best (by path length) element of Q; Add path extensions anywhere in Q

	Q
1	(0 S)
2	(2 AS) (5 BS)
3	(4 CAS) (6 DAS) (5 BS)
4	(6 DAS) (5 BS)
5	(6 DBS) (10 GBS) (6 DAS)
6	(8 GDBS) (9 CDBS) (10 GBS) (6 DAS)



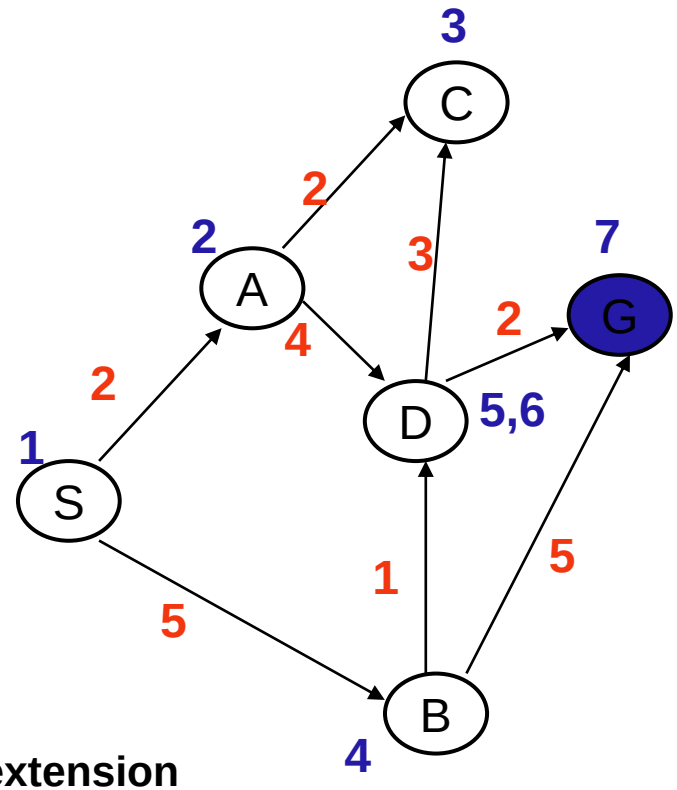
Added paths in blue; underlined paths are chosen for extension

Paths are shown in reverse order; the node's state is the first entry

Uniform Cost (UC)

- Pick best (by path length) element of Q; Add path extensions anywhere in Q

	Q
1	(0 S)
2	(2 AS) (5 BS)
3	(4 CAS) (6 DAS) (5 BS)
4	(6 DAS) (5 BS)
5	(6 DBS) (10 GBS) (6 DAS)
6	(8 GDBS) (9 CDBS) (10 GBS) (6 DAS)
7	(8 GDAS) (9 CDAS) (8 GDBS) (9 CDBS) (10 GBS)



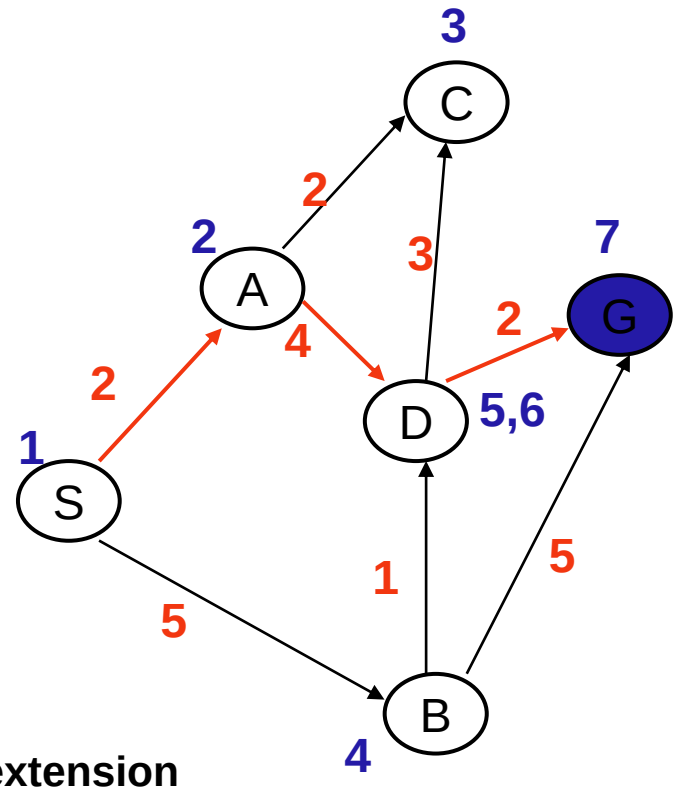
Added paths in blue; underlined paths are chosen for extension

Paths are shown in reverse order; the node's state is the first entry

Uniform Cost (UC)

- Pick best (by path length) element of Q; Add path extensions anywhere in Q

	Q
1	(0 S)
2	(2 AS) (5 BS)
3	(4 CAS) (6 DAS) (5 BS)
4	(6 DAS) (5 BS)
5	(6 DBS) (10 GBS) (6 DAS)
6	(8 GDBS) (9 CDBS) (10 GBS) (6 DAS)
7	(8 GDAS) (9 CDAS) (8 GDBS) (9 CDBS) (10 GBS)



Added paths in blue; underlined paths are chosen for extension

Paths are shown in reverse order; the node's state is the first entry

Why not stop on first visiting a goal?

