COMP3411 Week 03 Tutorial

Yifan He

z5173587@unsw.edu.au
`https://github.com/hharryyf/COMP3411-24T1-tutoring`

Path Finding Algorithms

- A start state S and a terminal state T with some intermediate states, find a path from S to T.
- Many path-finding algorithms
  1. visited = {},DS = {S}
  2. Repeat the following until we reach T
  3. Pop a node n from DS with the minimum $f(n)$
  4. If n has not been visited before,
     1. Insert n to visited.
     2. $\forall v \in next(n)$ and $v \notin visited$, insert v into DS
- DFS, BFS, Greedy, UCS, and A*
  - DFS: Stack
  - BFS: Queue
  - UCS, A*, Greedy: Priority Queue

Path Finding Algorithms

Algorithm	Time	Space	Optimal	Complete
DFS	$O(b^d)$	$O(d)$	No	No
BFS	$O(b^d)$	$O(b^d)$	Yes	Yes
UCS	$O(b^d)$	$O(b^d)$	Yes	Yes
Greedy	$O(b^d)$	$O(b^d)$	No	No
A*	$O(b^d)$	$O(b^d)$	Yes	Yes

Table: Graph with branching factor b, depth d, not memorizing visited states

Q3

Prove each of the following statements, or give a counterexample:

- Breadth First Search is a special case of Uniform Cost Search

Q3

Prove each of the following statements, or give a counterexample:

- Breadth First Search is a special case of Uniform Cost Search
  - Yes
  - BFS is essentially UCS when all edges have the same cost

Q3

- Breadth First Search, Depth First Search, and Uniform Cost Search are special cases of Greedy search.

Q3

- Breadth First Search, Depth First Search, and Uniform Cost Search are special cases of Greedy search.
  - Yes
  - Greedy search reduces to BFS when $f(n) =$ the number of edges from the start node
  - Greedy search reduces to UCS when $f(n) = g(n)$
  - Greedy search reduces to DFS when $f(n) =$ -the number of edges from the start node

Q3

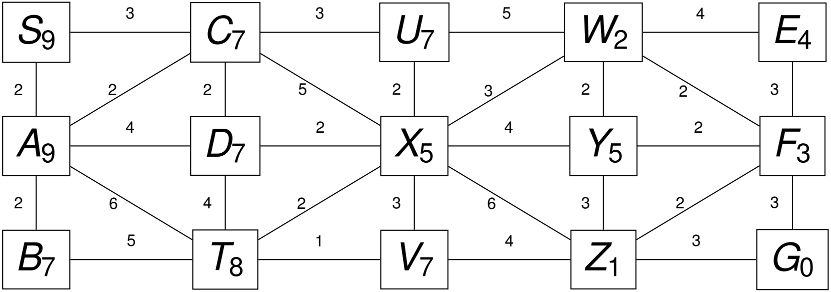
- Uniform Cost Search is a special case of A*Search

Q3

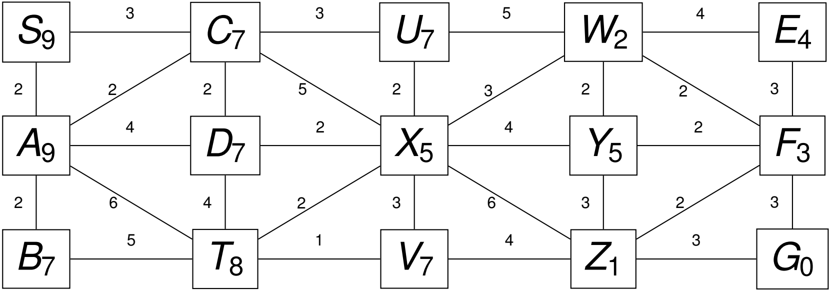
- Uniform Cost Search is a special case of A*Search
  - Yes
  - A* reduces to UCS when $h(n) = 0$

Q1

- Trace the search algorithm (alphabetical order, skip repeated states).

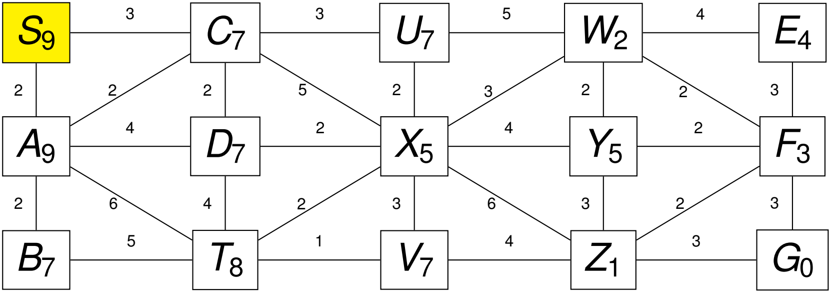


Q1 (DFS)



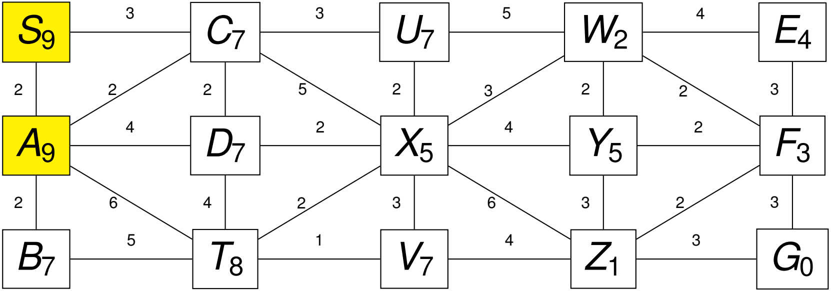
- Visited nodes:
- Candidate nodes: S

Q1 (DFS)



