Sorting (Sunday)

- Inversion
- Bubble sort
- Insertion Sort
- Lower bound on comparison based sorting
- Counting Sort (Non comparison)
- Radix sort

Apply → Dynamic programming

- All pair shortest path
- Bellman ford Algorithm
- 2 HW
- Optimal Binary Search tree
  Sunday – morning

Data structure

- Friday – Saturday
- Graph
  - Traversal
    - DFS
    - BFS

Directed as well as undirected

Application of DFS
- paranthesization Theorem
- Classification of edges

- topological sort
- Strongly connected component
- Biconnected component

All Pair Shortest Path

GATE 2004 (IT)

Data structure Heap - Basic Introduction

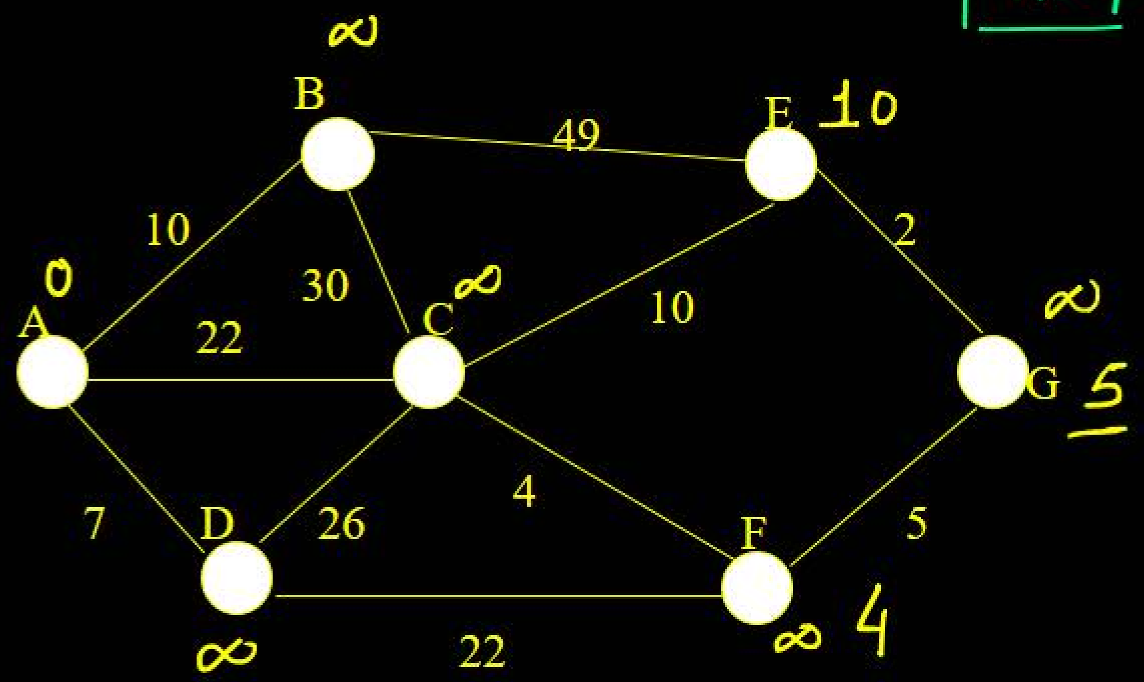
$\boxed{\text{Heap}}$ - solve

- CBT
- Insertion
- Deletion
- Adjust

operation, complexity

- Algorithm
- Build Heap
- Heapsort

Consider the undirected graph below:



(A) (E, G), (C, F), (F, G), (A, D), (A, B), (A, C)

(B) (A, D), (A, B), (A, C), (C, F), $\boxed{\text{(G, E)}}$, (F, G) ✗

(C) (A, B), (A, D), (D, F), (F, G), (G, E), (F, C)

(D) (A, D), (A, B), (D, F), (F, C), (F, G), (G, E) ✓

A	B	C	D	E	G	F
0	∞	∞	∞	∞	∞	∞
–	10	22	7	∞	∞	∞
–	10	22	–	∞	∞	22
–	–	22	–	49	∞	22

GATE 2004 (IT)

Using Prim's algorithm to construct a minimum spanning tree starting with node A, which one of the following sequences of edges represents a possible order in which the edges would be added to construct the minimum spanning tree?