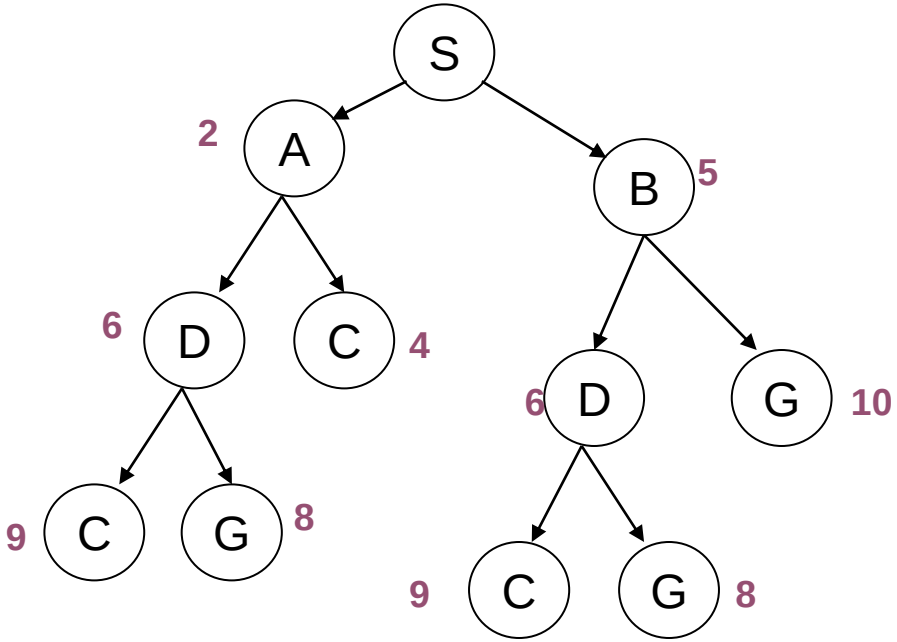
- When doing Uniform Cost, it is not correct to stop the search when the first path to a goal is generated, that is, when a node whose state is a goal is added to Q.

- We must wait until such a path is pulled off the Q and tested in step 3. It is only at this point that we are sure it is the shortest path to a goal since there are no other shorter paths that remain unexpanded

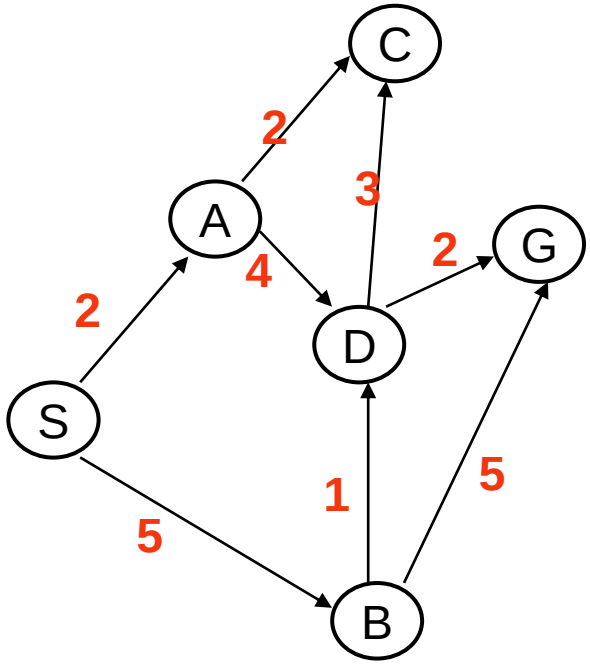
- This contrasts with the not-optimal searches where the choice of where to test for a goal was a matter of convenience and efficiency, not correctness

- In the previous example, a path to G was generated at step 5, but it was a different & shorter path at step 7 that was accepted.

