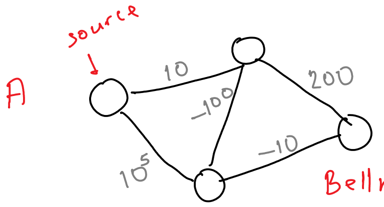
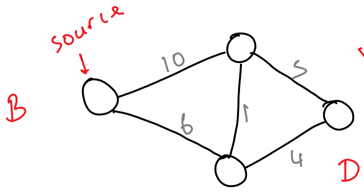
Week 06: Advanced BFS

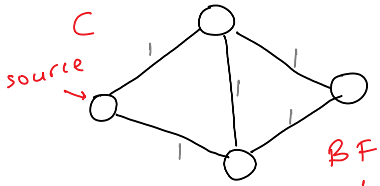
- When should we use BFS for shortest paths problem?
  (single source)

A


Bellman-Ford
↓ $O(N^2)$
vector

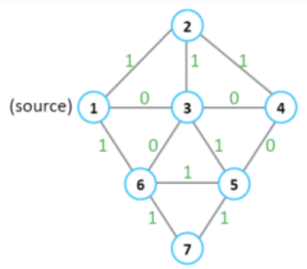
B


Dijkstra ($E \log V$)
↓
priority Queue

C


BFS $O(V+E)$
↓
Queue

0-1 BFS

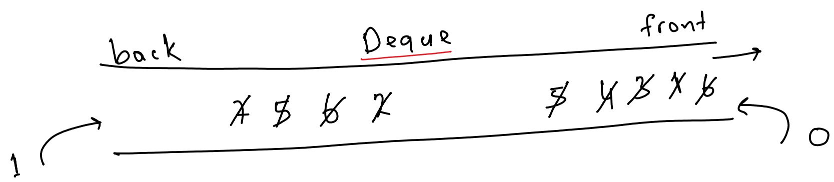


	1	2	3	4	5	6	7
distances	0	1	0	0	1	1	2

* In BFS, at the first time we visit a node, we use the shortest path.

source ↓

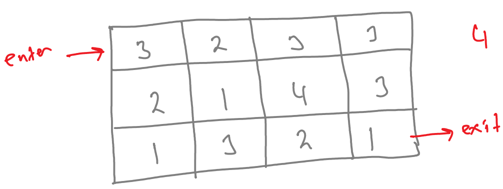
	1	2	3	4	5	6	7
distances	~~∞~~ 0	~~∞~~ 1	~~∞~~ 0	~~∞~~ 0	~~∞~~ ~~1~~ 0	~~∞~~ ~~1~~ 0	~~∞~~ 1

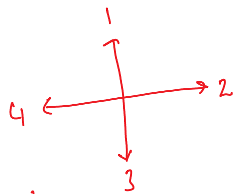


* Instead of a Deque you can use two separate queue structures.

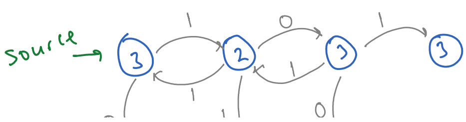
Exercise 1: Matrix

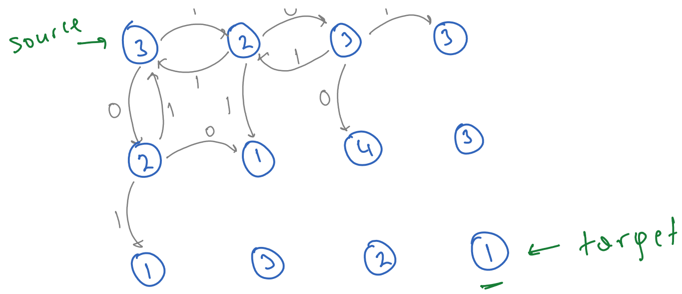
Sample Input	Sample Output
3 4	1
3 2 3 3	
2 1 4 3	
1 3 2 1	





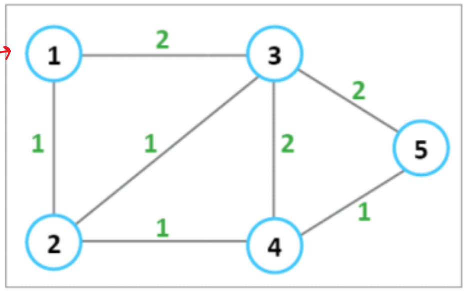
⇓ 0-1 Graph.





