Data Structures and Algorithms - **EE 390**

Module 8
Dynamic Programming

Dr. Anirban Dasgupta

Assistant Professor

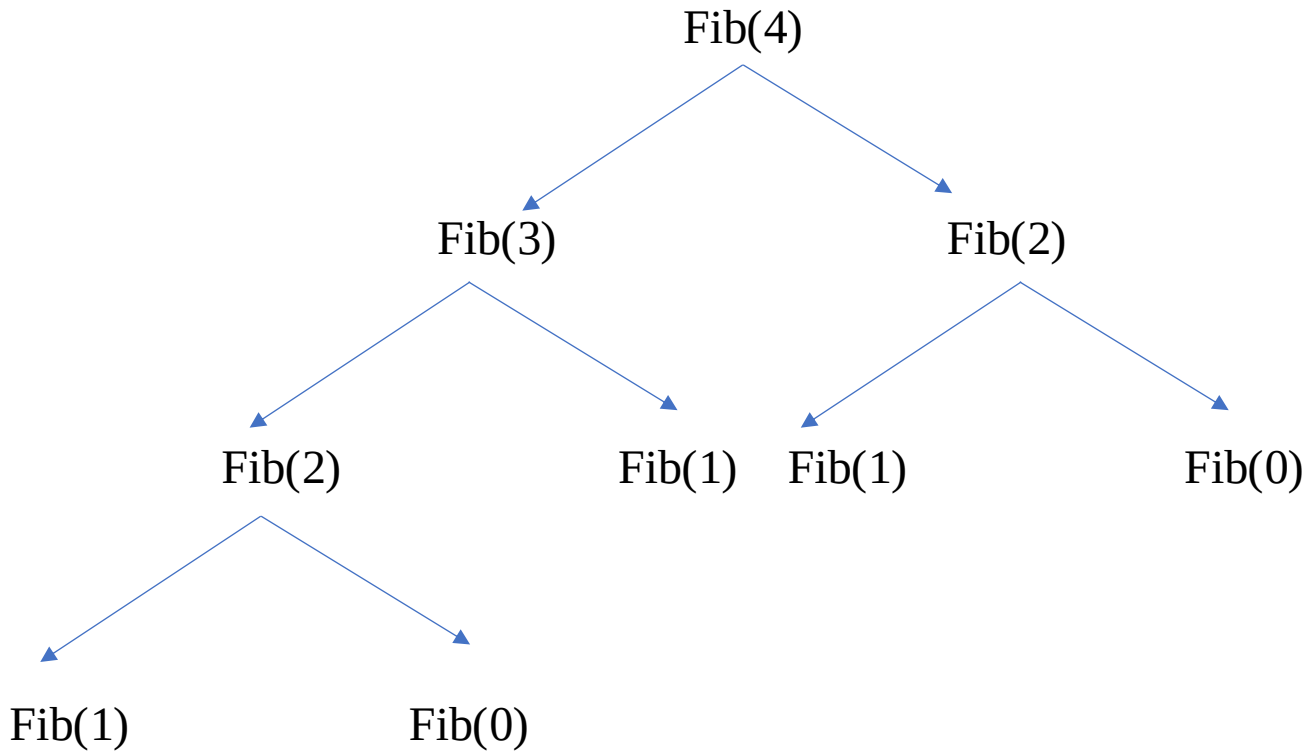
Department of Electronics And Electrical Engineering, IIT Guwahati

Dynamic Programming

- Dynamic Programming (DP) is mainly an optimization following principle of optimality.
- Wherever we see a recursive solution that has repeated calls for same inputs, we can optimize it using DP.
- The idea is to simply store the results of subproblems, so that we do not have to re-compute them when needed later.
- This simple optimization may reduce time complexities from exponential to polynomial.

