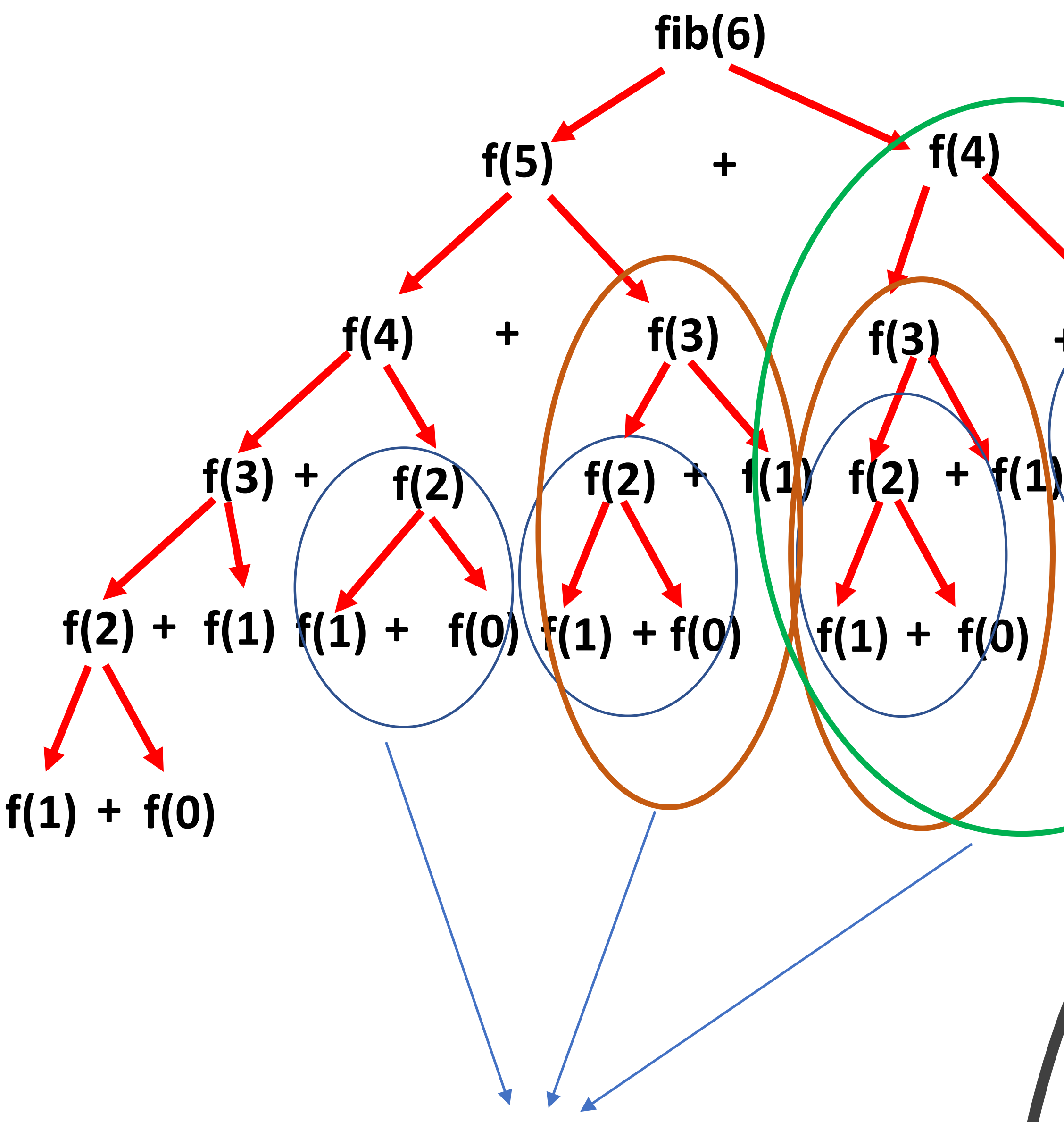
Dynamic Programming

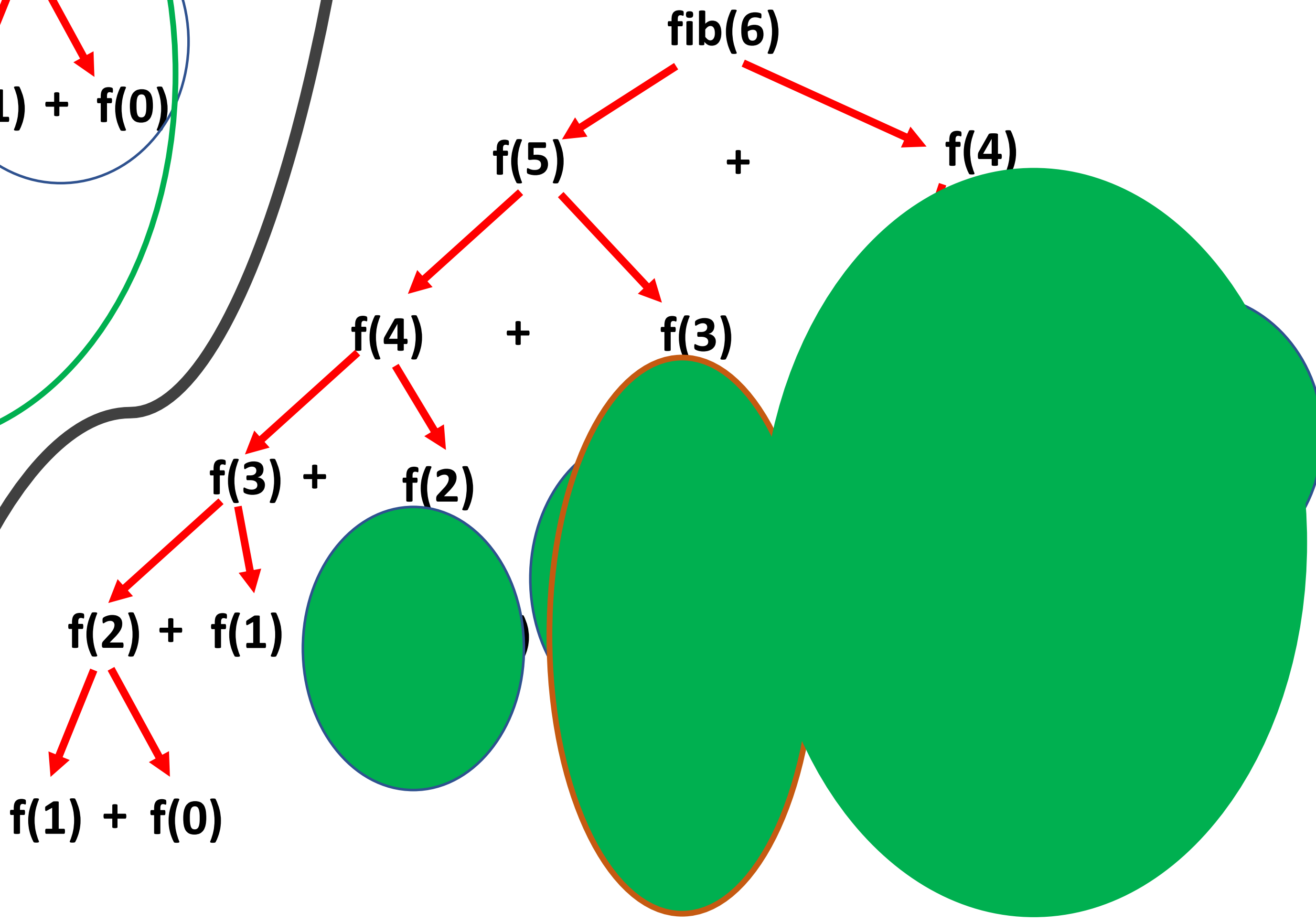
Comparison of Dynamic Programming over Divide-And-Conquer and Greedy Method

Divide-and-conquer	**Dynamic Programming**
• The Divide-and-conquer algorithms partition the problem into disjoint subproblems, **solve the subproblems recursively, and then combine their solutions to solve the original problem.** • A divide-and-conquer algorithm does more work than necessary, **repeatedly solving the common subsubproblems**.	• Dynamic programming **applies when the subproblems overlap—that is, when subproblems share subsubproblems.** • A dynamic-programming algorithm solves each subsubproblem just once and then **saves its answer in a table**, thereby avoiding the work of recomputing the answer every time it solves each subsubproblem
Greedy Method	**Dynamic Programming**
• Greedy Algorithms **first make a greedy choice**(the choice that looks best at the time) and then solve a resulting subproblem ,without bothering to solve all possible related smaller subproblems	• Dynamic Programming solves all its related subproblems of a problem and a choice is made to find an optimal solution to the problem. • Principle of optimality
Both are applied to optimization problems having optimal substructure.	