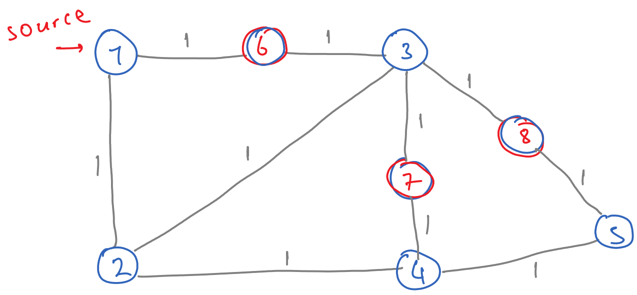
BFS with Splitting Edges

source

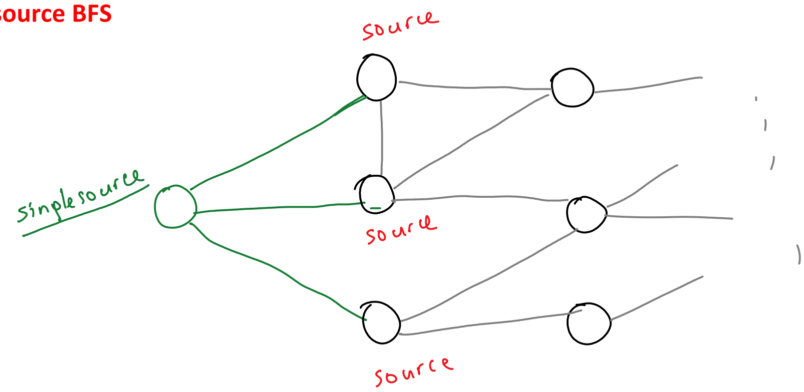


Sol 1: 0-1 BFS

Sol 2: Splitting edges + BFS



Multisource BFS



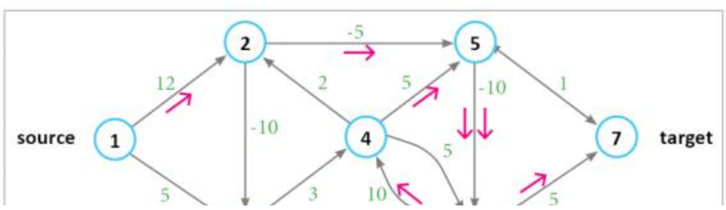
1) Add all sources into the queue and go on.

2) Create a new source (artificial node) and connect it to other sources.
Single-source shortest path.

Minimum Cost Path with K Edges on a Weighted Graph

Given a weighted, directed graph G(V, E), find the minimum cost path from a given source to a target node with exactly k edges on the path.

K = 7

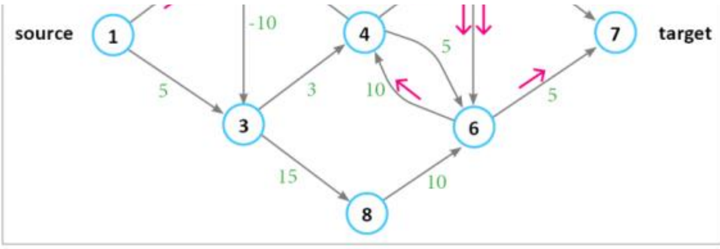


~~Dijkstra~~ negative weights

~~Belman-Ford~~ Does not consider # of edges on the path

~~Floyd-Warshal~~ //

BFS → standard BFS cannot be used.



BFS → standard BFS cannot be used.

- Modified BFS.

- How can we modify standard BFS implementation to solve this problem?

How to hadle # of edges on a path?

back	Queue	front
	(3,2) (5,2) (3,1) (2,1) (1,0)	

1 → node ID, 0 → length

	1	2	3	4	5	6	7	8
distances	~~∞~~ 0	∞	∞	∞	∞	∞	∞	∞

- For every node we need to memorize K different distances

$O(K(V+E))$

distances ↓ K	1	2	3	4	5	6	7	8 → N
0	0							
1		12	5					
2			2		7			
3								
4								
5								
6								
7								

Question: Implement the K-edges min cost path problem with DP tabulation technique (without queue).