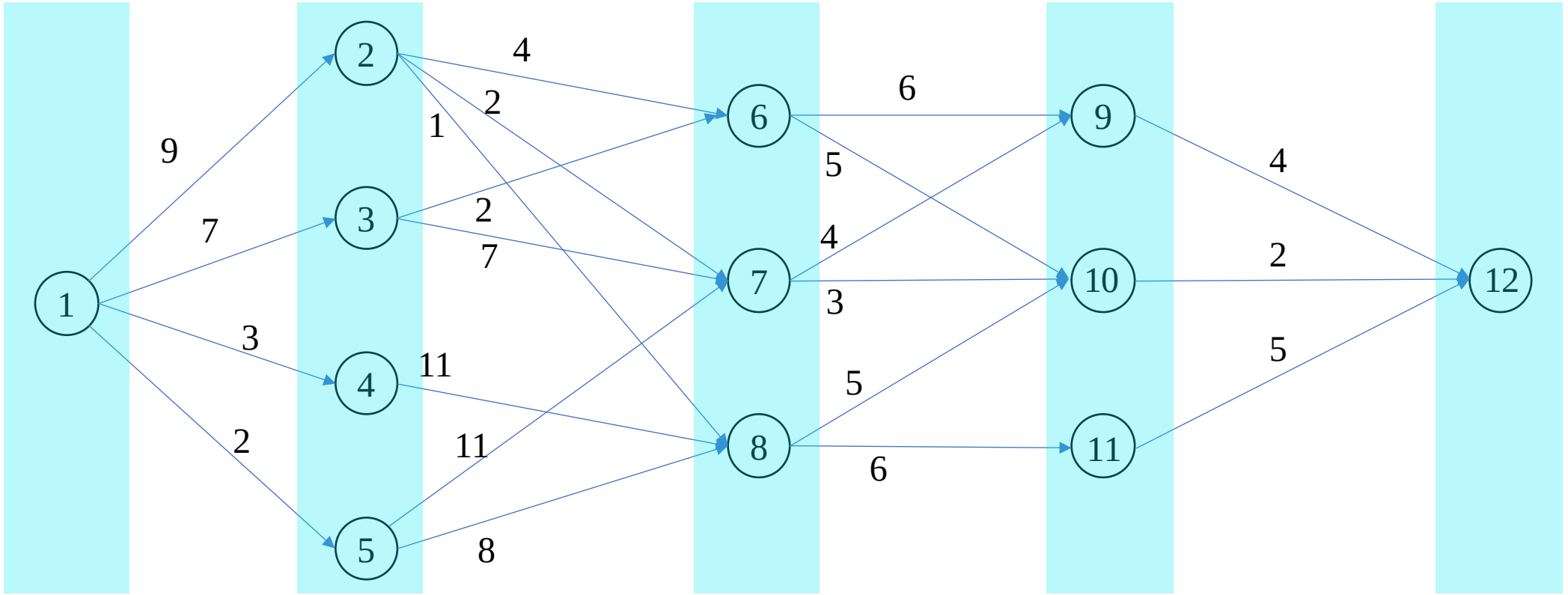
Application of Dynamic Programming

- Optimal substructure
  - an ideal answer for an issue can be built from ideal answers for subproblems
- Overlapping problems
  - solutions of the same sub-problems are required again and again

