Dijkstra's Algorithm

To Find:- Single-Source Shortest Paths using Greedy Method

e.g. to plan shortest paths for travelling!

$G = (V, E)$ // digraph ⇒ one-way streets

v_0 = source

c = weighting function // weights of edges

v_0 to all remaining vertices ?

Shortest Path? (Greedy Technique)

⇒ Shortest Path to the nearest vertex is generated. Then a shortest path to the second nearest vertex is generated & so on!

$O((n+E)\log n)$ — $V-S$ — Red Black Tree

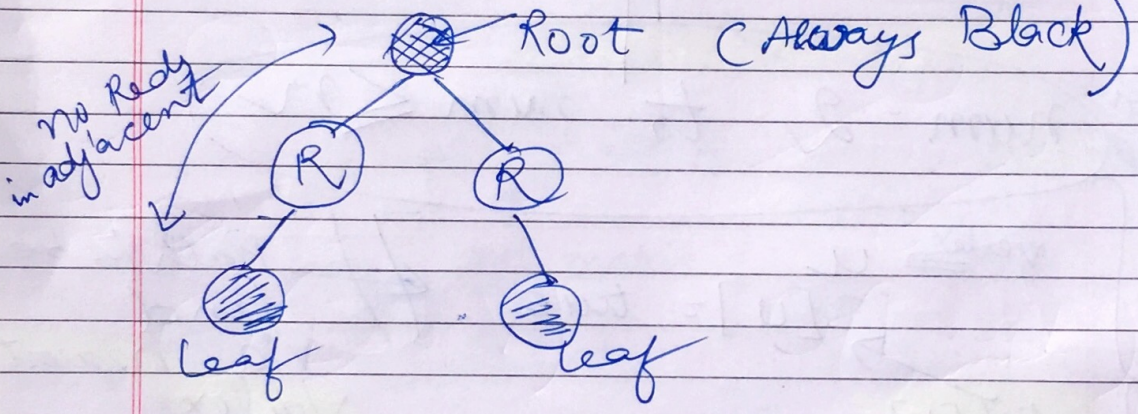
Or $O((V+E)\log V)$ — nearest vertex — examine edge — update dist

Red Black Tree

→ self-balancing binary tree
↳ binary search tree
BST

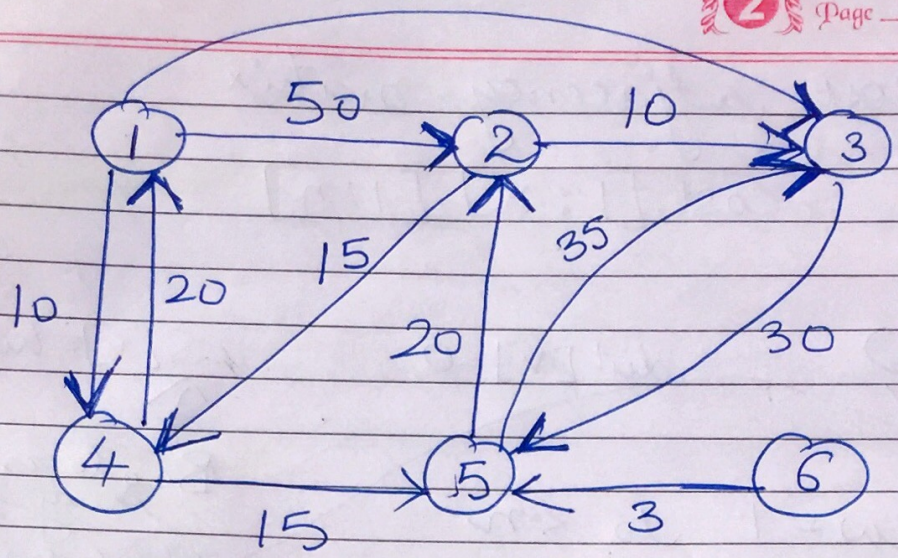
⇒ extra bit → to store color (Red or Black)

⟹ O(log n) search time, insert, delete



⟹ Reduces rotations than AVL trees

45



Dijkstra's Algorithm (Greedy Technique)

Single-source-shortest paths

PATH

Source 1

1, 4

LENGTH

10

shortest path (nearest node)