(A) (E, G), (C, F), (F, G), (A, D), (A, B), (A, C)

(B) (A, D), (A, B), (A, C), (C, F), (G, E), (F, G)

(C) (A, B), (A, D), (D, F), (F, G), (G, E), (F, C)

(D) (A, D), (A, B), (D, F), (F, C), (F, G), (G, E)

Single Source shortest path for every vertices then

$(V+E)\log V$

No. of vertices

$V \cdot (V+E)\log V$

$V^2 \log V + VE\log V$

$\simeq O(V^3 \log V)$ (from dense graph)

$E = \frac{(V(V-1))}{2}$

Maximum value

$V \cdot E = V^3$

$E = \frac{V(V-1)}{2}$

$E = V-1$

Minimum

- Given a weighted digraph $G = (V,E)$ with weight function $w : E \rightarrow$ R, *(R* is the set of real numbers), determine the length of the shortest path (i.e., distance) between all pairs of vertices in G. Here we assume that there are no cycles with zero or negative cost.

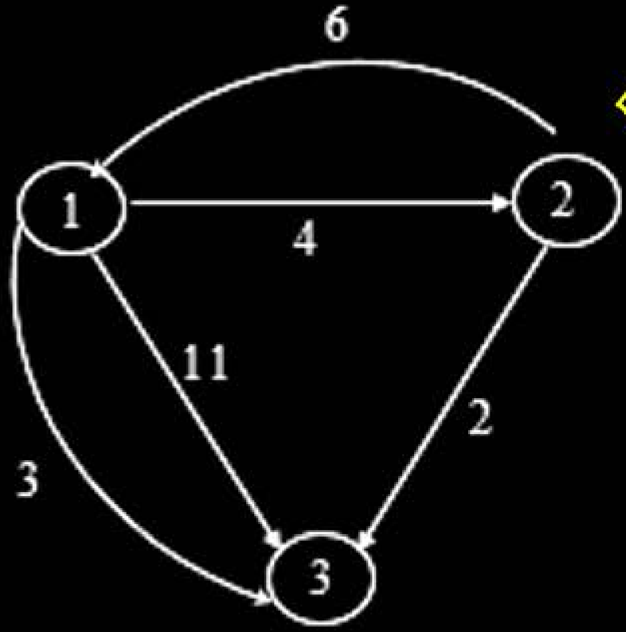
First Approach

Running dijktra's algorithm for each & every vertex.

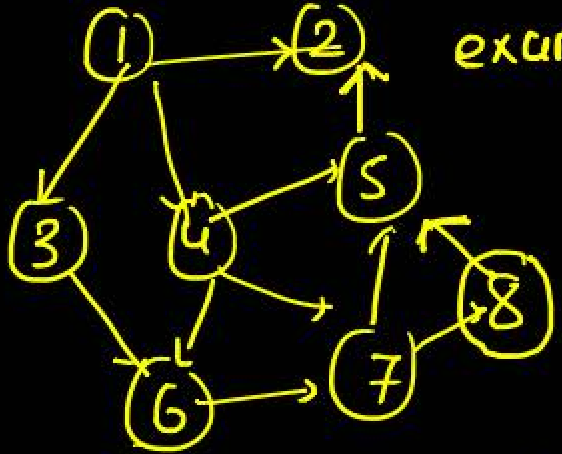
First Approach

- If there are no negative costs edges apply Dijkstra's algorithm to each vertex (as the source) of the digraph.
- Recall that Dijkstra's algorithm runs in $O((|V|+|E|) \log V)$.
- This gives a $O(|V|(|V| + |E|) \log V)$
- $= O(|V|^2 \log V + |V||E| \log V)$ time algorithm,
- If the digraph is dense i.e. complete graph($E = \frac{V(V-1)}{2}$) , this is an $O(|V|^3 \log V)$algorithm.

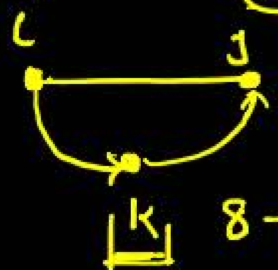
All Pair Shortest Path



(a) Example diagraph



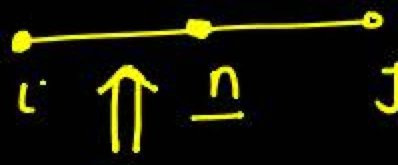
example graph



8 - n

Many intermediate vertex can occur or it may Not occure.

H.W



(i, n) (n, j)?