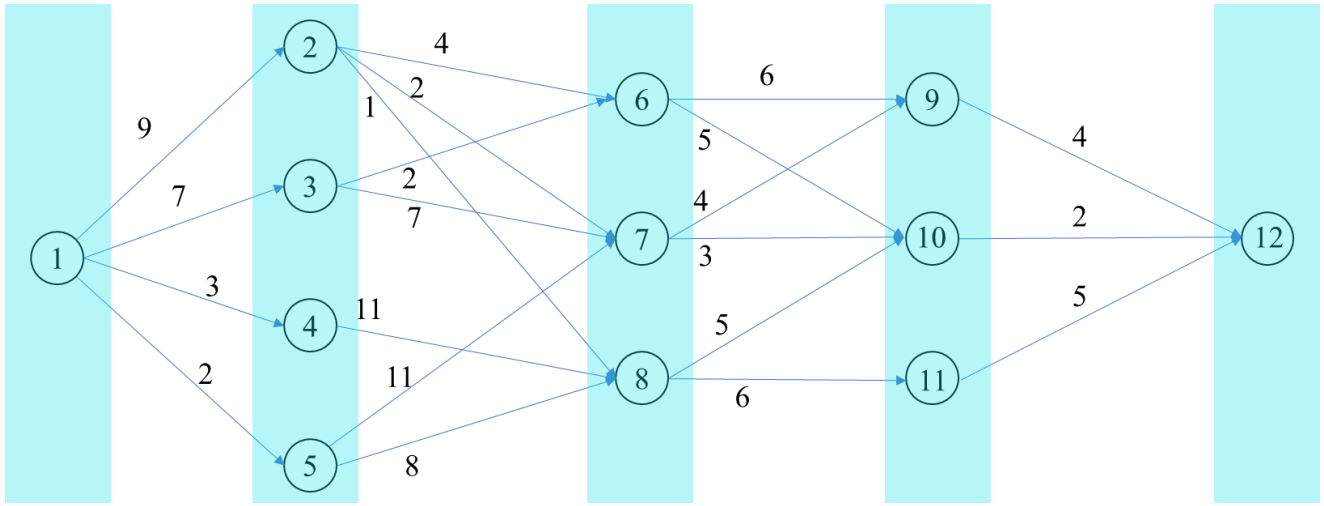


Shortest Path in Multistage Graph

A multistage graph is a directed, weighted graph in which the nodes can be divided into a set of stages such that all edges are from a stage to next stage only



Multistage Graph



$W(S,V)$

$W(4,9) = 4$
$W(4,10) = 2$
$W(4,11) = 5$

$$C(3,6) = \min\begin{cases} W(6,9) + C(4,9) \\ W(6,10) + C(4,10) \end{cases}$$

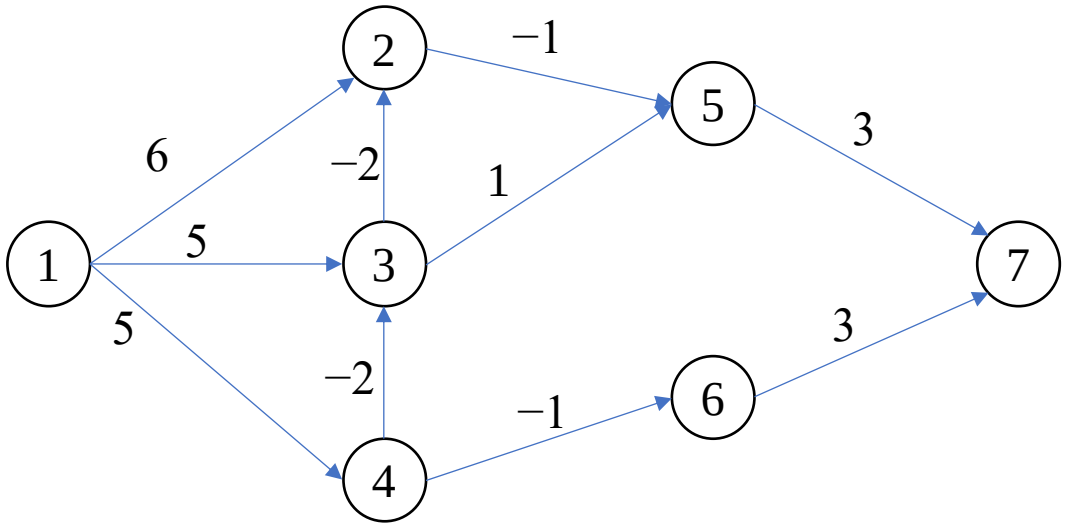
$$C(3,6) = \min\begin{cases} 6+4 \\ 5+2 \end{cases}$$

$$C(3,7) = \min\begin{cases} 4+4 \\ 3+2 \end{cases}$$

$$C(3,8) = \min\begin{cases} 5+2 \\ 6+5 \end{cases}$$

V	1	2	3	4	5	6	7	8	9	10	11	12
C	16	7	9	18	15	7	5	7	4	2	5	0
d	2/3	7	6	8	8	10	10	10	12	12	12	12

Bellman Ford Algorithm
–Single Source Shortest Path



Dijkstra's algorithm may fail for negative edge weights

Time complexity of Bellman-Ford is $O(V * E)$, compared to Dijkstra's $O((V + E) \log V)$