1, 4, 5

10 + 15 = 25

1, 4, 5, 2

50

× direct edge

or ⇒ 10 + 15 + 20

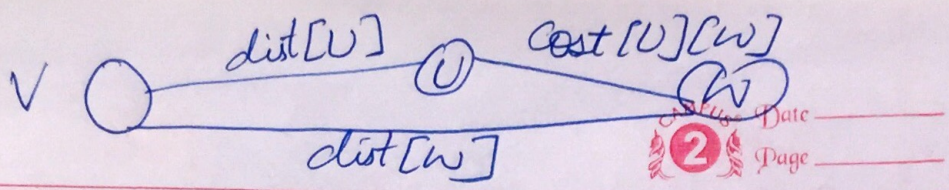
⇒ 45

45

1, 3

45

or 1, 4, 5, 2, 3 → 10 + 15 + 20 + 10 = 55



cost adjacency matrix
⇒ cost [1:n][1:n]

direct edge

dist[i] = cost[v][i]
S[i] = false
for i = 1 to n

Boolean Array

v ○ Source

S[v] = true
dist[v] = 0

for num = 2 to num ≤ n

vertex ⇒ u
S[u] = true

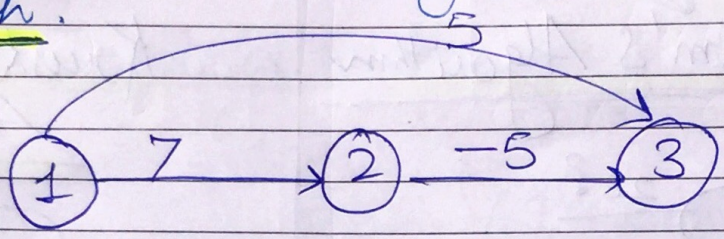
// Path to be find via nearest vertex

dist[u] minimum

for (w ⇒ 1 to n)
if (S[w] == false) &&
dist[w] > dist[u] + cost[u][w]
dist[w] = dist[u] + cost[u][w]

Limitation of Dijkstra's Algorithm

⇒ It does not necessarily give the correct results on the digraph [directed graph] when some or all the edges have negative length.



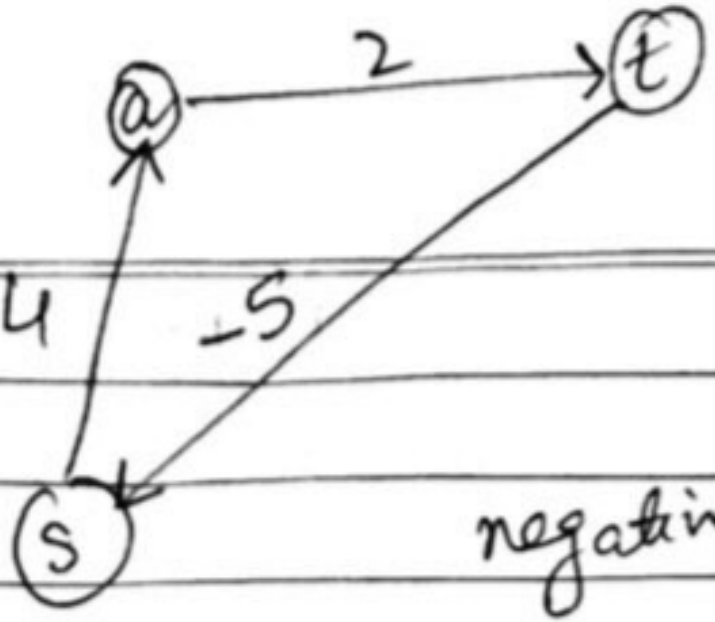
dist[2] = 7
dist[3] = 5
} by Dijkstra's Algorithm (nearest vertex)

dist[3] = 2 || Actual

SOLUTION is :- to use another algorithm

BELLMAN & FORD ALGORITHM

(using dynamic programming)



negative edge?

Chemistry problem

$s \to a$ conversion ⇒ produce heat 4KJ

$a \to t$ conversion ⇒ produce heat 2KJ

but You need 5KJ to get back state s from state t

Single-Source Shortest Path Problem

Greedy method

Dijkstra's Algorithm

(does not work on negative edges)

dynamic programming

Bellman & Ford Algorithm

(works for graph involving negative edges)

DYNAMIC PROGRAMMING

- It is an algorithm design method that can be used when the solution to a problem can be viewed as the result of a sequence of decisions.

PRINCIPLE OF OPTIMALITY

The principle of optimality states that an optimal sequence of decisions has the property that:—

whatever the initial state & decision are, the remaining decisions must constitute an optimal decision sequence with regard to the state resulting from the first decision.