Re computation of same subproblems

By saving the sub problems results we can reuse the result which avoids the re-computation of same sub problems (i.e., no recursive calls for same sub problems). Which is the top-down implementation of DP for nth Fibonacci number.



Dynamic Programming – Introduction

The Sequence of four steps to be followed for developing a dynamic-programming algorithm :

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution, typically in a bottom-up fashion.
4. Construct an optimal solution from computed information.

<u>Elements of Dynamic Programming:</u>

1. Optimal substructure:

   A problem exhibits ***optimal substructure*** if an optimal solution to the problem contains within it optimal solutions to subproblems.
2. Overlapping subproblems.
   - Two subproblems are overlapping if they are really the same subproblem that occurs as a subproblem of different problems
   - Saving the solution of subproblems in a table can reduce the re-computation of similar subproblem.

Contd….

There are two equivalent ways to implement a dynamic-programming approach.

1. top-down with memoization.
2. bottom-up method.

Top-down with memoization:

- Write the recursive procedure naturally
- ***Save the result of each subproblem in a table***
- Each new recursive call **first checks the table for the instance of the subproblem to be solved**. If table contains the solution, it return the saved value otherwise procedure compute the value in the usual manner

Top-down approach can be considered as the recursive procedure has been ***memoized***; it "remembers" what results it has computed previously.

Bottom-up approach:

- Solves the subproblems based on the size.
- Sort the subproblems by size and solve them in the size order, smallest first.
- When solving a particular subproblem S_i, ensure that all the subproblems of S_i are solved and saved the result.

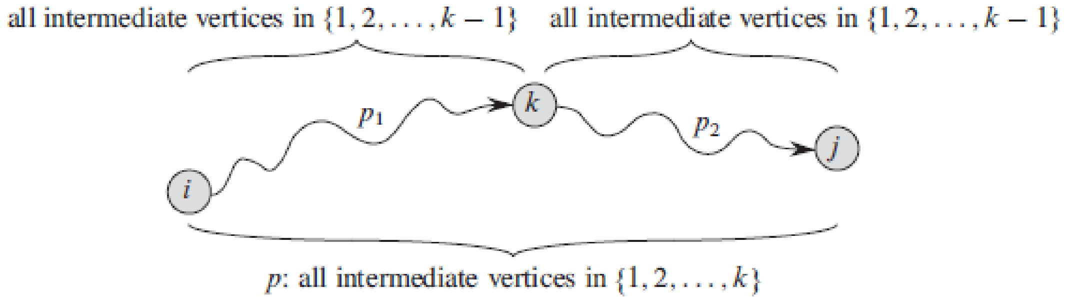
These two approaches yield algorithms with the **same asymptotic running time**, The bottom-up approach often has much better constant factors, since it has less overhead for procedure calls.