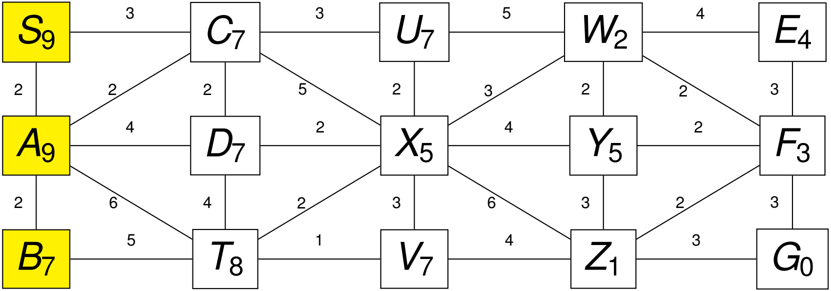
- Visited nodes: S
- Candidate nodes: A,C

Q1 (DFS)



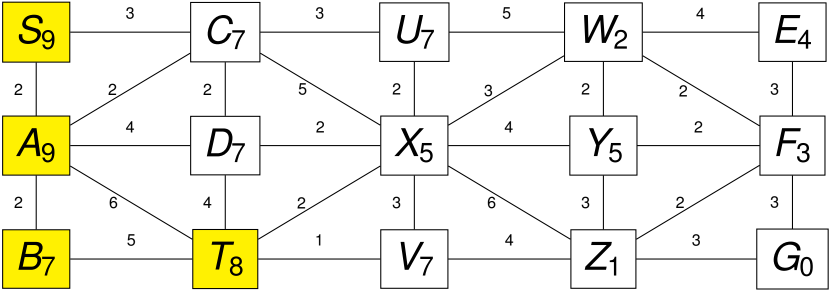
- Visited nodes: S,A
- Candidate nodes: B,D,T,C

Q1 (DFS)



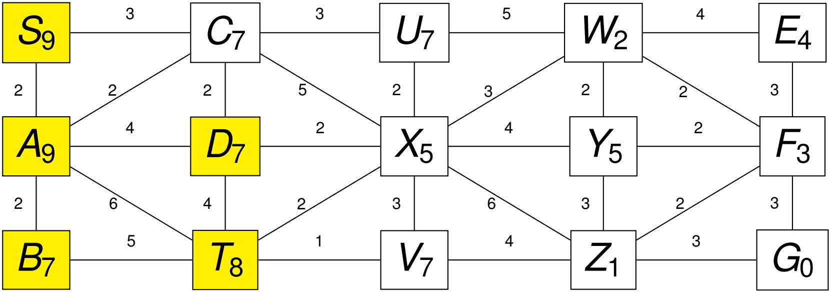
- Visited nodes: S,A,B
- Candidate nodes: T,D,T,C

Q1 (DFS)



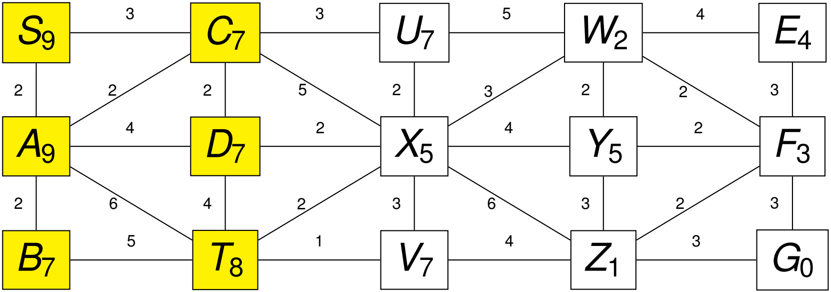
- Visited nodes: S,A,B,T
- Candidate nodes: D,V,X,D,T,C

Q1 (DFS)



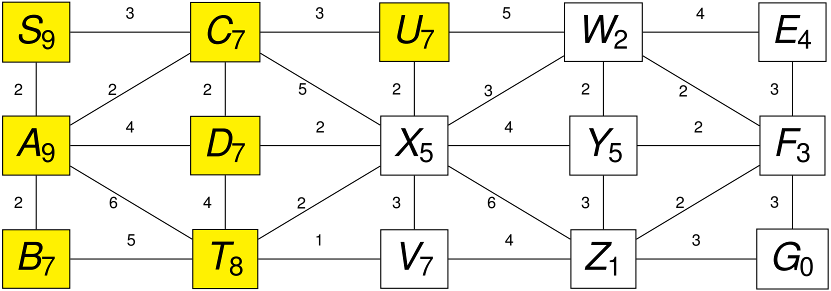
- Visited nodes: S,A,B,T,D
- Candidate nodes: C,X,V,X,D,T,C

Q1 (DFS)



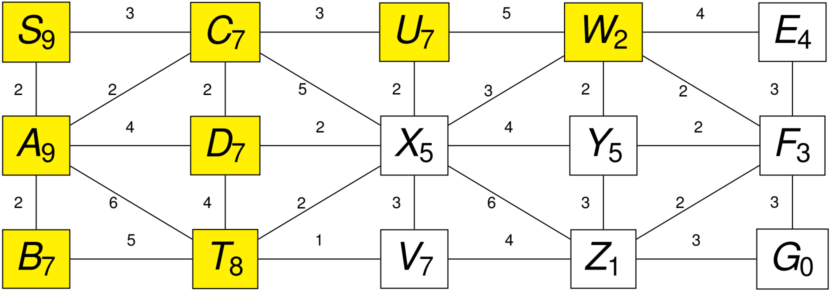
- Visited nodes: S,A,B,T,D,C
- Candidate nodes: U,X,X,V,X,D,T,C

Q1 (DFS)



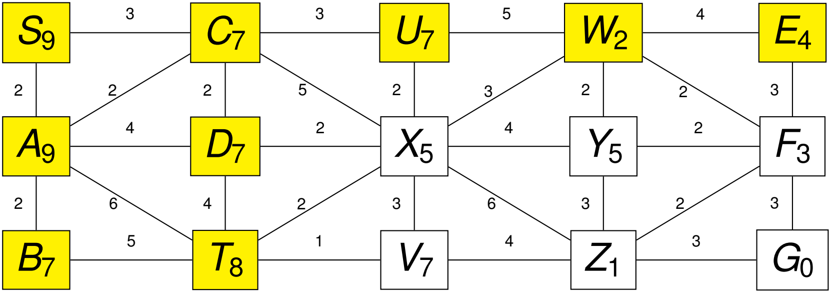
- Visited nodes: S,A,B,T,D,C,U
- Candidate nodes: W,X,X,X,V,X,D,T,C