Uniform Cost (UC) – Another way to see it

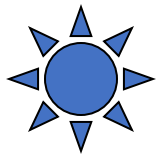


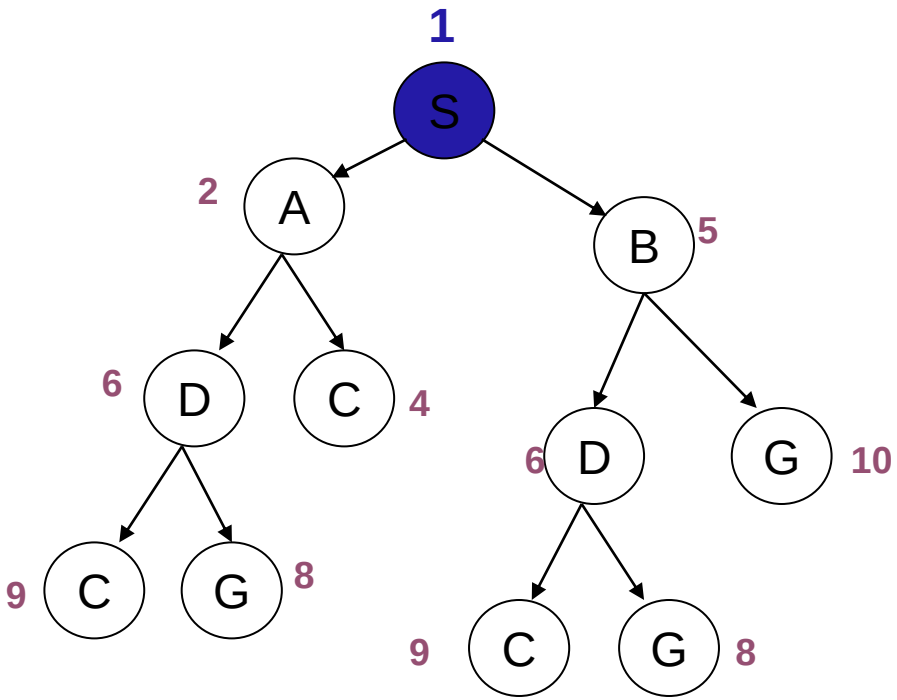
Total path cost

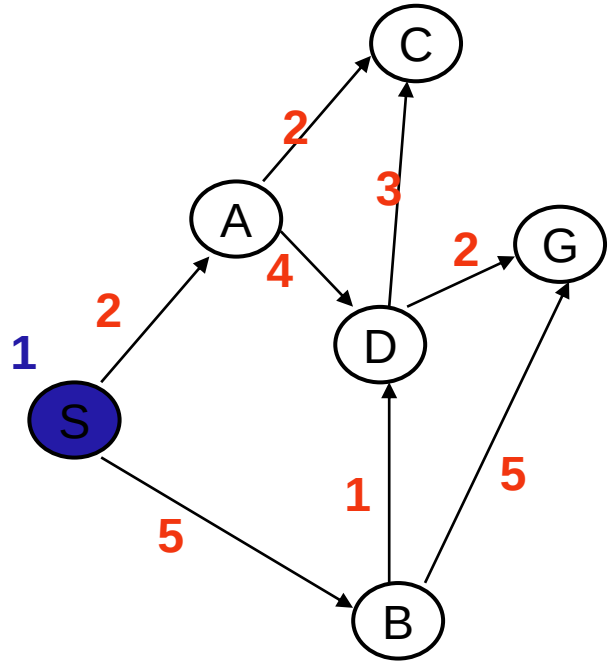


UC enumerates paths in order of total path cost!

Uniform Cost (UC) – Another way to see it





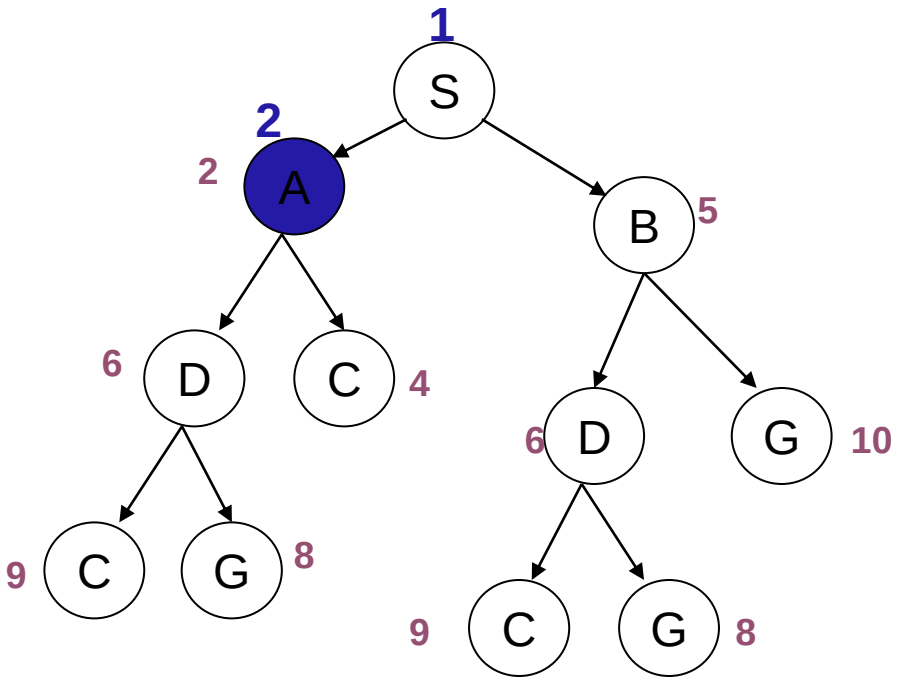


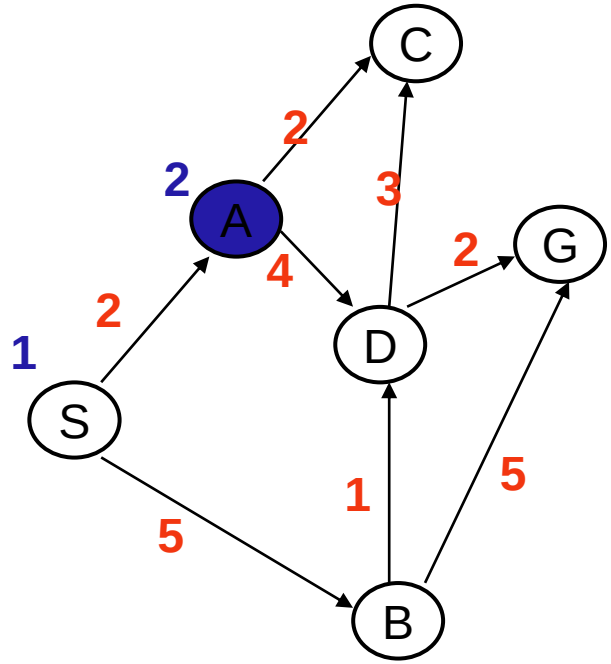
Total path cost

Order pulled off of Q (expanded)

UC enumerates paths in order of total path cost!