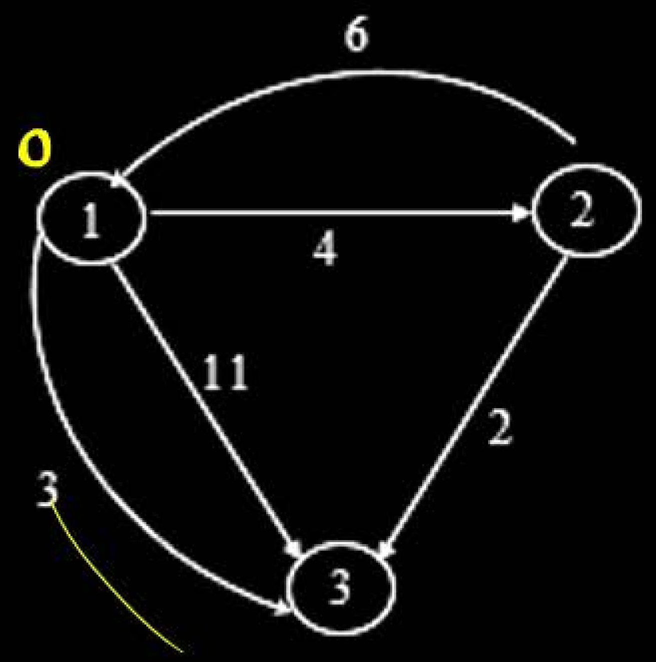
If I take ~~pair~~ pair of vertice i, j

General

i → j shortest path

maximum index (1, 2, 3, 4 . . . 8) of ~~it~~ intermediate vertex can occure

All Pair Shortest Path



(a) Example diagraph

All pair shortest path : shortest path between every pair of vertices.

Any pair of vertices (i , j)

(i —— n —— j)
with highest index

Each vertex is labeled as 1 to n

(i —— n)
Largest ~~to~~ Index of intermediate

If shortest path going through (i — n) shortest (n - j) will also be the shortest between the pair

Principal of Optimality

- • Let $d_{ij}^{(k)}$ be the weight of a shortest path from vertex i to vertex j for which all intermediate vertices are in the set {1, 2, . . . , k}. Dynamic programming enumerates all possibilities.

i — n — j

i — n — j

1

$\boxed{n.1}$

$$d_{ij}^{(k)} = \min\{ d_{ij}^{k-1},\ d_{ik}^{k-1} + d_{kj}^{k-1} \}$$ Recursion

$d_{ij}^{k.2}$, $d_{ik}^{k.2} + d_{kj}^{k.2}$

d_{ij}^{0} ← Represent the cost of direct path from i to j

Principal of Optimality

Principal of Optimality

- We can regard the construction of a shortest i to j path as first requiring a decision as to which is the highest indexed intermediate vertex k.
- Using $A^k(i,j)$ to represent the length of a shortest path from i to j going through no vertex of index greater than k, we obtain
- $A^k(i,j) = \min\left\{\min_{1 \le k \le n} \{A^{k-1}(i,k) + A^{k-1}(k,j), cost(i,j)\}\right\}$

- Clearly, $A^0(i,j) = \mathrm{cost}(i,j)$, $1 \le i \le n$, $1 \le, j \le n$.