Q1 (DFS)



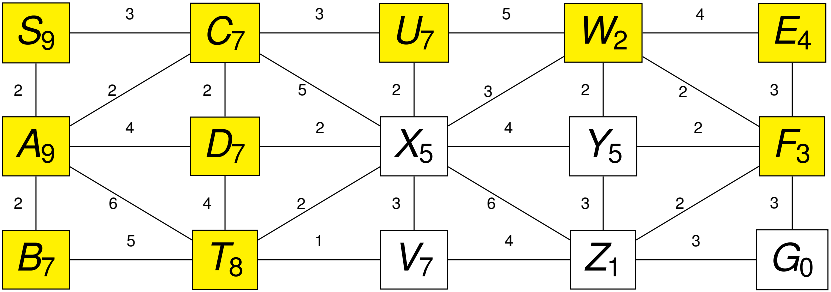
- Visited nodes: S,A,B,T,D,C,U,W
- Candidate nodes: E,F,X,Y,X,X,X,V,X,D,T,C

Q1 (DFS)



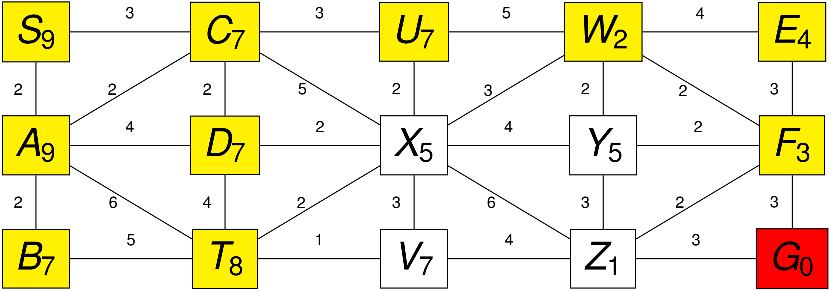
- Visited nodes: S,A,B,T,D,C,U,W,E
- Candidate nodes: F,F,X,Y,X,X,X,V,X,D,T,C

Q1 (DFS)



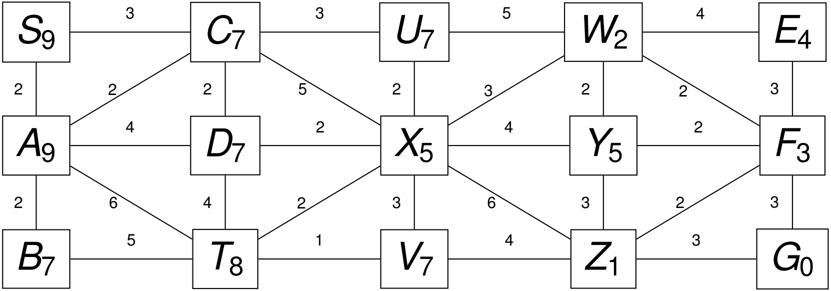
- Visited nodes: S,A,B,T,D,C,U,W,E,F
- Candidate nodes: G,Y,Z,F,X,Y,X,X,X,V,X,D,T,C

Q1 (DFS)



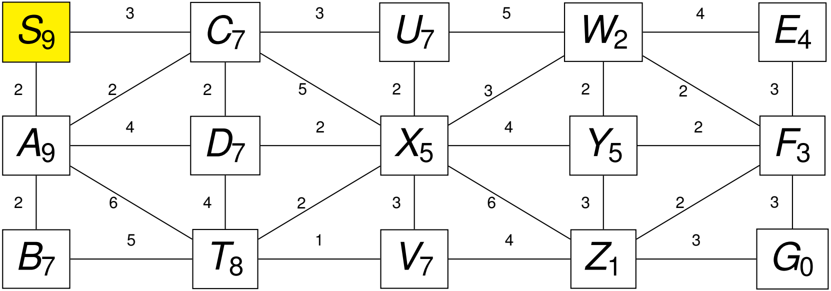
- Visited nodes: S,A,B,T,D,C,U,W,E,F,G
- Candidate nodes: Y,Z,F,X,Y,X,X,X,V,X,D,T,C

Q1 (BFS)



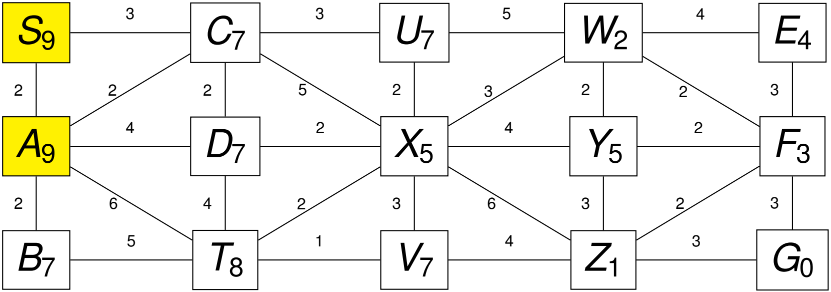
- Visited nodes:
- Candidate nodes: S

Q1 (BFS)



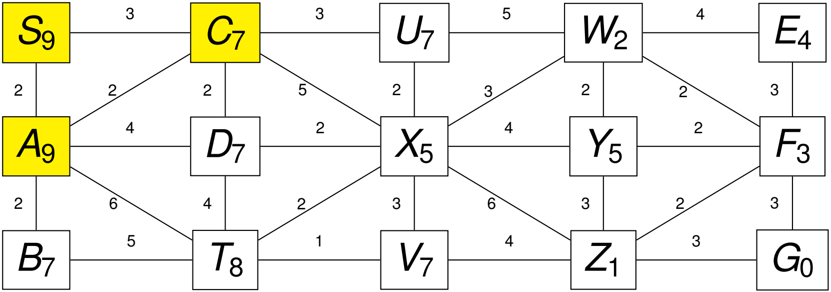
- Visited nodes: S
- Candidate nodes: A,C

Q1 (BFS)