Uniform Cost (UC) – Another way to see it



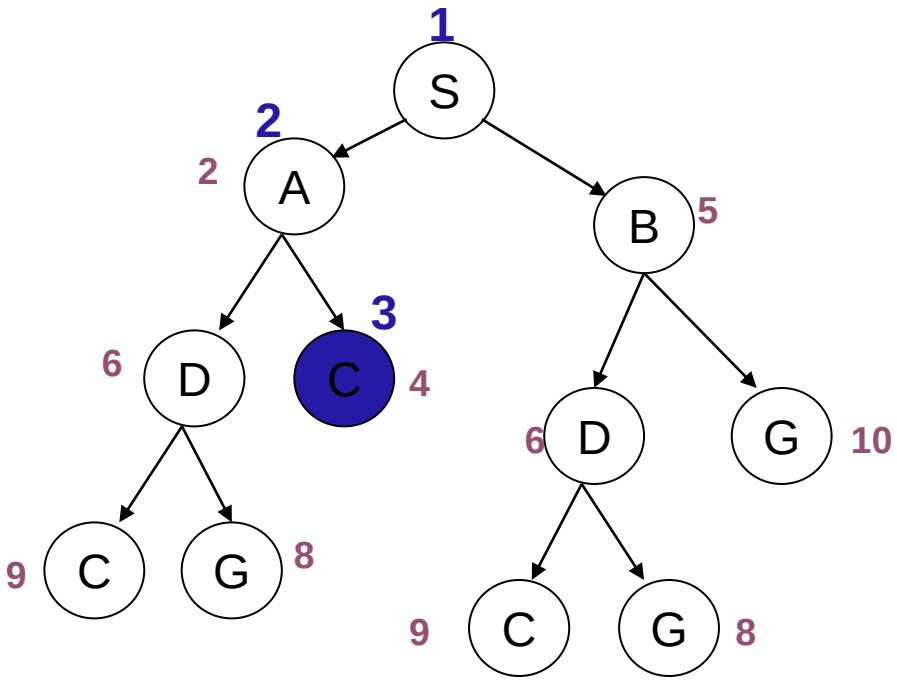


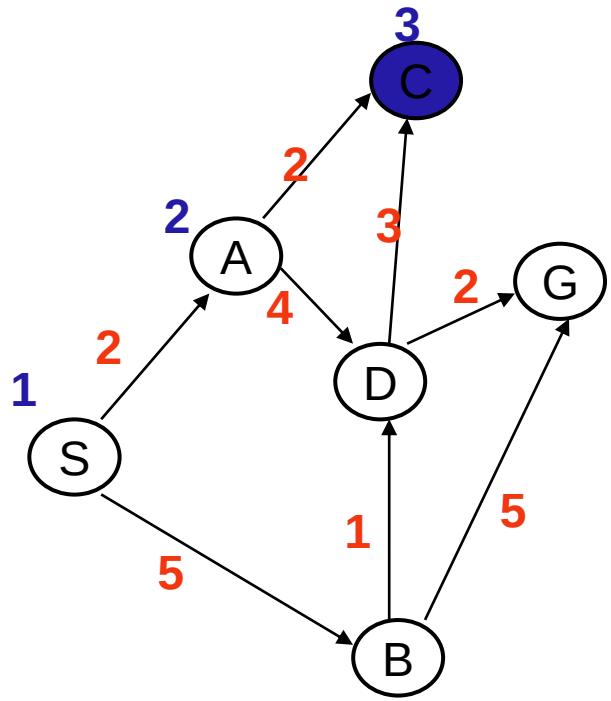
Total path cost

Order pulled off of Q (expanded)

UC enumerates paths in order of total path cost!