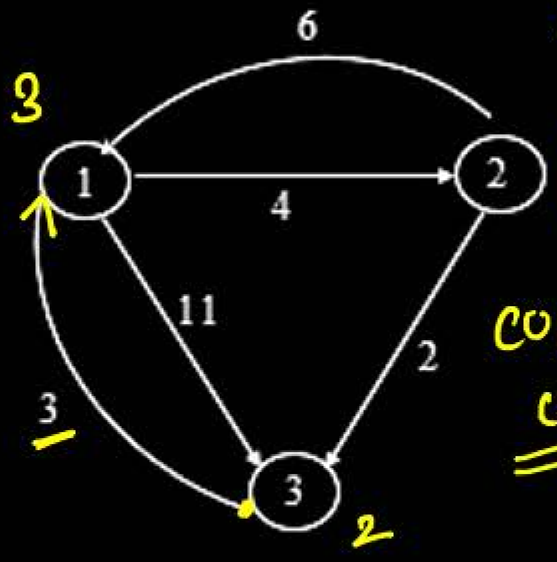
Bottom up tabulation

Smaller instance to Larger instance

top down approach is Recursion with Memorization

$d^3(i,j)$

Example



(a) Example diagraph

$\boxed{1\ 2}$ — $(1\ 1) + (1,2)$

$\underline{3\ 1}$ — $(3,1) + (1,1)$ $0 +$

Simplest

copy the cost

A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

$\uparrow$ A^1

If there is No direct edge ~~then~~ between (i,j) then $cost(i,j) = \infty$

$A^1 =$

$A^1(i,j) = \min\{A^0_{ij},\ A^0_{i1} + A^0_{1j}$

$A^1(2,3) = \min\{A^0(2,3)$ 2

$A^0(2,3) = \textcircled{2}$, $A^0(2,1) + A^0(1,3)$

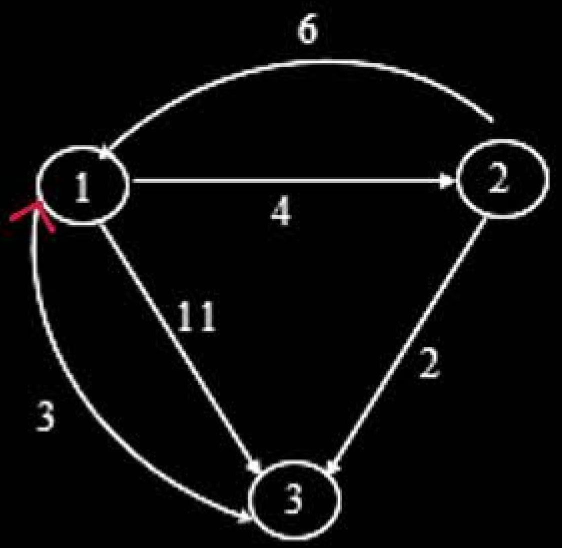
$6 + 11$

$A^1(3,2) = \min\{A^0(3,2)$ ∞

$A^0(3,1) + A^0(1,2)$

$3 + 4 = \textcircled{7}$

Example



(a) Example diagraph

k source

k- destination

A^1	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

Am clear?

Compute $A^1(1,3)$

$A^0(1,1) + A^0(1,3)$

$0 + A^0(1,3)$

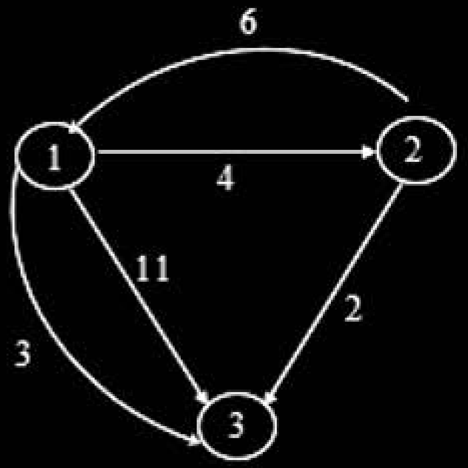
3 $A^2(3,1) = \min\{A^1(3,1), A^1(3,2) + A^1(2,1)$

$3 \qquad 7 + 6$

6 $A^2(1,3) = \min\{A^1(1,3), A^1(1,2) + A^1(2,3)$

$11 \qquad 4 + 2$

Example



(a) Example diagraph

A^2	1	2	3
1	0	4	6
2	$\boxed{6}$	0	2
3	3	7	0

Enumerating - No vertex can be skipped

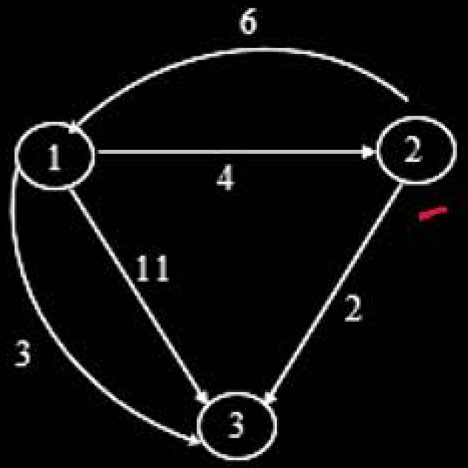
$$A^3(1,2) = \min\{\underbrace{A^2(1,2)}_{4},\ \underbrace{A^2(1,3)}_{6} + \underbrace{A^2(3,2)}_{7}\}$$

$= 4$

$$A^3(2,1) = \min\{\underbrace{A^2(2,1)}_{6},\ \underbrace{A^2(2,3) + A^2(3,1)}_{2+3}\}$$