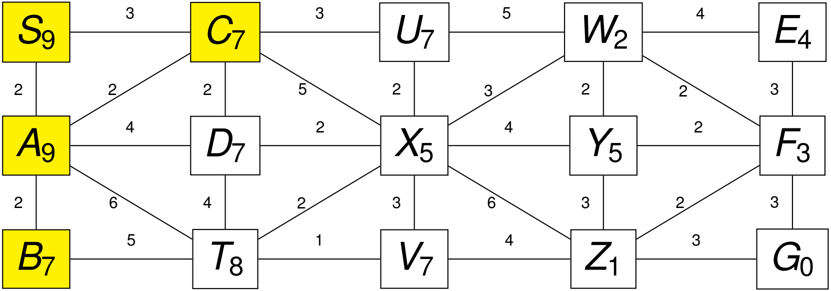
- Visited nodes: S,A
- Candidate nodes: C,B,D,T

Q1 (BFS)



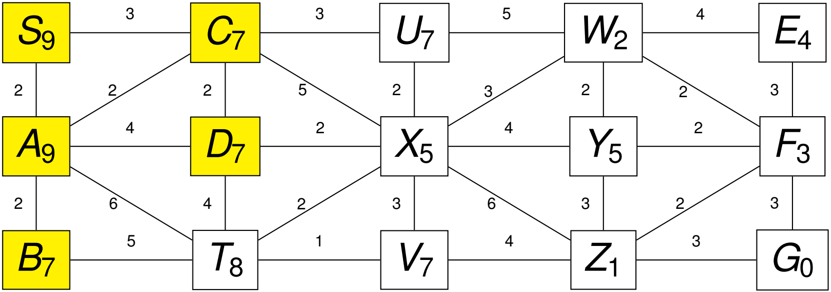
- Visited nodes: S,A,C
- Candidate nodes: B,D,T,D,U,X

Q1 (BFS)



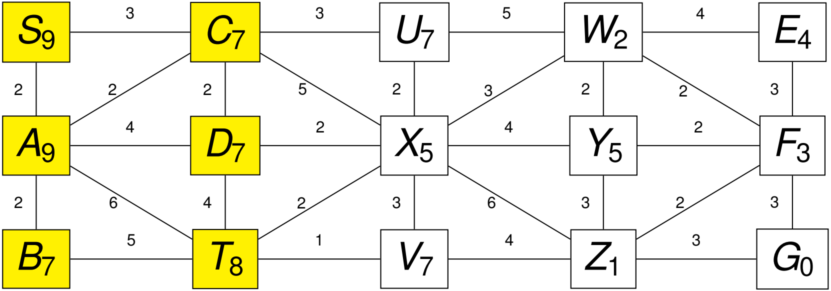
- Visited nodes: S,A,C,B
- Candidate nodes: D,T,D,U,X,T

Q1 (BFS)



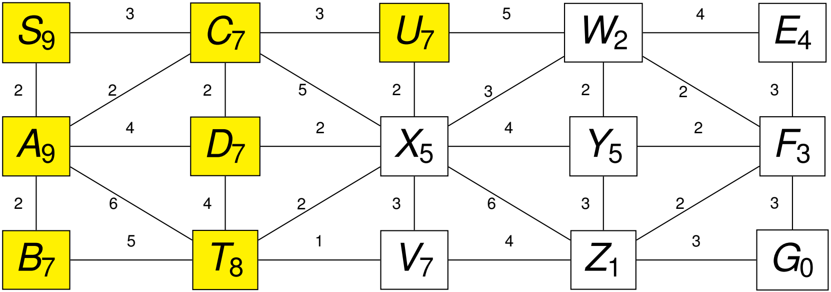
- Visited nodes: S,A,C,B,D
- Candidate nodes: T,D,U,X,T,T,X

Q1 (BFS)



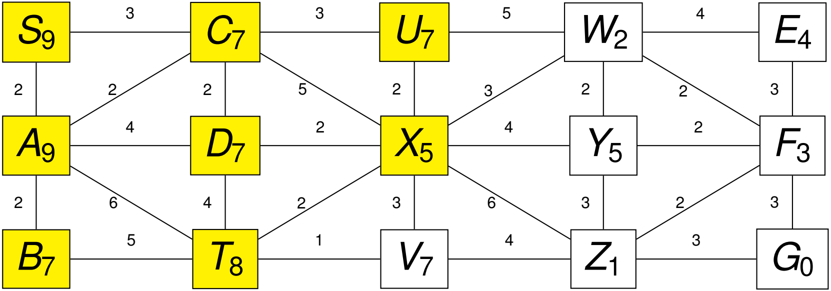
- Visited nodes: S,A,C,B,D,T
- Candidate nodes: D,U,X,T,T,X,V,X

Q1 (BFS)



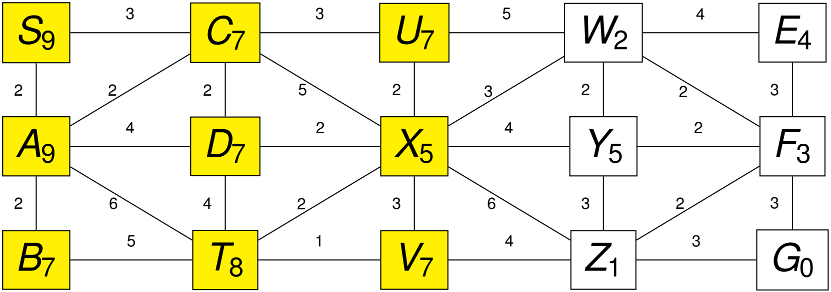
- Visited nodes: S,A,C,B,D,T,U
- Candidate nodes: X,T,T,X,V,X,W,X

Q1 (BFS)