Uniform Cost (UC) – Another way to see it



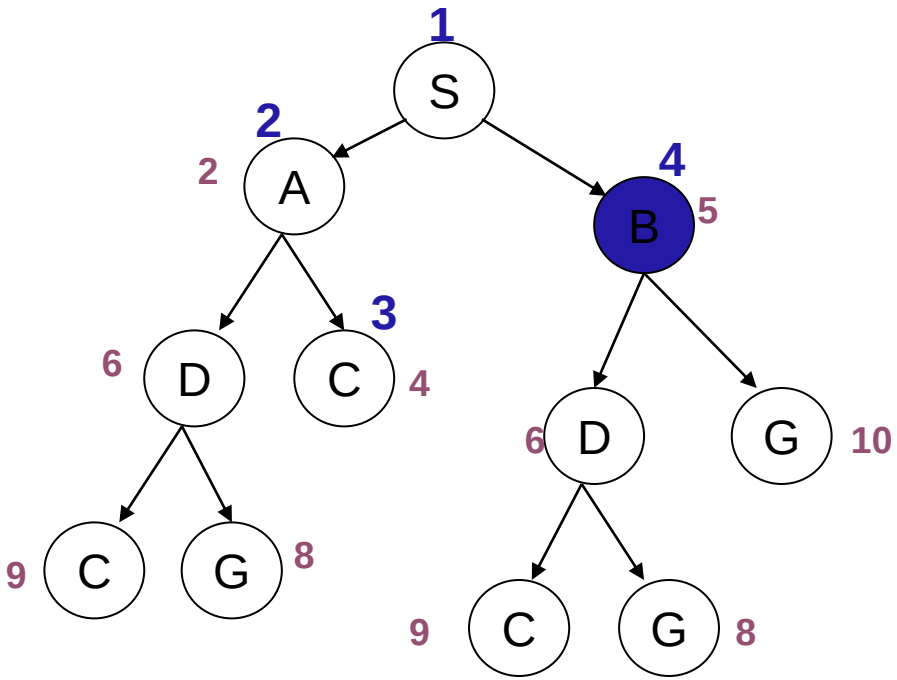


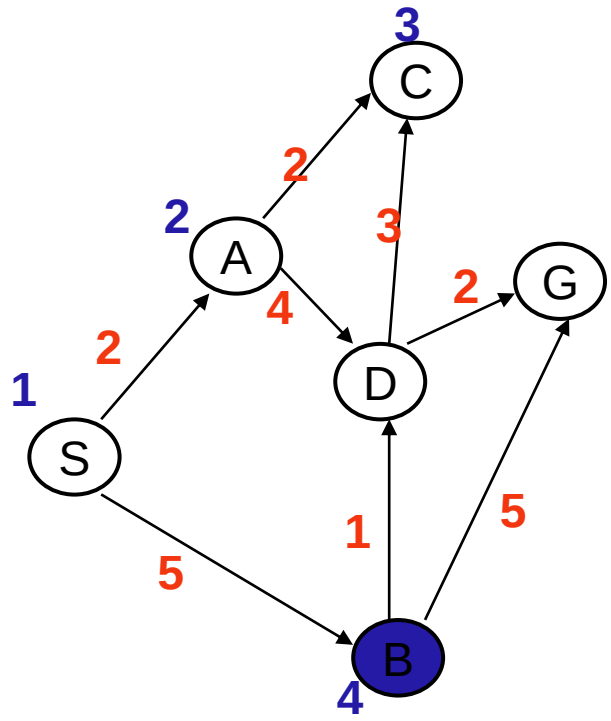
Total path cost

Order pulled off of Q (expanded)

UC enumerates paths in order of total path cost!

Uniform Cost (UC) – Another way to see it



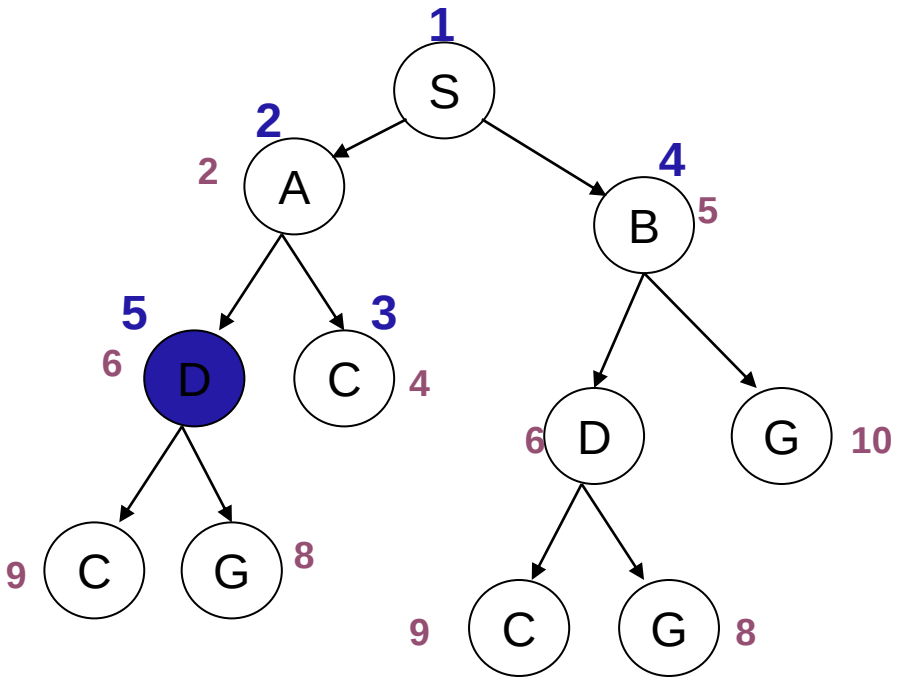


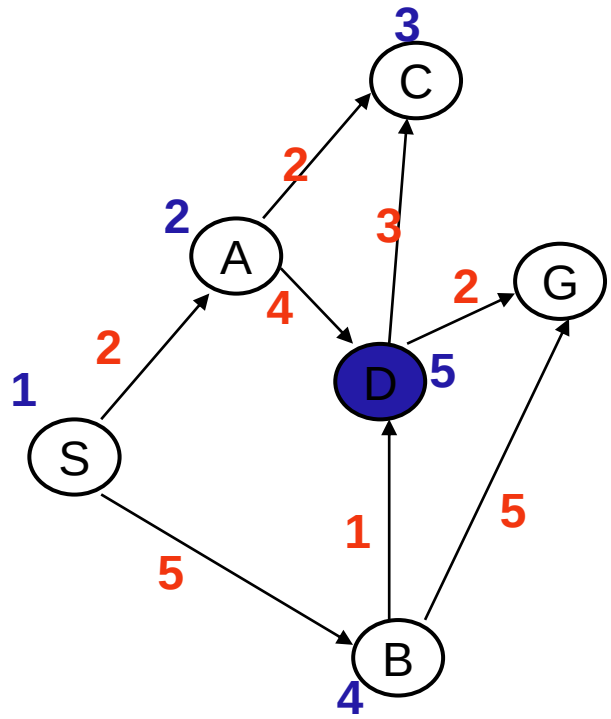
Total path cost

Order pulled off of Q (expanded)

UC enumerates paths in order of total path cost!