5

Example

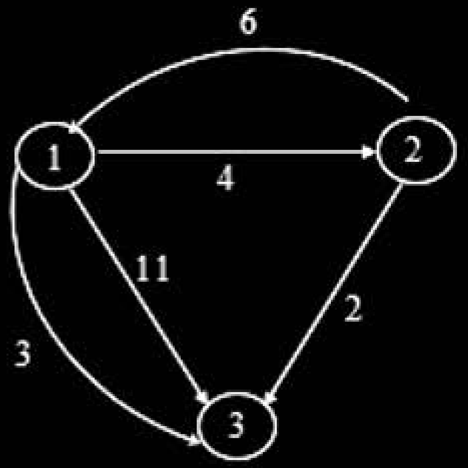


(a) Example diagraph

A^3	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0

final matrix
Representing (
Shortest path between
every pair of vertex

Example



(a) Example diagraph

A^0	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

A^3	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

Algorithm

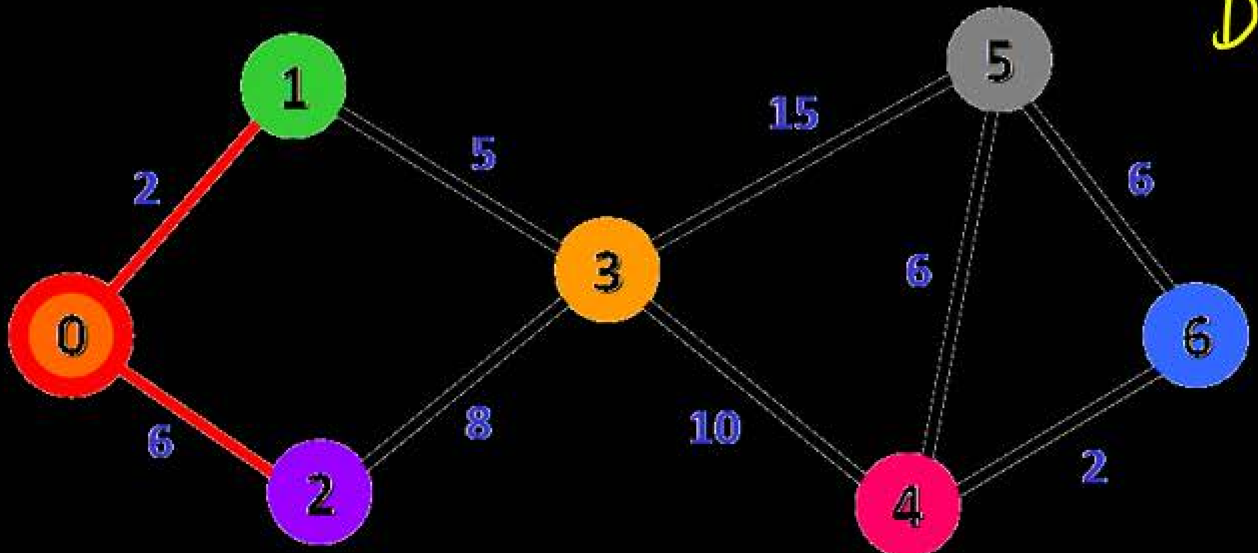
```
Algorithm AllPaths (cost, A, n)
      / / cost[1:n, 1:n] is the cost adjacency matrix of a graph with
      / / n vertices; A[i,j] is the cost of a shortest path from vertex
      / / i to vertex j. cost[i,i] = 0.0, for
      {
          for i : = 1 to n do
              for j: = 1 to n do
                  A[i,j] : = cost[i,j]; / / Copy cost into A.
          for k : = 1 to n do
              for i: = 1 to n do
                  for j: = 1 to n do
                          A[i,j] : = min(A[i,j], A[i,k] + A[k,j]);
      }
```

(All pair shortest path)

$A^0[i,j]$ - edge weight

$A^0 \leftarrow \boxed{A^1}$

A^2



Bellman ford Algorithm

- Similarly in argument with All pair shortest

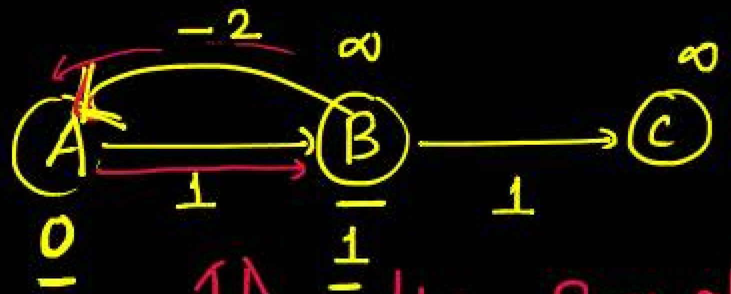
Dijktra's. Single Source shortest path

- May or may not work with Negative edge weight
- Negative edge weight cycle then does not give Right answer

Bellmanford Algorithm: Single source shortest path.