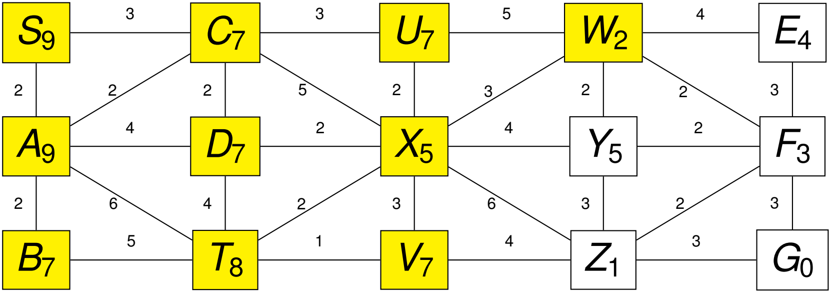
- Visited nodes: S,A,C,B,D,T,U,X
- Candidate nodes: T,T,X,V,X,W,X,V,W,Y,Z

Q1 (BFS)



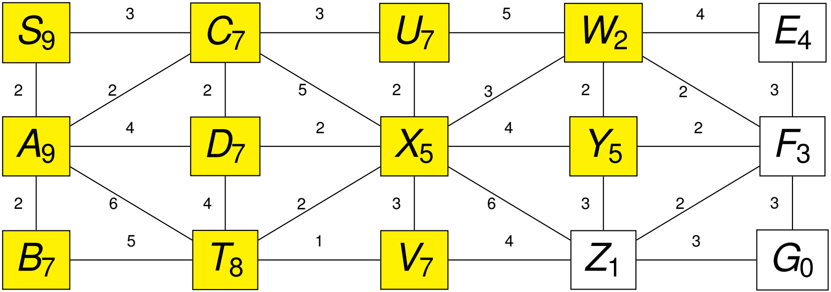
- Visited nodes: S,A,C,B,D,T,U,X,V
- Candidate nodes: X,W,X,V,W,Y,Z,Z

Q1 (BFS)



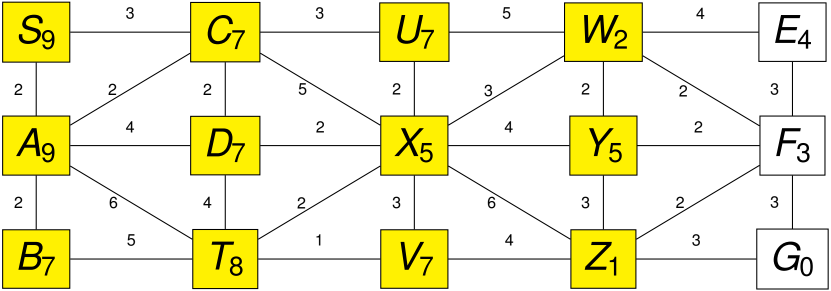
- Visited nodes: S,A,C,B,D,T,U,X,V,W
- Candidate nodes: X,V,W,Y,Z,Z,E,F,Y

Q1 (BFS)



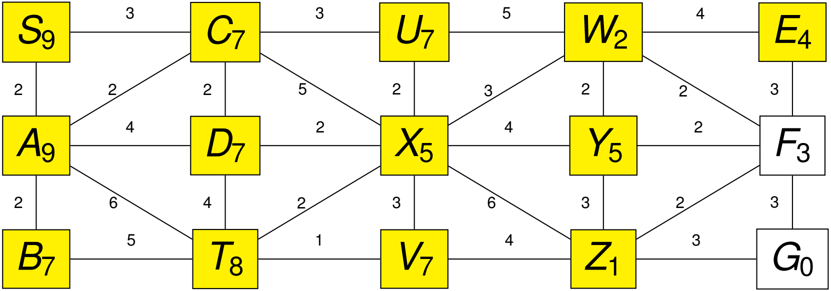
- Visited nodes: S,A,C,B,D,T,U,X,V,W,Y
- Candidate nodes: Z,Z,E,F,Y,F,Z

Q1 (BFS)



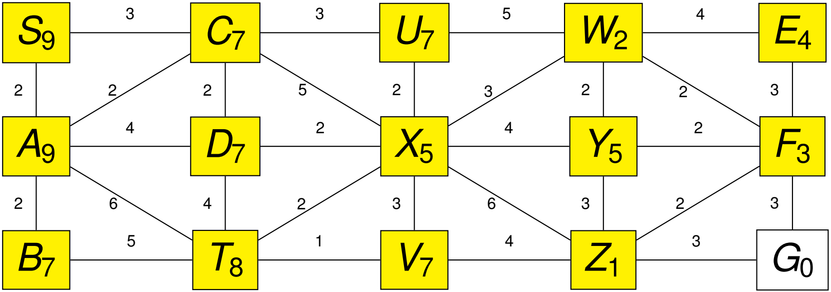
- Visited nodes: S,A,C,B,D,T,U,X,V,W,Y,Z
- Candidate nodes: Z,E,F,Y,F,Z,F,G

Q1 (BFS)



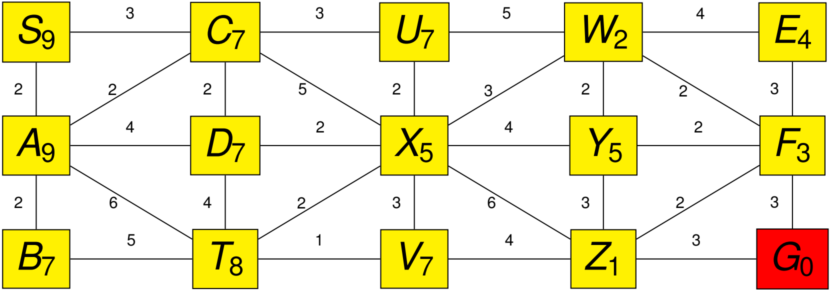
- Visited nodes: S,A,C,B,D,T,U,X,V,W,Y,Z,E
- Candidate nodes: F,Y,F,Z,F,G,F

Q1 (BFS)