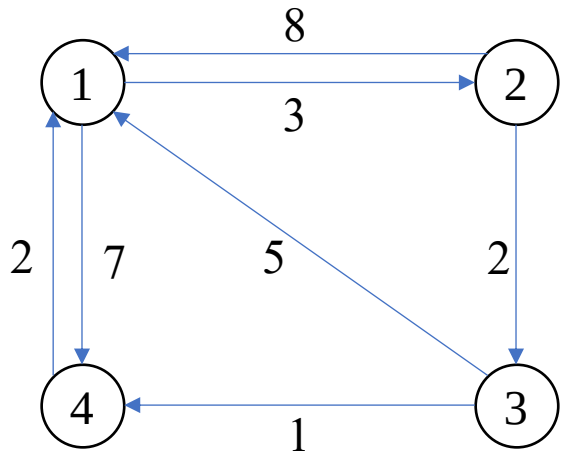
Epoch	1	2	3	4	5	6	7
#1	0	∞	∞	∞	∞	∞	∞
#2	0	6	5	5	∞	∞	∞
#3	0	3	3	5	5	4	∞
#4	0	1	3	5	2	4	7
#5	0	1	3	5	0	4	5
#6	0	1	3	5	0	4	3

Optimization Problem

- Given a graph and a source vertex in the graph, find the shortest paths from source to all vertices in the given graph.
- The graph may contain negative weight edges.

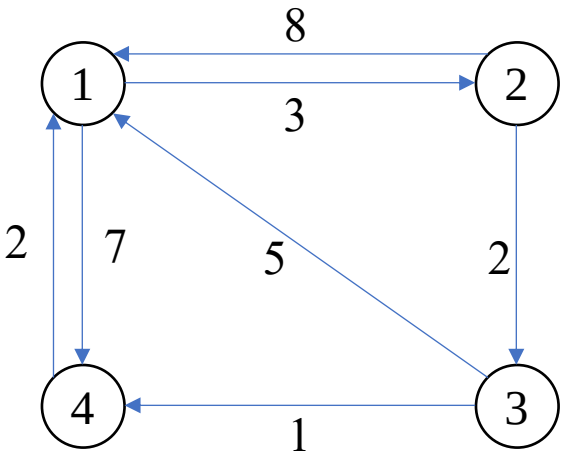
All Pairs Shortest Path (Floyd-Warshall)

- Floyd-Warshall algorithm used to find all pair shortest path problem from a given weighted graph.
- Generate a matrix, which will represent the minimum distance from any node to all other nodes in the graph.



- Dijkstra's algorithm can solve for one vertex in $O(n^2)$.
- Applying Dijkstra's algorithm to all n vertices will take $O(n^3)$.

All Pairs Shortest Path (Floyd-Warshall)



$$A_0 = \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & \infty \\ 5 & \infty & 0 & 1 \\ 2 & \infty & \infty & 0 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & \infty & 0 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & 3 & 5 & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 0 & 3 & 5 & 6 \\ 7 & 0 & 2 & 3 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} 0 & 3 & 5 & 6 \\ 5 & 0 & 2 & 3 \\ 3 & 6 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix}$$

Integer (0/1) Knapsack Problem

Problem: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection **as a whole** so that the total weight is less than or equal to a given limit and the total value is as large as possible.

objects	**O1**	**O2**	**O3**
weights	2	4	8
profits	20	25	60
profit/weight	10	6.25	7.5

Maximum Profit using Greedy Approach:
20+60 = 80

However, selecting 2 and 3 can give profit of 85!!!

Maximize profit, such that total weight<=12

n=3

m=12

Greedy approach will fail!!!

Integer Knapsack Problem – Brute Force

	O1	O2	O3	O4
W	3	4	6	5
P	2	3	1	4

Brute Force: $2^n = 2^4 = 16$

Maximize profit, such that total weight<=8

O1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
O2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
O3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
O4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
P	0	4	1	5	3	7	4	8	2	6	3	7	5	9	6	10
W	0	5	6	11	4	9	10	15	3	8	9	14	7	12	13	18

Integer Knapsack Problem – DP

	O1	O2	O3	O4
W	3	4	6	5
P	2	3	1	4

w	p	i	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
3	2	1	0	0	0	2	2	2	2	2	2
4	3	2	0	0	0	2	3	3	3	5	5
5	4	3	0	0	0	2	3	3	3	5	5
6	1	4	0	0	0	2	3	4	4	5	6

$$m(i, w) = \max\{m(i-1, w), m(i-1, w-w_i) + p_i\}$$

Sum of Subsets Problem

Problem: Given a set of positive integers and a target sum, the task is to find all the subsets of the given set whose elements sum up to the target sum.