- It computes shortest path from source to all
- Reports Negative edgeweight cycle

Negative Edge weight cycle



$-2+1=-1$

the sum of the cost of cycle is Negative

Shortest path

• Shortest path does not exists

A	B	C
0	$\infty > 0+1 = 1$	$\infty > 1+1 = 2$

(II)

A	B	C
$0 > 1+(-2) =$	$1 > -1+1$	$2 > 0+1$
-1	$1 > 0 - 0$	1
-2	-1	0
-3	-2	-1
$\vdots$	$\vdots$	$\vdots$
∞	$-\infty$	$-\infty$

Negative Edge weight cycle

Negative Edge weight cycle

If a graph $G = (V, E)$ contains a negative-weight cycle, then some shortest paths may not exist.

Finds all shortest-path lengths from a source $s \in V$ to all $u \in V$ or determines that a negative-weight cycle exists.

Optimal Solution

Single Source shortest path

$|(n-1)|$

Shortest

s – i

$\boxed{1}$ edge

2 edge

3 edge

⋮

n-1 edge

$\boxed{S}$ —•—o—•—$\overset{u}{•}$— i

Length of the shortest path between source and a vertex i

n vertice

Length. No. of edges

- it can pass through n-2 intermediate vertex and length of this shortest path can n-1.
- Is it necessary that shortest path will n-1 ~~be~~ edges

$$s \overbrace{\text{—}\cdot\text{—}\cdot \cdots \cdot\text{—}}^{\boxed{n-2}} j \xrightarrow{1} i$$

$\boxed{n-1}$ edges $\quad$ cost (j, i)

If ~~shorsl~~ shortest path does not not have n-1 edges then it may or may not have shortest path (n-2) ~~ele~~ edges

$$\left(d^{n-1}[s,i] = \min \left\{ d^{n-2}[s,i],\ \underbrace{d^{n-2}[s,j] + cost(j,i)}_{\forall j} \right\} \right)$$

does not have shortest path n-1 edges

Dynamic Programming Solution

The Goal is compute $dist^{n-1}[u]$ for all the vertices

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

Algorithm

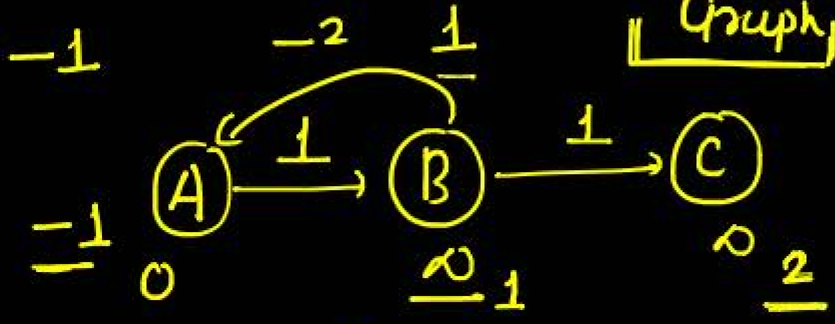
Complexity

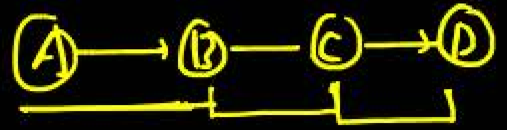
Graph ← $O(V \cdot E) \approx O(V^3)$

```
d[s]=0
        for each v= V-{s}
        do d[v]=∞
        for i= 1 to |v| - 1 do
                for each edge (u, v) ∈ E do
                        if d[v] > d|u] + w(u, v) then
                                d[v] = d[u]     + w(u, v)
                                π [v] = u
        for each edge (u, v) ∈ E do
                if d[v] > d[u] + w(u, v)
                        then report that a negative-weight cycle exists At the end,
```

updation

n-1





(A B)

$d[B] > d[A] + \text{cost}(A\text{-}B)$

$\infty > 0 + \infty \;\; 1$ yes

[B A]

$d[A] > d[B]$

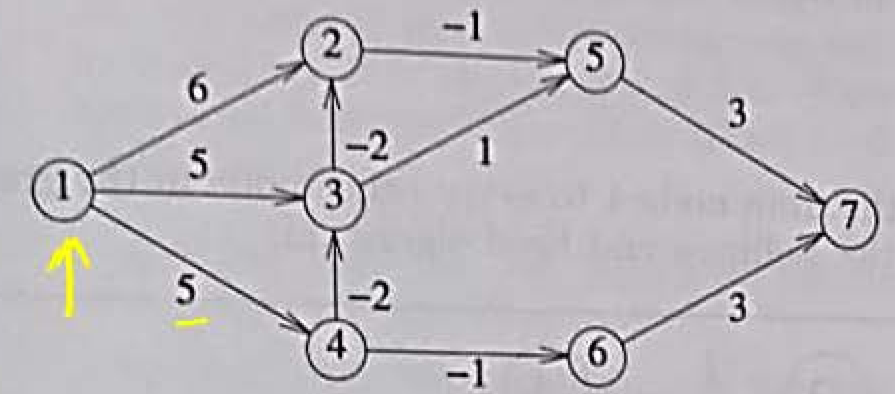
$0 \quad 1 + (-2)$

$\boxed{0 > -1}$

[A B]