All-Pairs Shortest paths

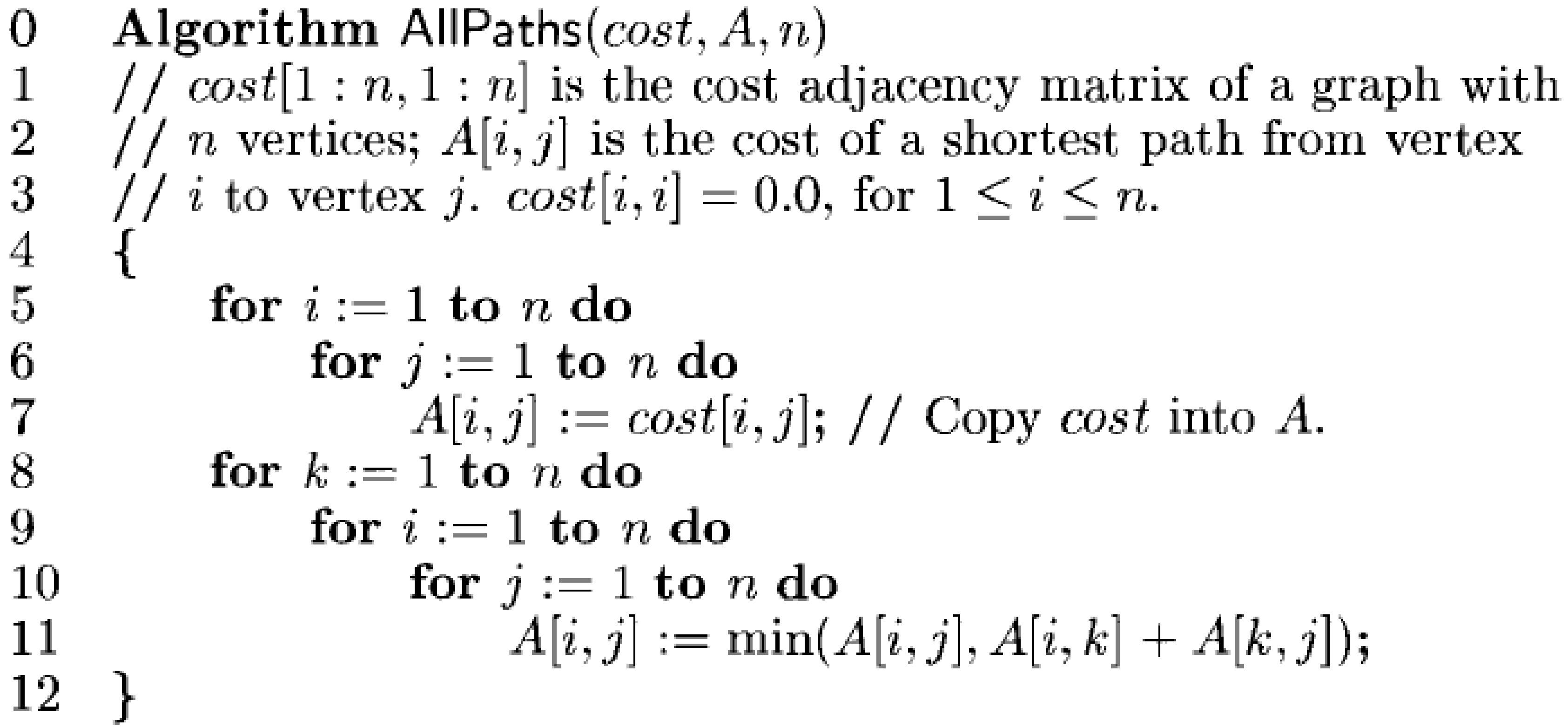
- Let G=(V,E) be a directed graph with n vertices and cost is the adjacency matrix for G such that cost(i , i) = 0 $1 \le i \le n$. The cost(i, j) is the length of edge <i, j> if <i , j> ϵ E(G) and cost (i , j)= ∞ if $i \neq j$ and <i , j> ϵ E(G) . The all-pairs shortest-path is to determine a matrix A such that A(i,j) is the length of a shortest path from i to j.

Terminology	**Description**
$A^{\mathbf{k}}(i , j)$	Length of Shortest path from i to j going through no vertex of index greater than k.
Optimal substructure of the problem	A Shortest path from i to j going through no vertex higher than k either goes through vertex k or it does not. If it does then $A^{k}(i , j) = A^{k-1}(i , k) + A^{k-1}(k , j)$. If it does not, then no intermediate vertex has index greater than k-1 then $A^{k}(i , j) = A^{k-1}(i , j)$
Recurrence formula	$A^{k}(i , j) = \min \{A^{k-1}(i , j) , A^{k-1}(i , k) + A^{k-1}(k , j)\}$ $k \ge 1$
Calculate the optimal value	$A^{n}(i , j)$ where n is the number of vertices.

All-Pairs Shortest paths

Terminology	**Description**
$A^{k}(i, j)$	Length of Shortest path from i to j going through no vertex of index greater than k.
Optimal substructure of the problem	A Shortest path from i to j going through no vertex higher than k either goes through vertex k or it does not. If it does then $A^{k}(i, j) = A^{k-1}(i, k) + A^{k-1}(k, j)$. If it does not, then no intermediate vertex has index greater than k-1 then $A^{k}(i, j) = A^{k-1}(i, j)$  all intermediate vertices in $\{1, 2, \ldots, k-1\}$ all intermediate vertices in $\{1, 2, \ldots, k-1\}$ k p_1 p_2 i j p: all intermediate vertices in $\{1, 2, \ldots, k\}$
Recurrence formula	$A^{k}(i, j) = \text{cost}[i, j] \quad k = 0$ $= \min \{A^{k-1}(i, j),\ A^{k-1}(i, k) + A^{k-1}(k, j)\} \quad k \geq 1$
Calculate the optimal value	$A^{k}(i, j)$ Calculate the optimal value in a bottom-up method in an increasing order of k (k=1,2,3,….n)