BFS on a Complementary Graph

Calculate all shortest distances from the source node 1 to all
other nodes on the complementary graph.

Graph G (V, E) Complement of G, H (V, E)

source ↓	1	2	3	4	5
distances	0	2	2	1	1

Sol 1: Use BFS algorithm directly on the complementary graph.

$O(V + E')$, $E' \rightarrow n \cdot (n-1) - E$

Sol 2: Store input graph edges in a set, use BFS.

Sol 2: Store input graph edges in a set, use BFS.

↓

for every node we have to test all nodes

$O(V(V+\lg E))$

$O(V+\underbrace{E}_{})$

↓

$\frac{n.(n-1)}{2} - E$

* # of edges in the complementary graph can be a big number.

* Creating the complementary graph is $O(N^2)$

Sol 3: Two sets of nodes.

adjacentSet = { }

notAdjacentSet = {~~1~~, ~~2~~, ~~3~~, ~~4~~, ~~5~~}

source

1	2	3	4	5
0	2	2	1	1

Queue

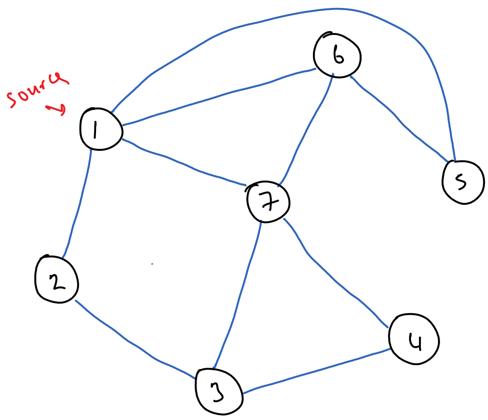
→ ~~3~~ ~~2~~ ~~5~~ ~~4~~ ~~1~~ →

while (!q.empty() && !notAdjacentSet.empty())

{

⋮

$O(V+E)$



tempset

~~2~~ ~~7~~ ~~5~~ ~~6~~

~~2~~ ~~7~~ ~~4~~

~~7~~ ~~6~~ ~~7~~

7

nodesSet

~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~

~~2~~ ~~5~~ ~~6~~ ~~7~~

~~2~~ ~~4~~ ~~6~~ ~~7~~

~~7~~

~~7~~

Queue

~~2~~ ~~6~~ ~~5~~ ~~4~~ ~~3~~ ~~7~~

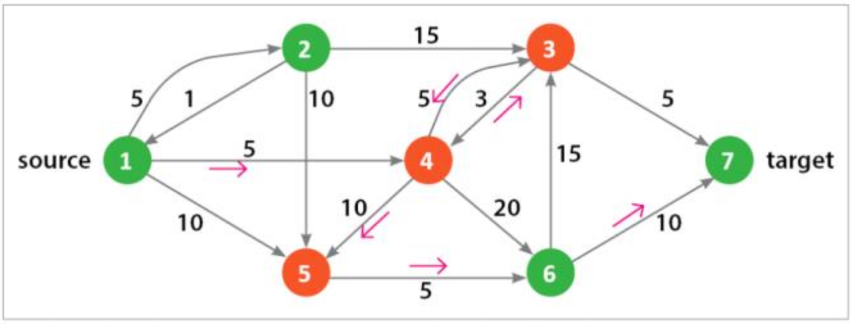
	1	2	3	4	5	6	7
distances	0	3	1	1	2	2	3

BFS with Bit Masking

Given a weighted, directed graph G (V, E), and a set X of vertices.

Find the Minimum Cost Path passing through all the vertices of the set X, from a given source vertex S to a target vertex T. The size of X is K. Source and target nodes are not member of X.

- We have to store the distance for every node, for all different sets of nodes on the path.



map <int, set<int>>

node ID

set of nodes

for one node

1 → 4 → 6   25 ← (6, {4})

1 → 5 → 6   15 ← (6, {5})

1 → 4 → 5 → 6   35 ← (6, {4, 5})

1 → 2 → 5 → 6   20 ← (6, {5})

vector <int, set<int>> distances $O(NK\log N)$

use bit masking instead of set.

vector <int, int> distances   → $O(NK)$

5	4	3
0	0	0
1	0	1