PAGE No. DATE / /

Mod 3

→ 3.1 : Greedy
- → Dijkstras (single source shortest path)
- → Knapsack
- → Job sequencing
- → Ford fulkerson method (Max flow)

* Dijkstras (for single source shortest path) [O(V^2)] / [O($V\log V$)]

① Given a graph
② Create source / destination table
③ First all destination empty
④ Then keep choosing shortest dist node to source
⑤ Check & calculate from new perspective
⑥ Get a final graph of all nodes min dist.

* Knapsack [O($n\log n$)]

① Given Profit, Weight & KS capacity
② Calculate P/W ratio & arrange in decreasing order
③ Keep adding weight to knapsack
if full can't be accomodated

$$\frac{\text{remaining capacity}}{\text{weight}} \times \text{profit}$$

④ Get final profit

* Jobsequencing [O($n \log n$)]

① Given Profit, deadline
② Max deadline → total Jobs to be sequenced
③ 0 ___ 1 ___ 2 ___ 3
④ Jobs to be arranged for max profit.

Note : deadline doesn't mean, job will take 2 months
It means, we can schedule either in first month or 2nd month.

PAGE No.
DATE / /

* Ford fulkerson method (for max flow)

① Given a graph with edgeweights, source & sink

② Trace every path from source to sink one by one
In each iteration choose min weight
Make a table of

Augmenting Path	Bottleneck capacity

Use that min weight to block one edge every iteration
Repeat until all paths covered & no path left

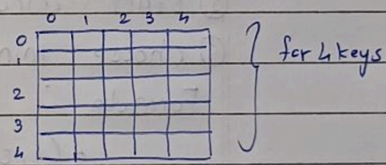
③ Max flow = Σ Bottleneck Capacity

→ 3.2 : Dynamic Programming

* OBST $[O(N^3)]$

① Given keys, frequency & numbers

② We create a matrix starting from 0 always



③ $l = j - i = 0$
directly frequency le skte (min)
fill inthe matrix

④ $l = j - i = 1$
again directly

⑤ $l = j - i = 2$
for each (_,_) there will be two trees get min freq
after totaling level × freq
fill in the matrix but freq o ← parent node number

⋮

$l = j - i = 4$ (serial no. max)

formula

$$c[i,j] = \min \{ c[i, k-1] + c[k,j] \} + w(i,j)$$

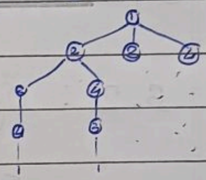
all combos

⑥ finally draw the OBST

PAGE No.
DATE / /

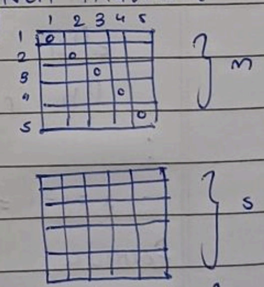
* All pair shortest path $[O(N^3)]$
(Floyd Warshall Algorithm)
① Given a graph,
② Find A^0 ← adjacency matrix
③ Find $A^1, A^2 \ldots A^n$ n : no. of nodes
Formula :
$$A[i,j] = \min \{ A^{k-1}[i,j], A^{k-1}[i,k] + A^{k-1}[k,j] \}$$

* Travelling Salesman Problem (Using DP) $[N^2 \times 2^n]$



① Given a graph
② Create Adjancency Matrix
③ Create BFS Tree
④ Such that ending goes to start again
⑤ start calculating weight at each node
⑥ Choose min & finally get one weight
Formula :
$$g(i,s) = \min_{k \in \{2,3,4\}} \{ C_{ik} + g(k, \{2,3,4\} \ldots \{k\}) \}$$
$$g(i,s) = \min_{k \in s} \{ C_{ik} + g(k, s - \{k\}) \}$$

* Matrix chain Multiplication $[O(N^3)]$
① We need to find Parenthesization of given Matrices
② Two tables starting | m & s



Find m[1,2], m[3,4], m[2,3] . . . -
$A_1 \cdot A_2$
$5 \times 4 \quad 4 \times 6$
Find m[1,3], m[2,4]
Find min at each step & fill full(half) table
Formula : $m[i,j] = \min \{ m[i,k] + m[k+1,j] + d_{i-1} * d_k * d_j \}$
③ Using s table get parenthesization.

PAGE N
DATE

* LCS [O(m×n)]

Formula :

```
if (input[i] == input[j])
        T[i][j] = T[i-1][j-1]+1 ;
else
        T[i][j] = max (T[i-1][j], T[i][j-1]) ;
```