Optimal decision sequence {
- पिछले optimal Decision का
- असर अगले Decision पर!

GREEDY METHOD VS DYNAMIC PROGRAMMING

In greedy method only one decision sequence is ever generated!

It takes decision & move forward & does not reconsider any thing

In dynamic Programming, many decision sequences may be generated!

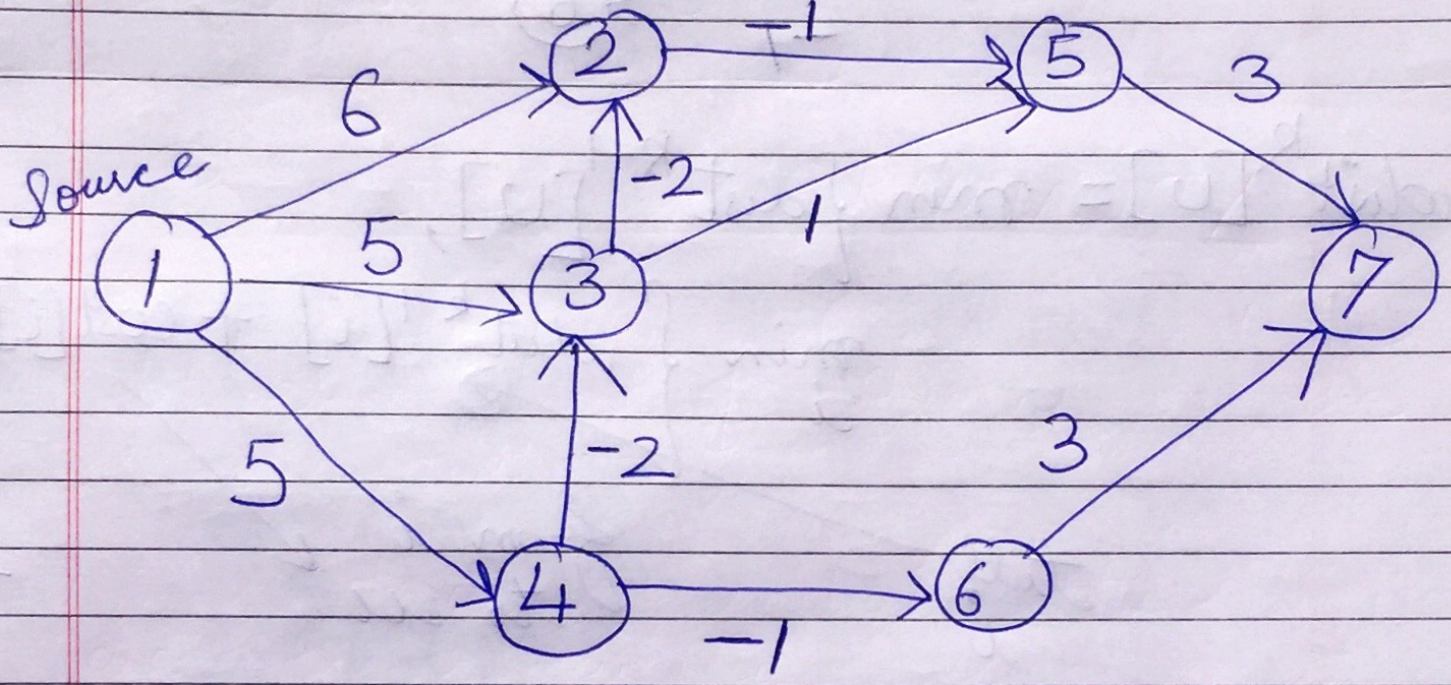
decisions are based on intermediate decisions

It first have an idea how to take decision & takes final decision slowly after studying various factors ⟹ DYNAMIC!