Flyod- warshall

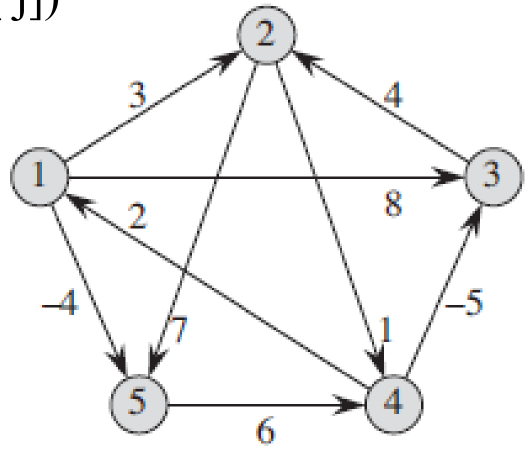


```
Algorithm AllPaths($cost, A, n$)
// $cost[1:n, 1:n]$ is the cost adjacency matrix of a graph with
// $n$ vertices; $A[i, j]$ is the cost of a shortest path from vertex
// $i$ to vertex $j$. $cost[i, i] = 0.0$, for $1 \leq i \leq n$.
{
    for $i := 1$ to $n$ do
        for $j := 1$ to $n$ do
   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

Time complexity : $\Theta(n^3)$ where n is number of vertices of the graph

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

$$\begin{array}{c|ccccc} A^{0} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & 8 & \infty & -4 \\ 2 & \infty & 0 & \infty & 1 & 7 \\ 3 & \infty & 4 & 0 & \infty & \infty \\ 4 & 2 & \infty & -5 & 0 & \infty \\ 5 & \infty & \infty & \infty & 6 & 0 \end{array}$$



$$A^{1}[1][2] = \min(A^{0}[1][2],\ A^{0}[1][1] + A^{0}[1][2])$$
$$= \min(3,\ 0+3) = 3$$

$$A^{1}[1][3] = \min(A^{0}[1][3],\ A^{0}[1][1] + A^{0}[1][3])$$
$$= \min(8,\ 0+8) = 8$$

$$A^{1}[1][4] = \min(A^{0}[1][4],\ A^{0}[1][1] + A^{0}[1][4])$$
$$= \min(\infty,\ 0+\infty) = \infty$$

$$A^{1}[1][5] = \min(A^{0}[1][5],\ A^{0}[1][1] + A^{0}[1][5])$$
$$= \min(-4,\ 0-4) = -4$$

$$A^{1}[4][1] = \min(A^{0}[4][1],\ A^{0}[4][1] + A^{0}[1][1])$$
$$= \min(2,\ 2+0) = 2$$