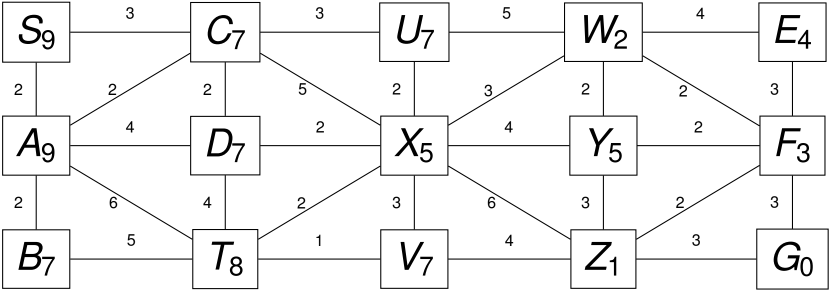
- Visited nodes: S,A,C,B,D,T,U,X,V,W,Y,Z,E,F
- Candidate nodes: Y,F,Z,F,G,F,G

Q1 (BFS)



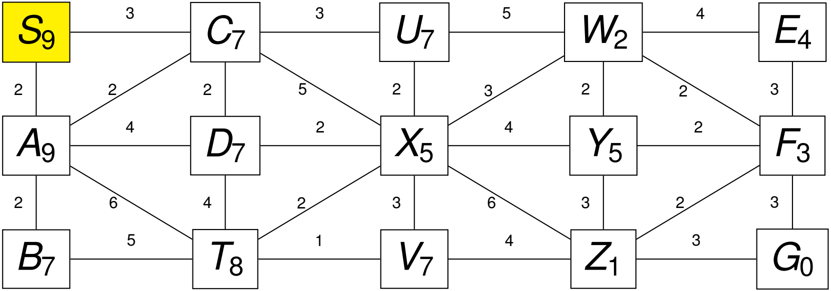
- Visited nodes: S,A,C,B,D,T,U,X,V,W,Y,Z,E,F,G
- Candidate nodes:

Q1 (Greedy)



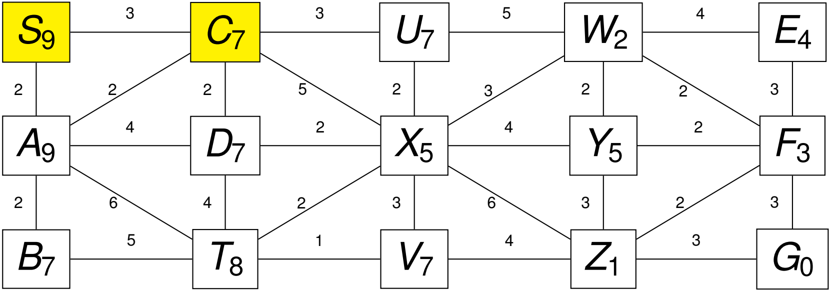
- Visited nodes:
- Candidate nodes: S(9)

Q1 (Greedy)



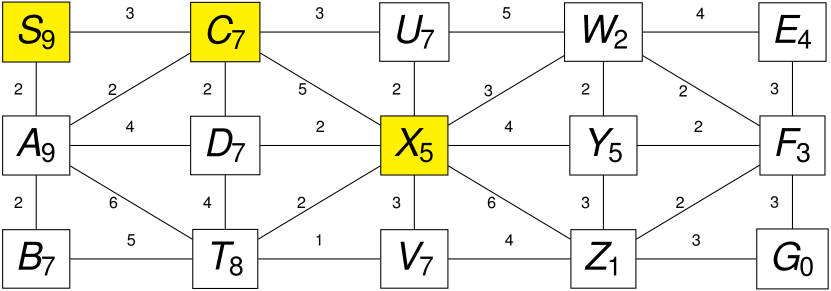
- Visited nodes: S
- Candidate nodes: A(9),C(7)

Q1 (Greedy)



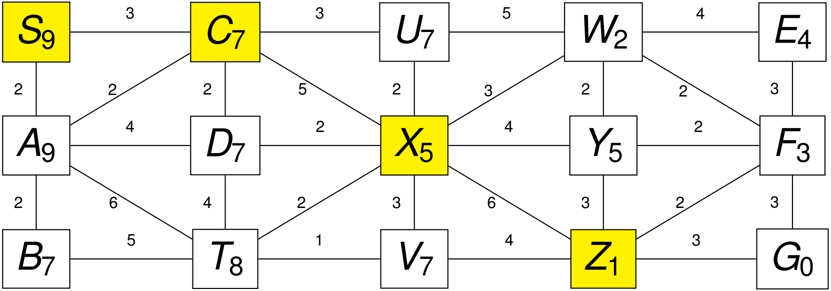
- Visited nodes: S,C
- Candidate nodes: A(9),D(7),X(5),U(7),A(9)

Q1 (Greedy)



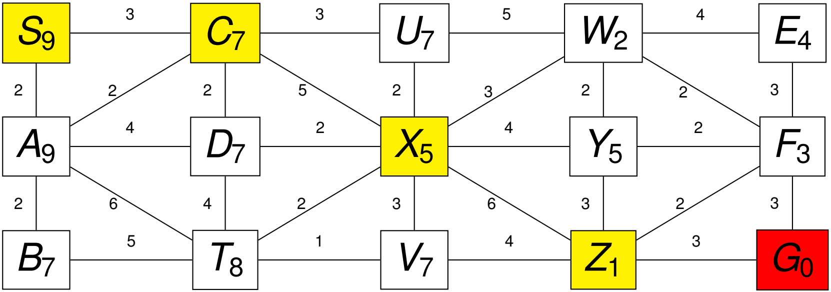
- Visited nodes: S,C,X
- Candidate nodes:
  A(9),D(7),U(7),A(9),W(2),T(8),U(7),D(7),T(8),
  Z(1),Y(5)

Q1 (Greedy)



- Visited nodes: S,C,X,Z
- Candidate nodes: A(9),D(7),U(7),A(9),W(2),T(8),U(7),D(7),T(8), Y(5),V(7),G(0),F(3),Y(5)

Q1 (Greedy)