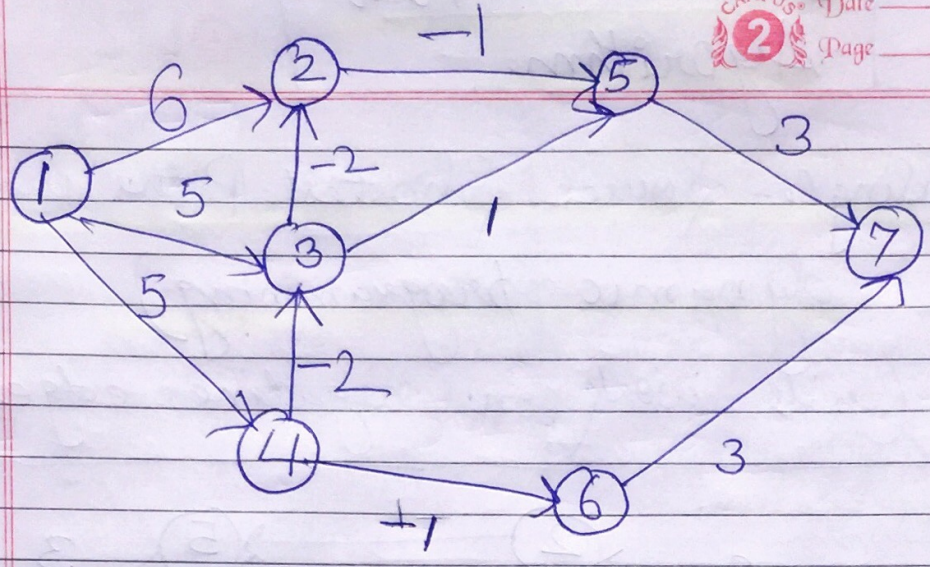
Bellman & Ford Algorithm

(Single-Source Shortest Paths using dynamic programming)

(will work on negative edges)



$$dist^{k}[v] = \min\{ dist^{k-1}[v], \min_{i}\{ dist^{k-1}[i] + cost[i][v] \} \}$$



$dist^k[v] = \min\{dist^{k-1}[v],$

$\min_i \{dist^{k-1}[i] + cost[i][v]\}$

till i — from i to u

$dist^1[2]=6$, $dist^1[3]=5$, $dist^1[4]=5$

$dist^2[2] = \min\{dist^1[2], \min_i dist^1[i] + cost[i][2]\}$

$= \min\{6,$

$i=1 \Rightarrow dist^1[1] = 0 + 6$

$i=3 \Rightarrow dist^1[3] = 5 + (-2)$

$i=4 \Rightarrow dist^1[4] = 5 + \infty$

$i=5 \Rightarrow dist^1[5] = \infty + \infty$

$i=6 \Rightarrow dist^1[6] = \infty + \infty$

$i=7 \Rightarrow dist^1[7] = \infty + \infty$

6

3

3

from (1) source

to

$dist^k$

R / alternative	1	2	3	4	5	6	7
1	0	6	5	5	∞	∞	∞
2	0	3	3	5	5	4	∞
3	0	1	3	5	2	4	7
4	0	1	3	5	0	4	5
5	0	1	3	5	0	4	3
6	0	1	3	5	0	4	3

$dist^1[1] = 0$

$dist^1[2] = 6$

$dist^2[2] = 34$

$$dist^3[2] = \min\{dist^2[2],$$

$$= dist^2[1] + cost[1][2] \quad 0 + 6 = 6 \quad = 6$$

$$dist^2[2] + cost[2][2] \quad 3 + 0 \quad = 3$$

$$dist^2[3] + cost[3][2] \quad 3 + (-2) \quad = 1$$

via 4

$$dist^2[4] + cost[4][2] \quad 5 + \infty \quad = \infty$$

$$dist^2[5] + cost[5][2] \quad 5 + \infty \quad = \infty$$

$$dist^2[6] + cost[6][2] \quad 4 + \infty \quad = \infty$$

$$dist^2[7] + cost[7][2] \quad \infty + \infty \quad = \infty$$

$$\}$$

$$= \min\{3, 1\}$$

$$= 1$$

Time Complexity of Bellman & Ford Algorithm

$O(n^3)$ or $O(V^3)$

(if adjacency matrix is used)

$O(ne)$ or $O(VE)$

(if adjacency list is used)

Greedy Techniques!
Dynamic Programming

CAMPUS 2 Date Page

Problem	Algorithm	Technique	Time Complexity O
Knapsack Problem	-	Greedy Method	$O(n)$
Minimum cost Spanning Tree	(a) Prim's Algorithm	Greedy Method	$O(E \log V)$
	(b) Kruskal's Algorithm	Greedy Method (binary heap is used)	$O(E \log E)$
Single-Source Shortest Paths Problem	(a) Dijkstra's Algorithm	Greedy Method $[O(V+E)\log V]$ (red black tree is used)	
	(b) Bellman & Ford Algorithm	DYNAMIC PROGRAMMING $O(VE)$ (adjacency list is used)	
		or $O(V^3)$ (adjacency matrix is used)	