$$A^{1}[4][2] = \min(A^{0}[4][2],\ A^{0}[4][1] + A^{0}[1][2])$$
$$= \min(\infty,\ 2+3) = 5$$

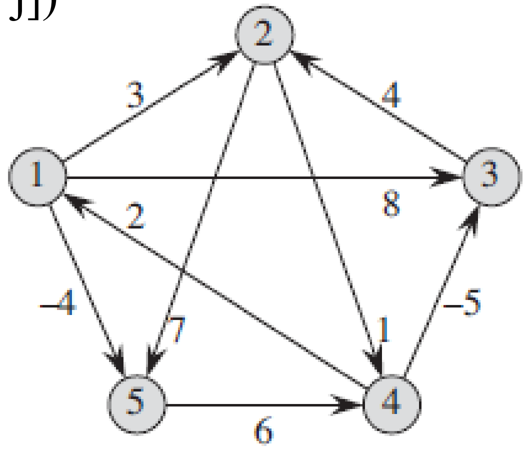
$$A^{1}[4][3] = \min(A^{0}[4][3],\ A^{0}[4][1] + A^{0}[1][3])$$
$$= \min(-5,\ 2+8) = -5$$

$$A^{1}[4][5] = \min(A^{0}[4][5],\ A^{0}[4][1] + A^{0}[1][5])$$
$$= \min(\infty,\ 2-4) = -2$$

$$\begin{array}{c|ccccc} A^{1} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & 8 & \infty & -4 \\ 2 & \infty & 0 & \infty & 1 & 7 \\ 3 & \infty & 4 & 0 & \infty & \infty \\ 4 & 2 & \cancel{\infty}\ 5 & -5 & 0 & \cancel{\infty}\ -2 \\ 5 & \infty & \infty & \infty & 6 & 0 \end{array}$$

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

A^1	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$$A^{2}[1][2] = \min(A^{1}[1][2],\ A^{1}[1][2] + A^{1}[2][2])$$
$$= \min(3, 3+0) = 3$$
$$A^{2}[1][3] = \min(A^{1}[1][3],\ A^{1}[1][2] + A^{1}[2][3])$$
$$= \min(8, 3+\infty) = 8$$
$$A^{2}[1][4] = \min(A^{1}[1][4],\ A^{1}[1][2] + A^{1}[2][4])$$
$$= \min(\infty, 3+1) = 4$$
$$A^{2}[1][5] = \min(A^{1}[1][5],\ A^{1}[1][2] + A^{1}[2][5])$$
$$= \min(-4, 3+7) = -4$$

A^2	1	2	3	4	5
1	0	3	8	~~∞~~ 4	-4
2	∞	0	∞	1	7
3	∞	4	0	~~∞~~ 5	~~∞~~ 11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$$A^{2}[3][4] = \min(A^{1}[3][4],\ A^{1}[3][2] + A^{1}[2][4])$$