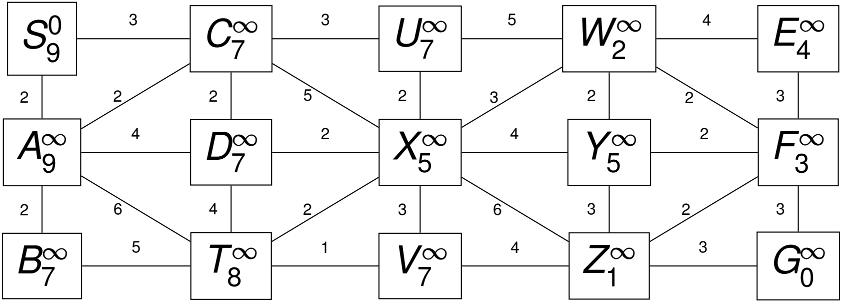


- Visited nodes: S,C,X,Z,G
- Candidate nodes: A(9),D(7),U(7),A(9),W(2),T(8),U(7),D(7),T(8), Y(5),V(7),F(3),Y(5)

Q1 (UCS)

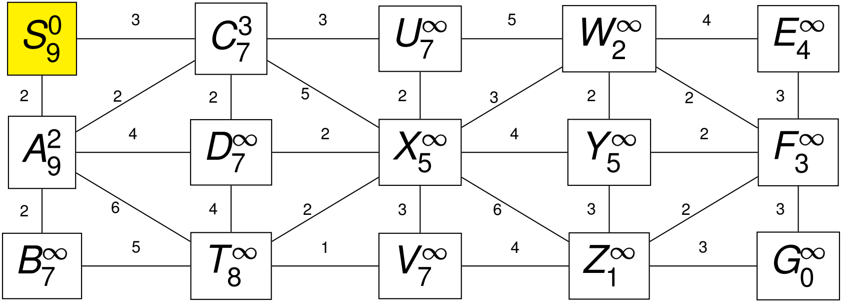
- We’ll skip UCS, it’s just a special case of A*, you can just set all h values to 0 and run A* algorithm.

Q1 (A*)



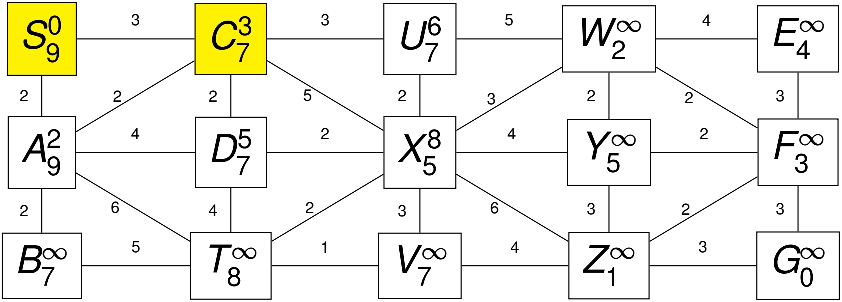
- Visited nodes:
- Candidate nodes: S(9)

Q1 (A*)



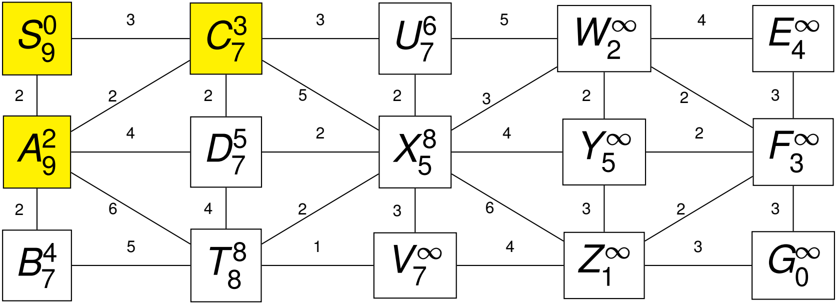
- Visited nodes: S
- Candidate nodes: A(11),C(10)

Q1 (A*)



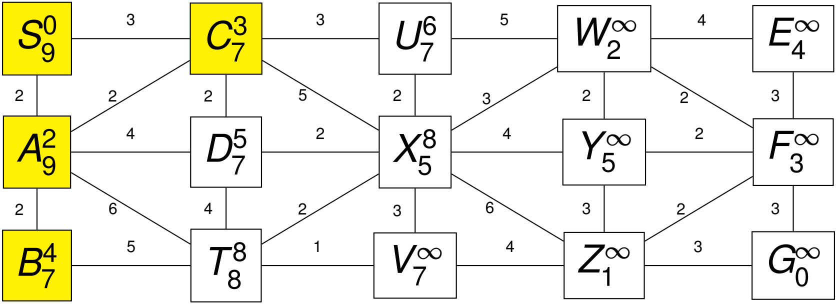
- Visited nodes: S,C
- Candidate nodes: A(11),D(12),X(13),U(13)

Q1 (A*)



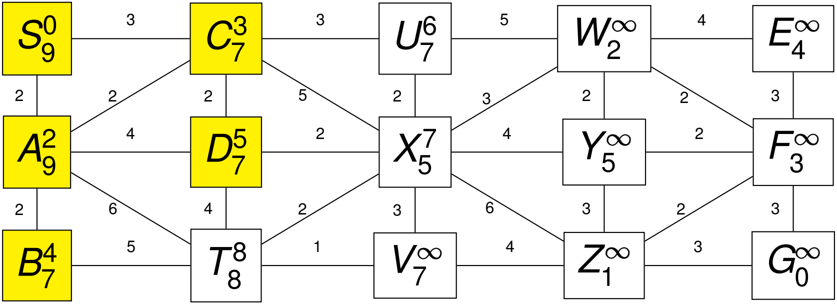
- Visited nodes: S,C,A
- Candidate nodes: D(12),X(13),U(13),B(11),T(16)

Q1 (A*)



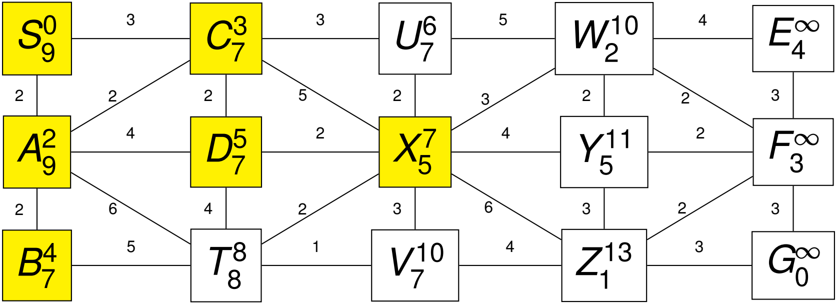
- Visited nodes: S,C,A,B
- Candidate nodes: D(12),X(13),U(13),T(16)