$d[B] > -1 + 1$

$\boxed{1 > 0}$



(a) A directed graph

k				$dist^k[1..7]$			
	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

(b) $dist^k$

Minimum No. of edge possible from source to any other vertex

1

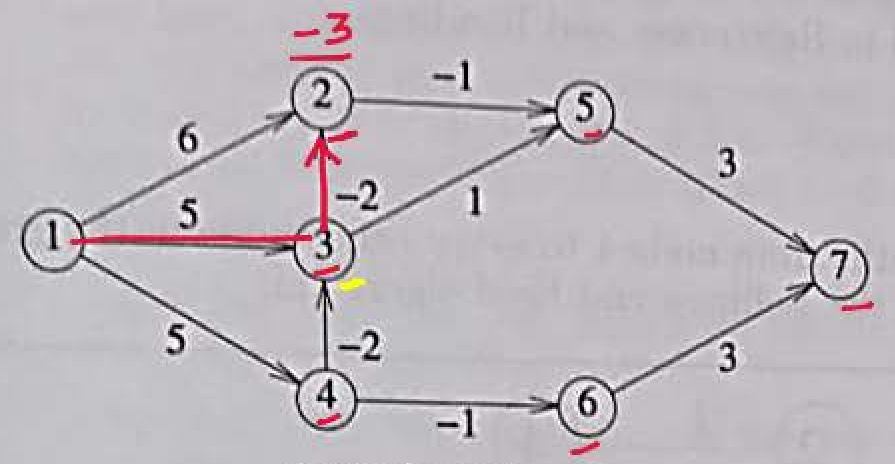
1 Source

$d^1[1] = \underset{}{\overline{\ 0\ }}$ $d^0[1]$

$d^1[2]$ – Shortest distance of vertex 2 with 1 edge $= 6$

$d^1[3] = 5$

$d^1[4] = 5$



(a) A directed graph

k	$dist^k[1..7]$						
k	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

(b) $dist^k$

$d^1[2] = \boxed{6}$

$d^1[3] = 5$

$d^1[4] = 5$

$d^1[5] = \infty$

$d^1[6] = \infty$

$d^1[7] = \infty$

$\boxed{d^2[2]}$ – shortest path for 2 using 2 edges

5 – 2

$$= \min\left\{ d^1[2],\ \min\left\{ \begin{array}{l} d^1[3] + cost(3,2) \\ d^1[6] + cost(4,2) \\ d^1[5] + cost(5,2) \\ d^1[6] + cost(6,2) \\ d^1(7) + cost(7,2) \end{array} \right. \right.$$

6 ,

5 + ∞

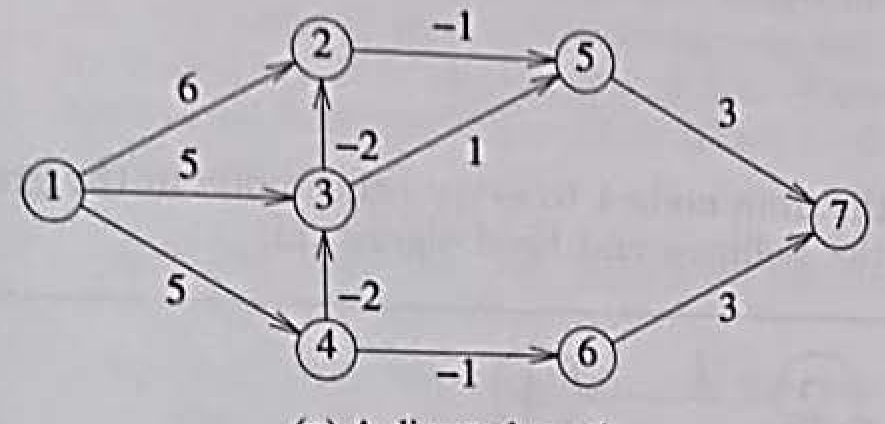
∞ + ∞

∞ + ∞

∞ + ∞

min{6, 3} 3

$d^2[2] = 3$



(a) A directed graph

k	1	2	3	4	5	6	7
	$dist^k[1..7]$						
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