Uniform Cost (UC) – Another way to see it



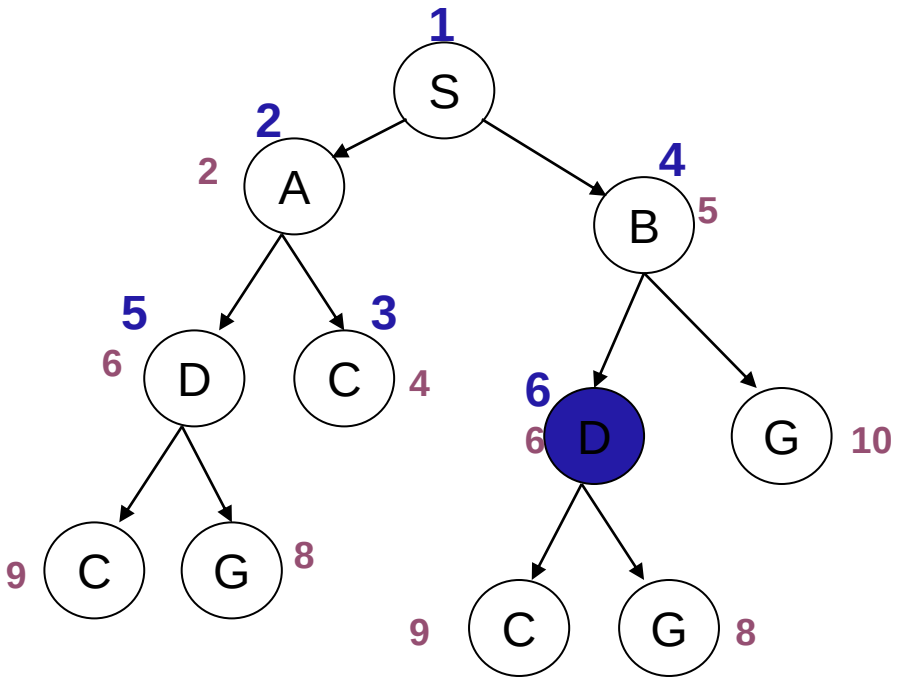


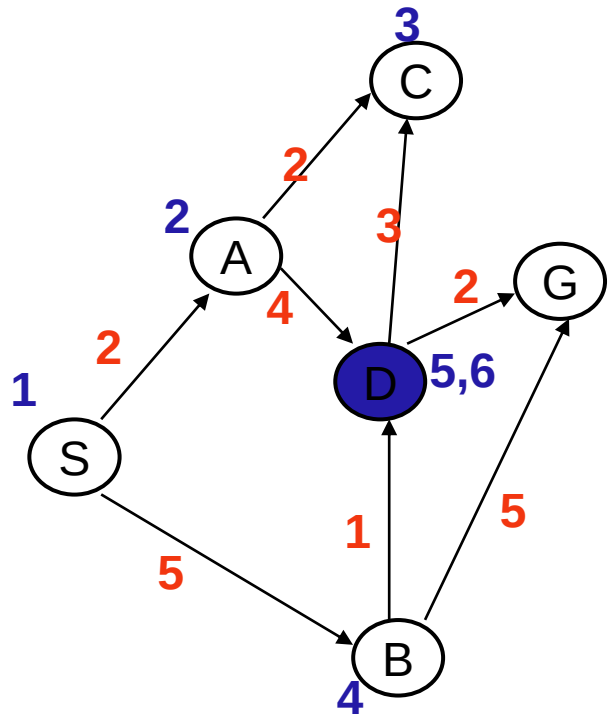
Total path cost

Order pulled off of Q (expanded)

UC enumerates paths in order of total path cost!