Q1 (A*)



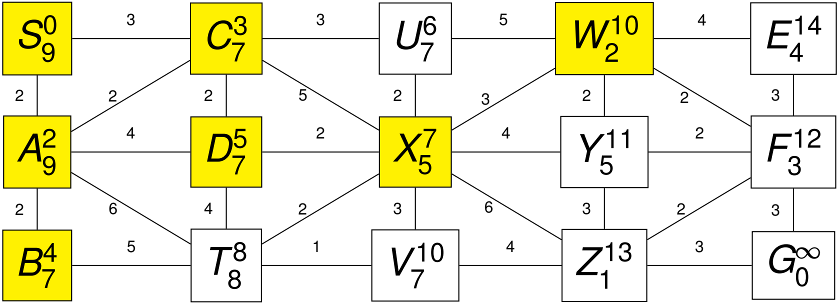
- Visited nodes: S,C,A,B,D
- Candidate nodes: **X(12)**,U(13),T(16),D(12)

Q1 (A*)



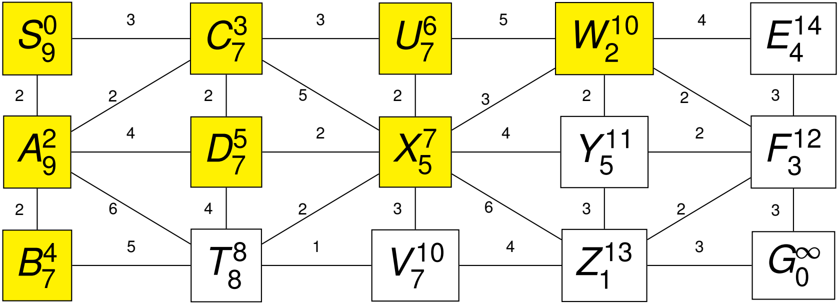
- Visited nodes: S,C,A,B,D,X
- Candidate nodes: U(13),T(16),T(16),V(17),Z(14) Y(16),W(12)

Q1 (A*)



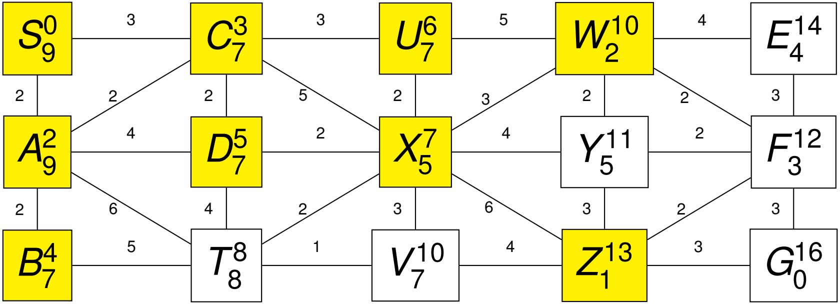
- Visited nodes: S,C,A,B,D,X,W
- Candidate nodes: U(13),T(16),V(17),Z(14),Y(16),F(15),E(18)

Q1 (A*)



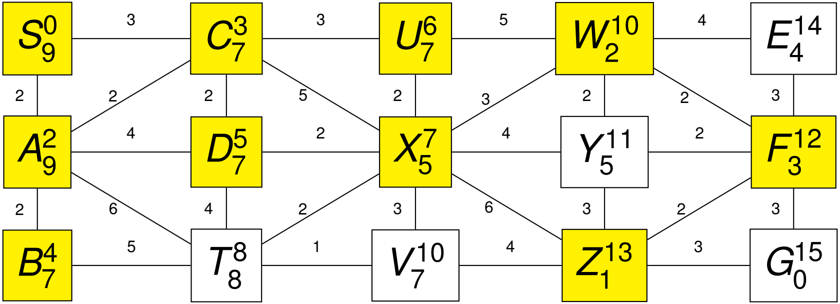
- Visited nodes: S,C,A,B,D,X,W,U
- Candidate nodes: T(16),V(17),Z(14),Y(16),F(15),E(18)

Q1 (A*)



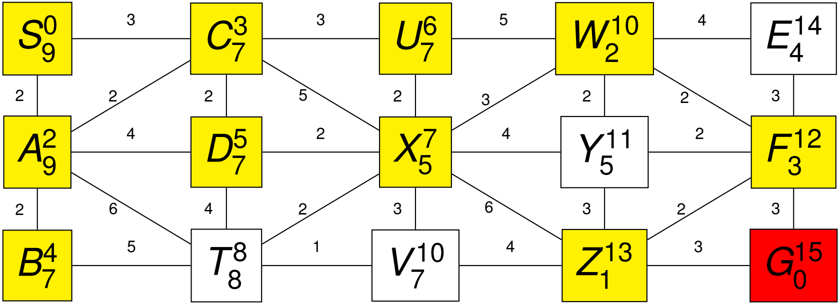
- Visited nodes: S,C,A,B,D,X,W,U,Z
- Candidate nodes: T(16),V(17),Y(16),F(15),E(18),G(16)

Q1 (A*)



- Visited nodes: S,C,A,B,D,X,W,U,Z,F
- Candidate nodes: T(16),V(17),Y(16),E(18),**G(15)**

Q1 (A*)



- Visited nodes: S,C,A,B,D,X,W,U,Z,F,G
- Candidate nodes: T(16),V(17),Y(16),E(18)

Q2

- Consider the following arrangement of tiles in the 8-puzzle:

1	2	3
8	5	
4	7	6

- Use A* to show how to arrive at the goal

1	2	3
4	5	6
7	8	

- Use total Manhattan Distance heuristic
- $\sum_{i=1..8}(|x[i]-i/3|+|y[i]-i\%3|)$, $x[i]$ is the row of the $i-th$ tile, $y[i]$ is the column of the $i-th$ tile,