$S = \{5,10,12,13,15,18\}$, No. of elements, $n = 6$, Sum required, $m = 30$

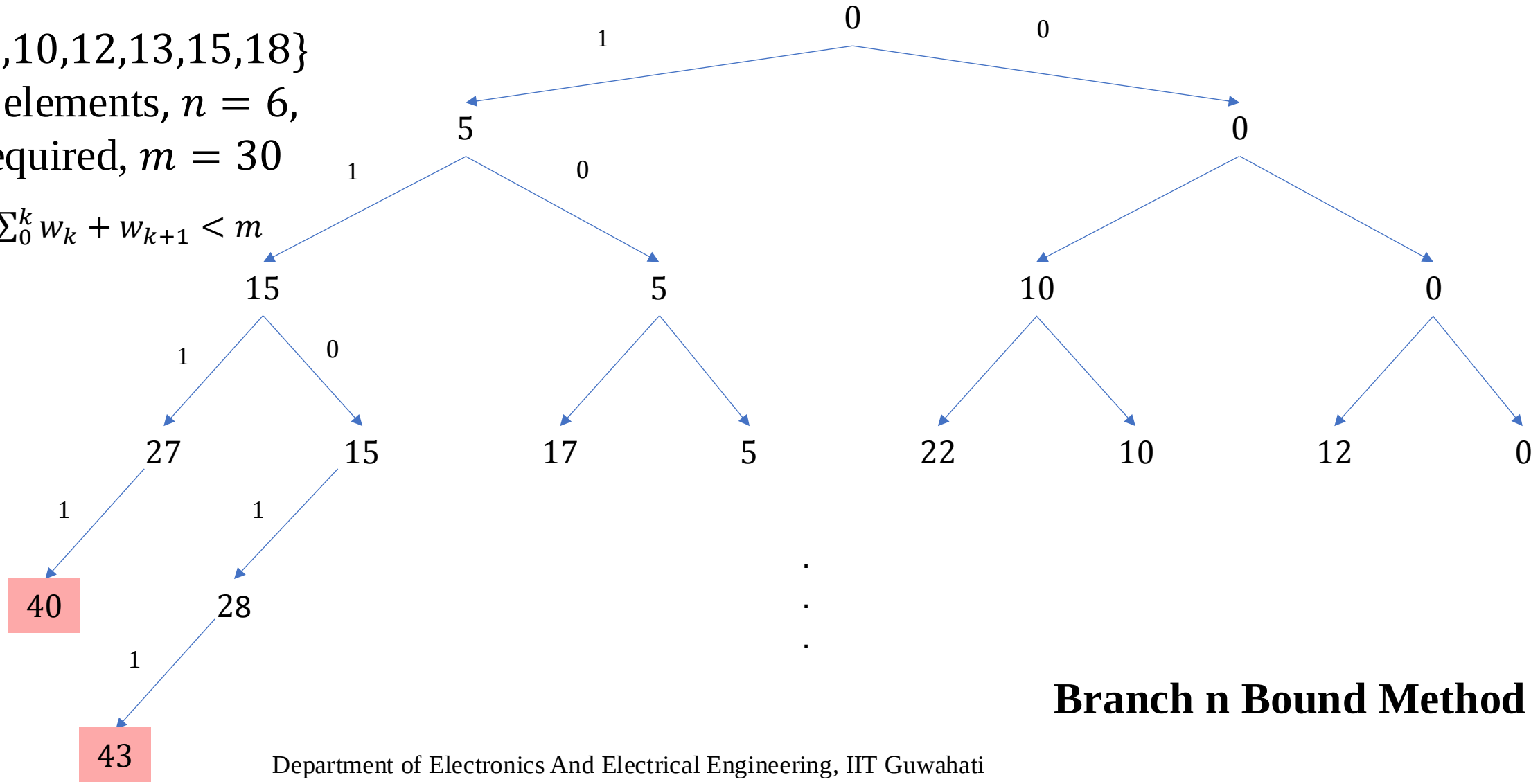
Brute Force: $2^n = 2^6 = 64$

5	0	0	0	0	0	0	0	0	0	0	…	1	1	1	1	1	1	1	1	1	1	1
10	0	0	0	0	0	1	1	0	0	0	…	0	1	0	1	0	0	0	1	1	1	1
12	0	0	0	0	1	0	1	1	0	0	…	1	0	1	1	0	0	1	0	1	0	1
13	0	0	0	1	0	0	0	0	0	1	…	1	0	1	0	0	1	0	0	1	1	1
15	0	0	1	0	0	0	0	0	1	1	…	0	1	0	0	1	0	0	1	0	1	1
18	0	1	0	0	0	0	0	1	1	0	…	1	1	0	1	1	1	0	0	1	1	0
Sum	0	18	15	13	12	10	22	30	33	28		38	48	30	45	41	36	17	30	58	61	55	73