Uniform Cost (UC) – Another way to see it



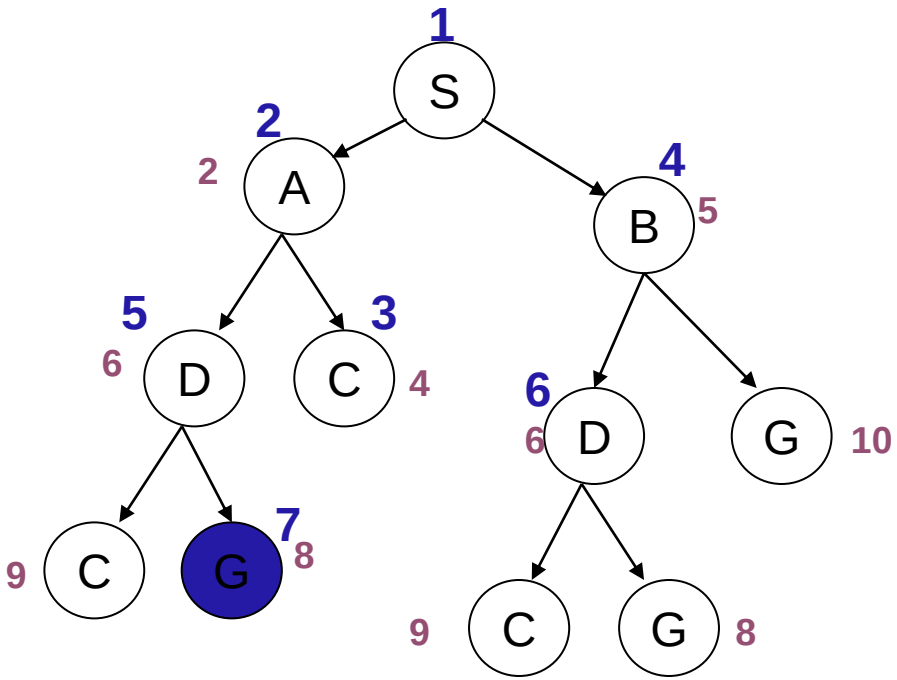


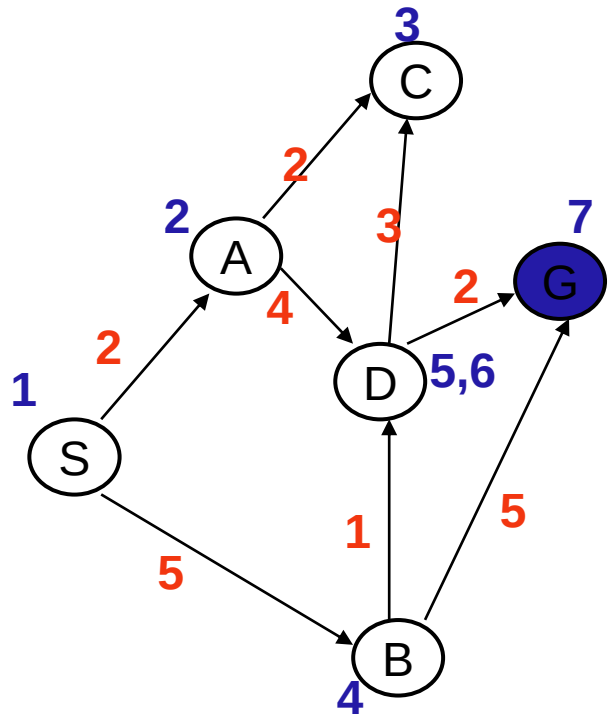
Total path cost

Order pulled off of Q (expanded)

UC enumerates paths in order of total path cost!

Uniform Cost (UC) – Another way to see it



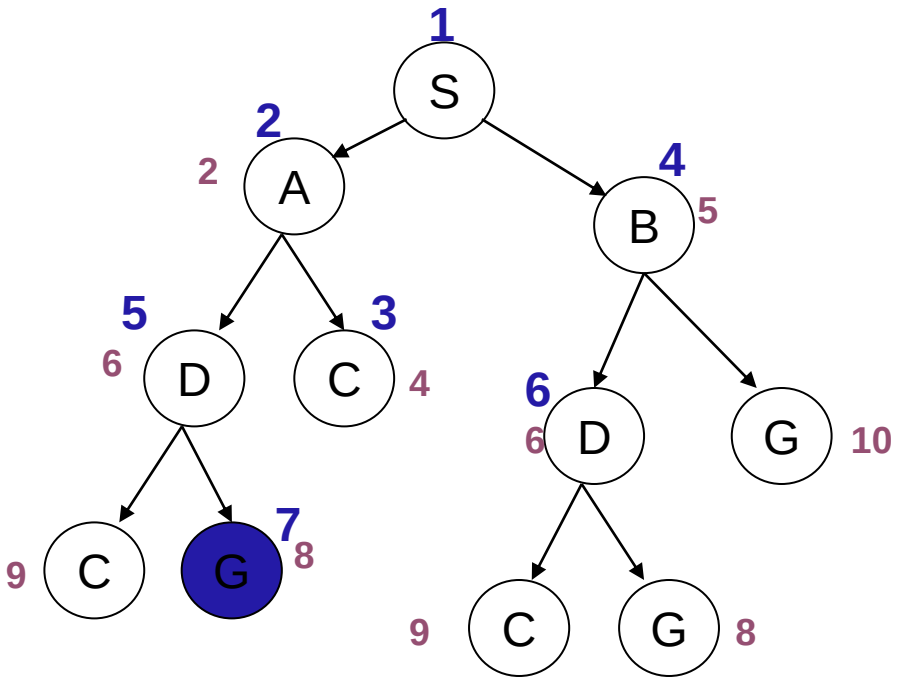


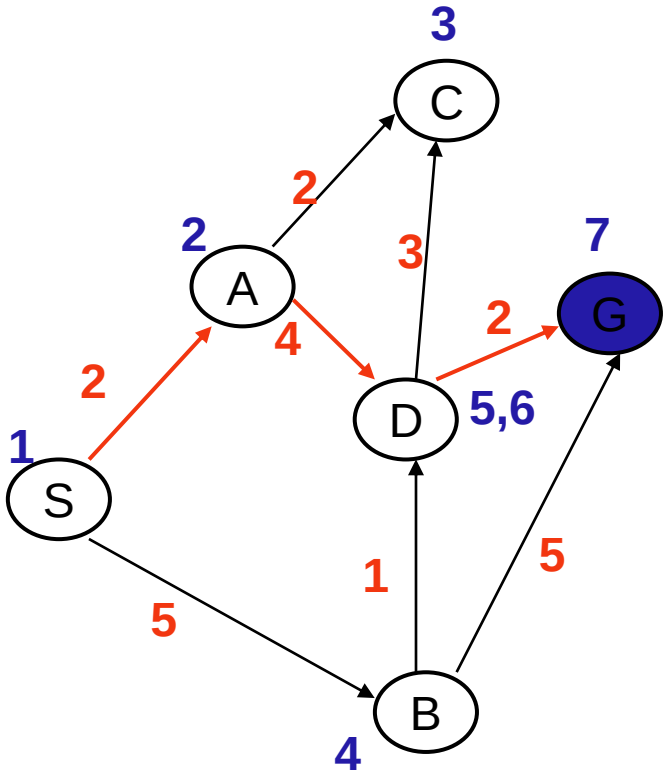
Total path cost

Order pulled off of Q (expanded)

UC enumerates paths in order of total path cost!