$$= \min(\infty, 4+1) = 5$$
$$A^{2}[3][5] = \min(A^{1}[3][5],\ A^{1}[3][2] + A^{1}[2][5])$$
$$= \min(\infty, 4+7) = 11$$

A^2	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

A^3	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	~~5~~ -1	-5	0	-2
5	∞	∞	∞	6	0

A^4	1	2	3	4	5
1	0	3	~~8~~ -1	4	-4
2	~~∞~~ 3	0	~~∞~~ -4	1	~~7~~ -1
3	~~∞~~ 7	4	0	5	~~11~~ 3
4	2	-1	-5	0	-2
5	~~∞~~ 8	~~∞~~ 5	~~∞~~ 1	6	0

$$A^3[4][2] = \min(A^2[4][2], A^2[4][3] + A^2[3][2])$$
$$= \min(5, -5+4) = -1$$

$$A^4[1][3] = \min(A^3[1][3], A^3[1][4] + A^3[4][3])$$
$$= \min(8, 4-5) = -1$$

$$A^4[2][1] = \min(A^3[2][1], A^3[2][4] + A^3[4][1])$$
$$= \min(\infty, 1+2) = 3$$

$$A^4[2][3] = \min(A^3[2][3], A^3[2][4] + A^3[4][3])$$
$$= \min(\infty, 1-5) = -4$$

$$A^4[2][5] = \min(A^3[2][5], A^3[2][4] + A^3[4][5])$$
$$= \min(7, 1-2) = -1$$

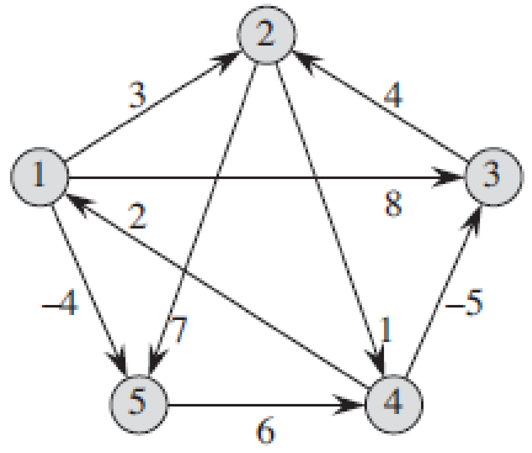
$$A^4[3][1] = \min(A^3[3][1], A^3[3][4] + A^3[4][1])$$
$$= \min(\infty, 5+2) = 7$$

$$A^4[3][5] = \min(A^3[3][5], A^3[3][4] + A^3[4][5])$$
$$= \min(11, 5-2) = 3$$

$$A^4[5][1] = \min(A^3[5][1], A^3[5][4] + A^3[4][1])$$
$$= \min(\infty, 6+2) = 8$$

$$A^4[5][2] = \min(A^3[5][2], A^3[5][4] + A^3[4][2])$$
$$= \min(\infty, 6-1) = 5$$

$$A^4[5][3] = \min(A^3[5][3], A^3[5][4] + A^3[4][3])$$