Sum of Subsets Problem

$S = \{5,10,12,13,15,18\}$
No. of elements, $n = 6$,
Sum required, $m = 30$

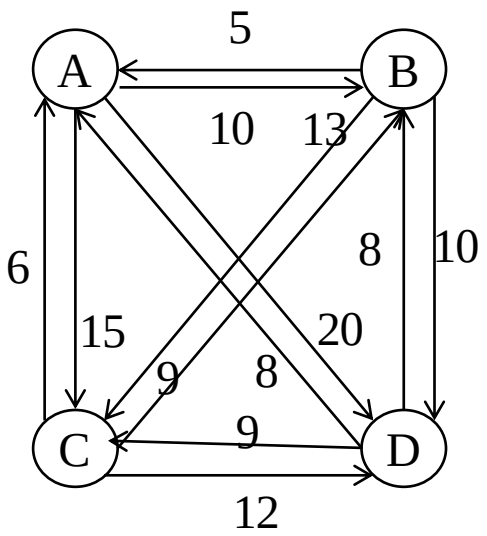
Constraint $\sum_0^k w_k + w_{k+1} < m$



Branch n Bound Method

Travelling Salesman Problem

Problem: Given a list of cities and the costs of travelling between each pair of cities, find the minimum cost that visits each city exactly once and returns to the origin city.



Greedy Approach:

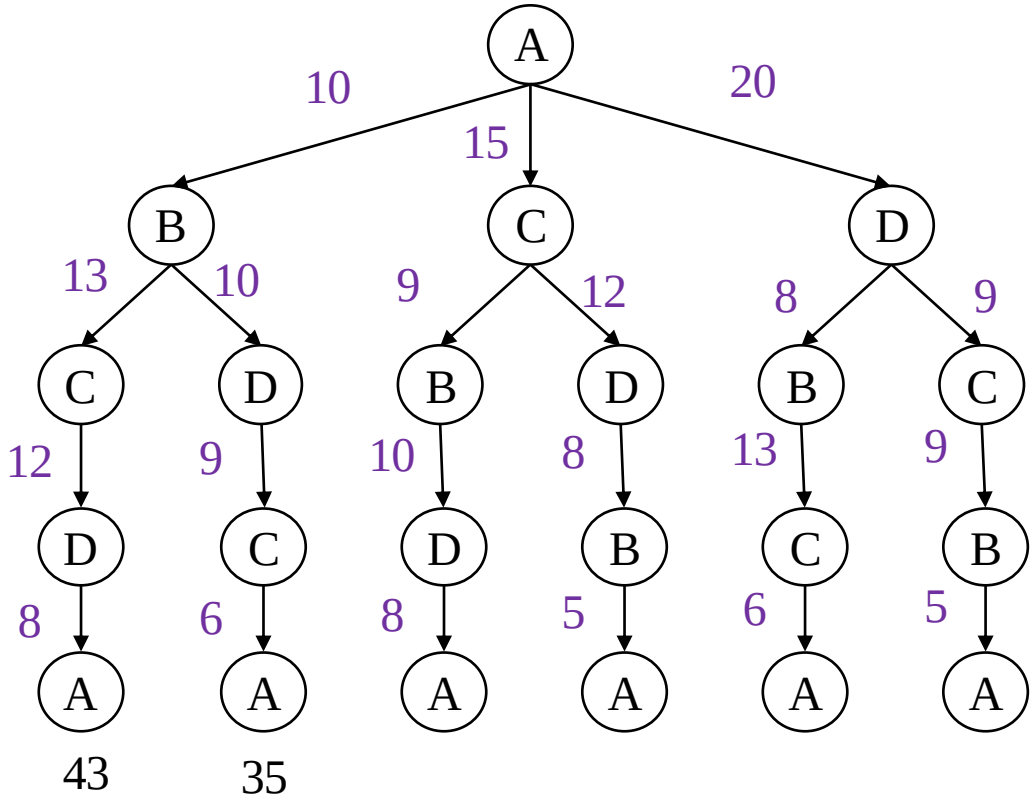
A→B → C→D →A

Cost = 10+13+12+ 8=43

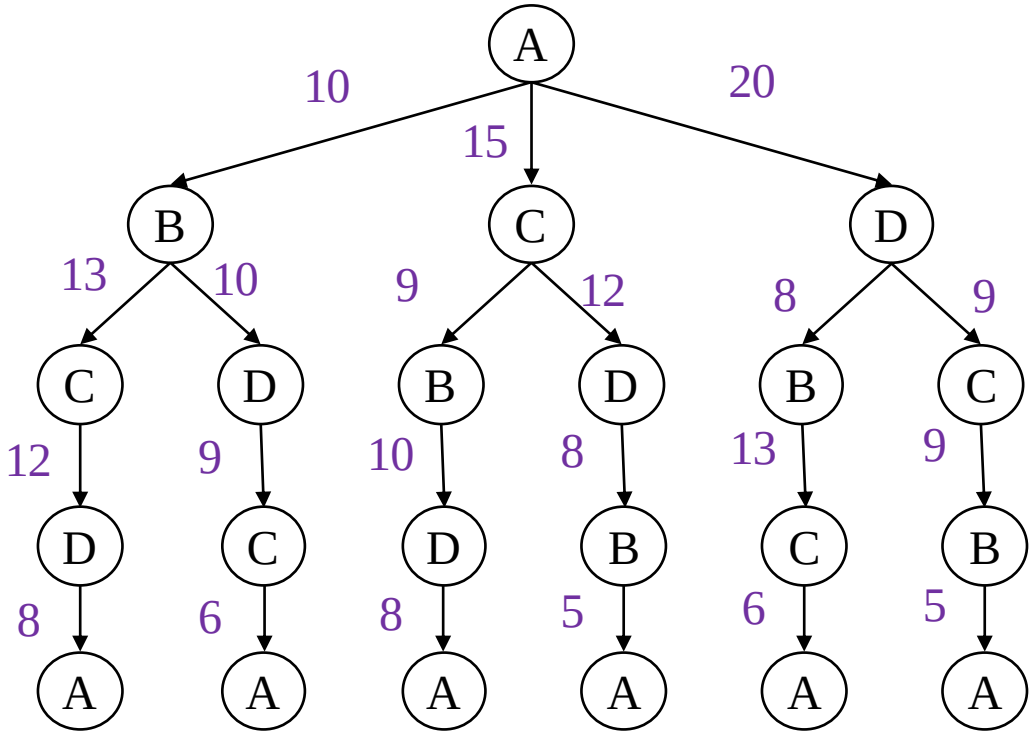
Adjacency matrix

	A	B	C	D
A	0	10	15	20
B	5	0	13	10
C	6	9	0	12
D	8	8	9	0

Greedy approach fails!!!