Uniform Cost (UC) – Another way to see it



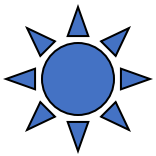


Total path cost

Order pulled off of Q (expanded)

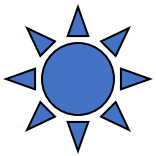
UC enumerates paths in order of total path cost!

Classes of Search



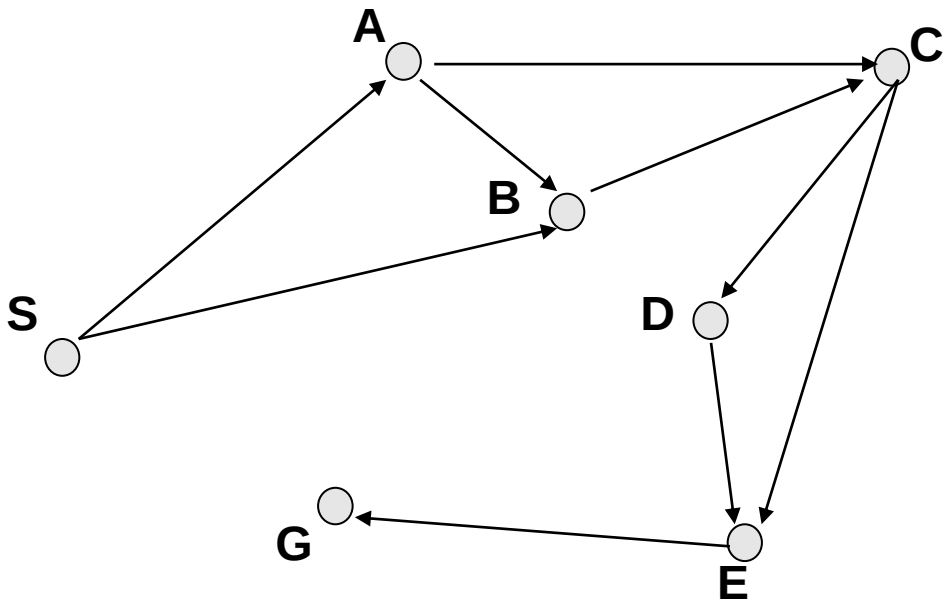
Class	Name	Operation
Any Path Uninformed	Depth-First Breadth-First	Systematic exploration of whole tree until a goal node is found
Any Path Informed	Best First	Uses heuristic measure of goodness of a state, e.g. estimated distance to goal
Optimal Uninformed	Uniform Cost	Uses path "length" measure. Finds "shortest" path.
Optimal Informed	A*	Uses path "length" measure and heuristic. Finds "shortest" path

Goal Direction

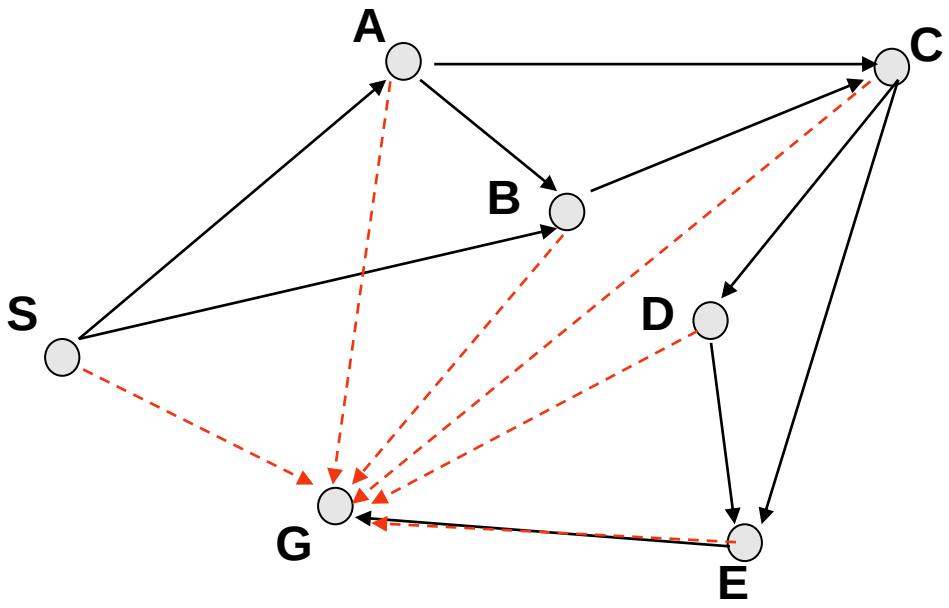


- UC is really trying to identify the shortest path to every state in the graph in order. It has no particular bias to finding a path to a goal early in the search
- We can introduce such a bias by means of heuristic function h(N), which is an estimate (h) of the distance from a state to the goal
- Instead of enumerating paths in order of just length (g), enumerate paths in terms of f= estimated total path length = g + h
- An estimate that always underestimates the real path length to the goal is called admissible. Straight line distance is admissible estimate for path length in Euclidean space.
- Use of an admissible estimate guarantees that UC will still find the shortest path.
- UC with an admissible estimate is known as A*.

Straight Line Estimate



Straight Line Estimate



Straight Line Estimate