(b) $dist^k$

$d^3[3] = d^2[3]$

$\min \{ d^2[$

$6 + \infty$

$d^1[2] + cost(2,3)$

$d^1[4] + cost(4,3)$ = 3

$5 + (-2)$

$d^1[5] + cost(5,3)$

$\infty + \infty$

$d^1[6] + cost(6,3)$

$\infty \quad \infty$

$d^1(7) + cost(7,3)$

$\infty \quad \infty$

$d^2[3] = \min \{ d^1[3], \ \min$

$5 \quad 3$

$= \boxed{3}$

GATE 2008 | 2 Marks Question

The subset-sum problem is defined as follows. Given a set of n positive integers, $S=\{a_1,a_2,a_3,\ldots,a_n\}$, and positive integer W, is there a subset of S whose elements sum to W? A dynamic program for solving this problem uses a 2-dimensional Boolean array, X, with n rows and $W+1$ columns. $X[i,j], 1\le i\le n, 0\le j\le W$, is TRUE if and only if there is a subset of $\{a_1,a_2,\ldots,a_i\}$ whose elements sum to j.

Which of the following is valid for $2\le i\le n$ and $a_i\le j\le W$?

(A) $X[i,j]=X[i-1,j]\vee X[i,j-a_i]$

(B) $X[i,j]=X[i-1,j]\vee X[i-1,j-a_i]$

(C) $X[i,j]=X[i-1,j]\wedge X[i,j-a_i]$

(D) $X[i,j]=X[i-1,j]\wedge X[i-1,j-a_i]$

GATE 2008 | 2 Marks Question

Which entry of the array X, if TRUE, implies that there is a subset whose elements sum to W?

(A) $X[1, W]$

(B) $X[n, 0]$

(C) $X[n, W]$

(D) $X[n-1, n]$

GATE 2021 Set-1 | 2 Marks Question

Define R_n to be the maximum amount earned by cutting a rod of length n meters into one or more pieces of integer length and selling them. For i> 0, let p[i] denote the selling price of a rod whose length is 1 metres. Consider the array of prices:

$$p[1]=1, p[2]=5, p[3]=8, p[4]=9, p[5]=10, p[6]=17, p[7]=18$$

Which of the following statements is/are correct about R_7 ?

(A) R_7 is achieved by three different solutions

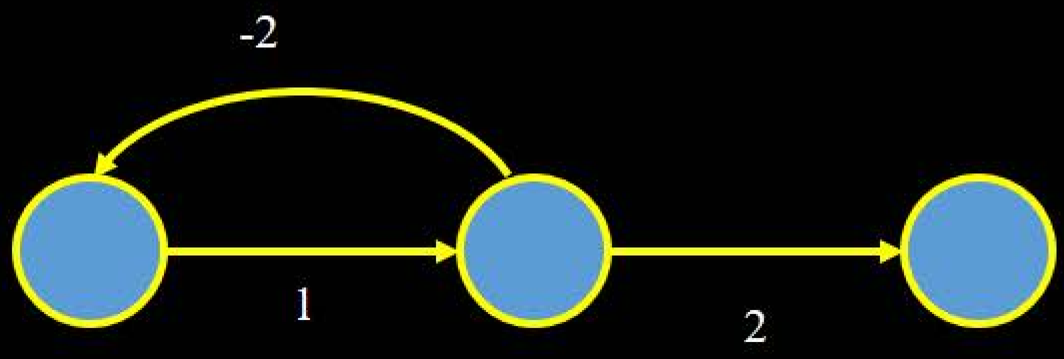
(B) $R_7 = 19$

(C) R_7 cannot be achieved by a solution consisting of three pieces

(D) $\mathrm{R}_7 = 18$

Analysis

$d[v] = \delta(s, v)$. Time = $O(|V|\,|E|)$.



List of Dynamic Programming Solution

- Longest Increasing Subsequence
- Edit Distance
- Minimum Partition
- Ways to Cover a Distance
- Longest Path In Matrix
- Subset Sum Problem
- Optimal Strategy for a Game
- 0-1 Knapsack Problem

Boolean Parenthesization Problem

Shortest Common Supersequence

Matrix Chain Multiplication

Partition problem

Rod Cutting

Coin change problem

Word Break Problem

Maximal Product when Cutting Rope

Dice Throw Problem

Box Stacking

Egg Dropping Puzzle