Travelling Salesman Problem



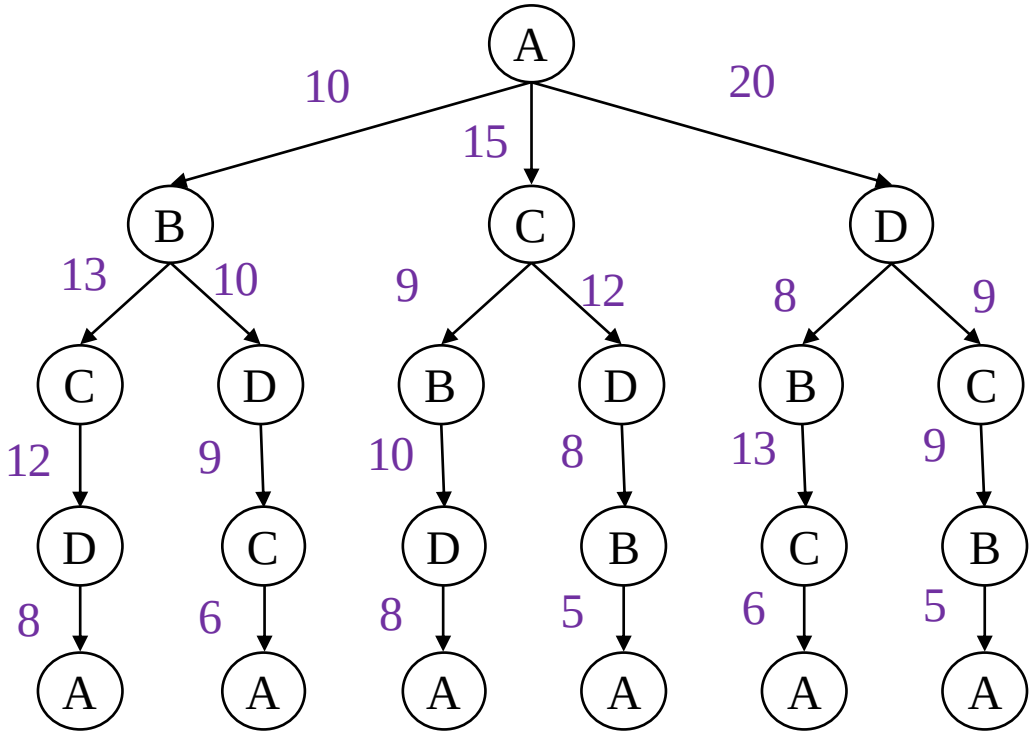
$$g(\mathrm{A},\{\mathrm{B},\mathrm{C},\mathrm{D}\}) = \min_{\mathrm{k}\in\{\mathrm{B},\mathrm{C},\mathrm{D}\}} C(\mathrm{A},k) + g(k,\{\mathrm{B},\mathrm{C},\mathrm{D}\}) - \{k\})$$

Recurrence relation

$$g(i,S) = \min_{\mathrm{k}\in S} C(i,k) + g(k,S-\{k\})$$

$$g(\mathrm{B},\{\phi\}) = 5$$
$$g(\mathrm{C},\{\phi\}) = 6$$
$$g(\mathrm{D},\{\phi\}) = 8$$
$$g(\mathrm{B},\{\mathrm{C}\}) = C(\mathrm{B},\mathrm{C}) + g(\mathrm{C},\phi) = 13+6 = 19$$
$$g(\mathrm{B},\{\mathrm{D}\}) = C(\mathrm{B},\mathrm{D}) + g(\mathrm{D},\phi) = 10+8 = 18$$
$$g(\mathrm{C},\{\mathrm{B}\}) = C(\mathrm{C},\mathrm{B}) + g(\mathrm{B},\phi) = 13+5 = 18$$
$$g(\mathrm{C},\{\mathrm{D}\}) = C(\mathrm{C},\mathrm{D}) + g(\mathrm{D},\phi) = 12+8 = 20$$
$$g(\mathrm{D},\{\mathrm{B}\}) = C(\mathrm{D},\mathrm{B}) + g(\mathrm{B},\phi) = 8+5 = 13$$
$$g(\mathrm{D},\{\mathrm{C}\}) = C(\mathrm{D},\mathrm{C}) + g(\mathrm{C},\phi) = 9+6 = 15$$

Travelling Salesman Problem



$$g(\mathrm{A},\{\mathrm{B},\mathrm{C},\mathrm{D}\}) = \min_{k\in\{\mathrm{B},\mathrm{C},\mathrm{D}\}} C(\mathrm{A},k) + g(k,\{\mathrm{B},\mathrm{C},\mathrm{D}\}) - \{k\})$$

Recurrence relation

$$g(i,S\}) = \min_{k\in S} C(i,k) + g(k,S-\{k\})$$

$$g(\mathrm{B},\{\mathrm{C},\mathrm{D}\}) = \min\{33,25\} = 25$$
$$g(\mathrm{C},\{\mathrm{B},\mathrm{D}\}) = \min\{27,25\} = 25$$
$$g(\mathrm{D},\{\mathrm{B},\mathrm{C}\}) = \min\{27,23\} = 23$$

$$g(\mathrm{A},\{\mathrm{B},\mathrm{C},\mathrm{D}\}) = \min\{35,40,43\} = 35$$

Path: A→B → D→C →A

References

- Cormen, Leiserson, Rivest, and Stein, *Introduction to Algorithms*, 3rd edition, The MIT Press, McGraw-Hill, 2001.
- Bellman, Richard E., and Stuart E. Dreyfus. *Applied dynamic programming*. Vol. 2050. Princeton University Press, 2015.
- Bellman, Richard E. *Dynamic programming*. Princeton University Press, 2010.