$$= \min(\infty, 6-5) = 1$$

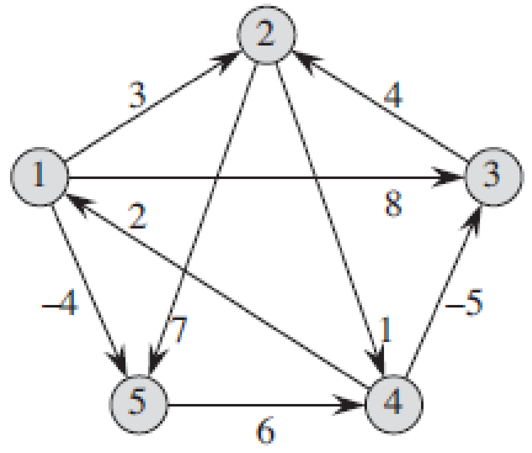


A^4	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

$$A^5[1][2] = \min(A^4[1][2], A^4[1][5] + A^4[5][2])$$
$$= \min(3, -4+5) = 1$$

$$A^5[1][3] = \min(A^4[1][3], A^4[1][5] + A^4[5][3])$$
$$= \min(-1, -4+1) = -3$$

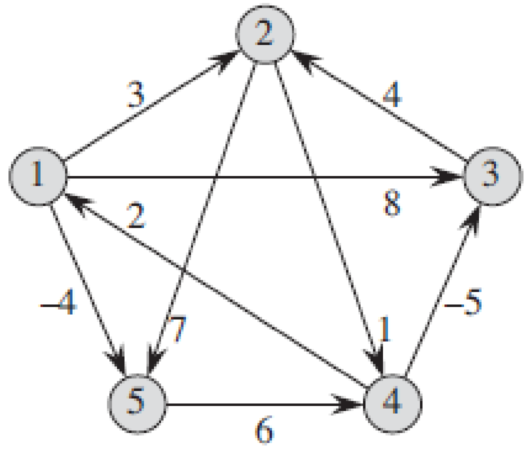
$$A^5[1][4] = \min(A^4[1][4], A^4[1][5] + A^4[5][4])$$
$$= \min(4, -4+6) = 2$$



A^5	1	2	3	4	5
1	0	~~3~~ 1	~~-1~~ -3	~~4~~ 2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

The shortest-paths optimal value is

A^5	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

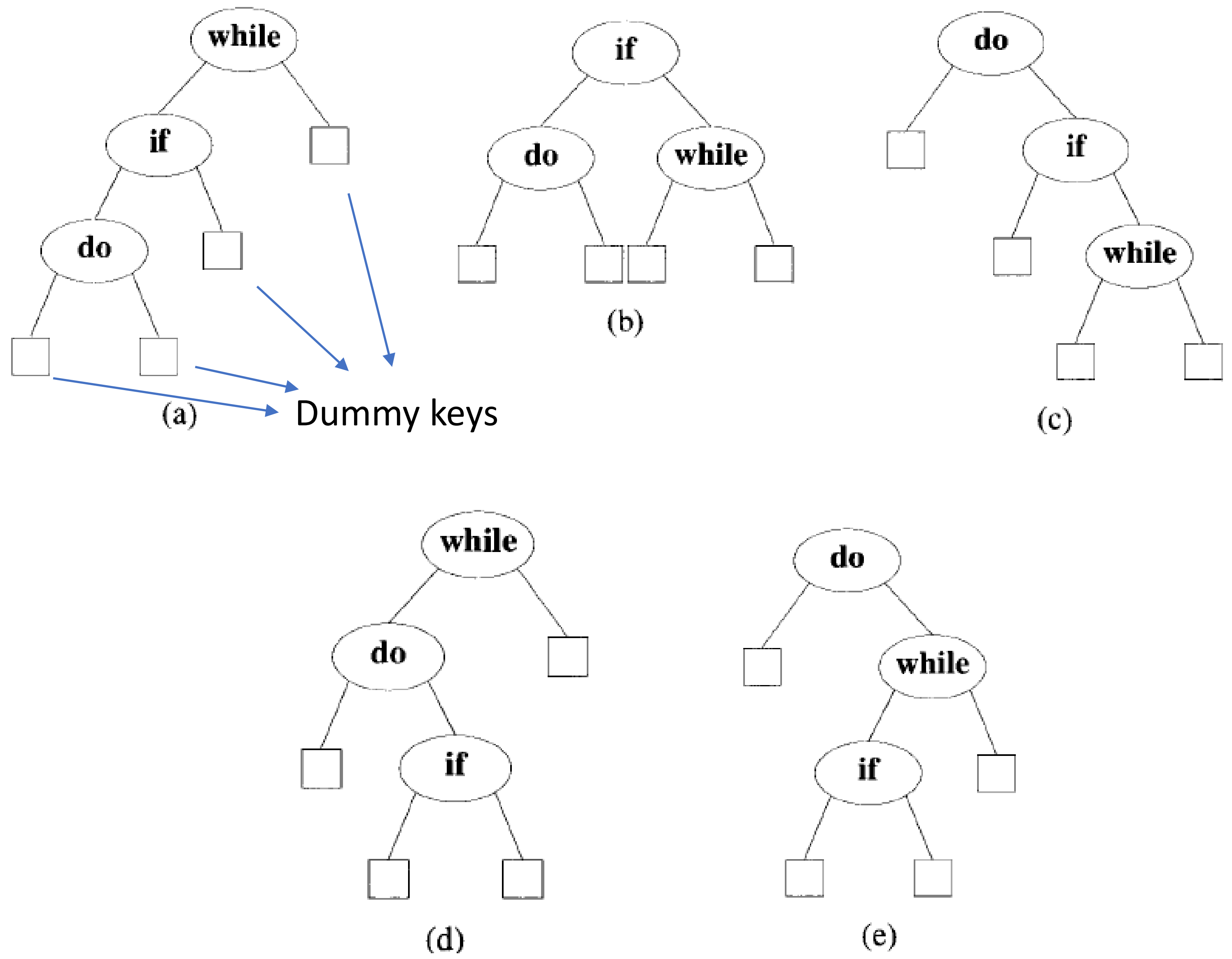


Optimal Binary Search Tree

- Given a sequence $K = \{a_1, a_2, a_3, \ldots a_n\}$ of n distinct keys in sorted order (so that $a_1 < a_2 < a_3 < \ldots < a_n$), and we wish to build a binary search tree from these keys. For each key a_i , we have a probability p_i that a search will be for a_i. Some searches may be for values not in K, and so we also have n + 1 “dummy keys”.

Given keys (a1,a2,a3)={do, if, while}

(p1,p2,p3) = (0.5,0.1,0.05)
(q0,q1,q2,q3) = (0.15,0.1,0.05,0.05)



Cost (tree a) = 2.65

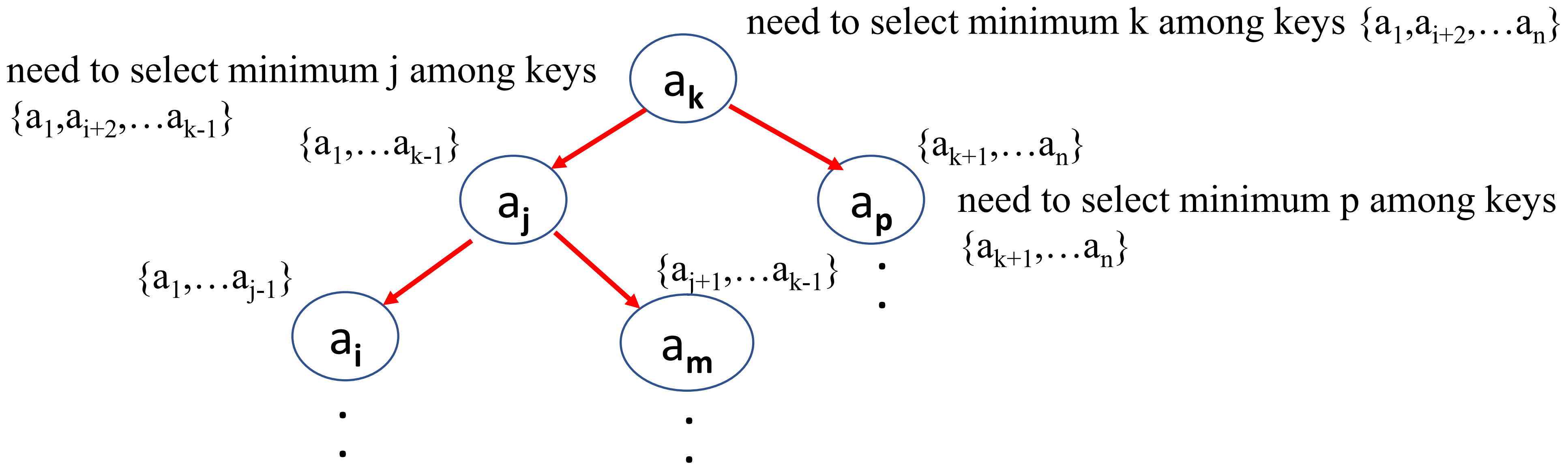
Cost (tree b) = 1.9

Cost (tree c) = 1.5

Cost (tree d) = 2.05

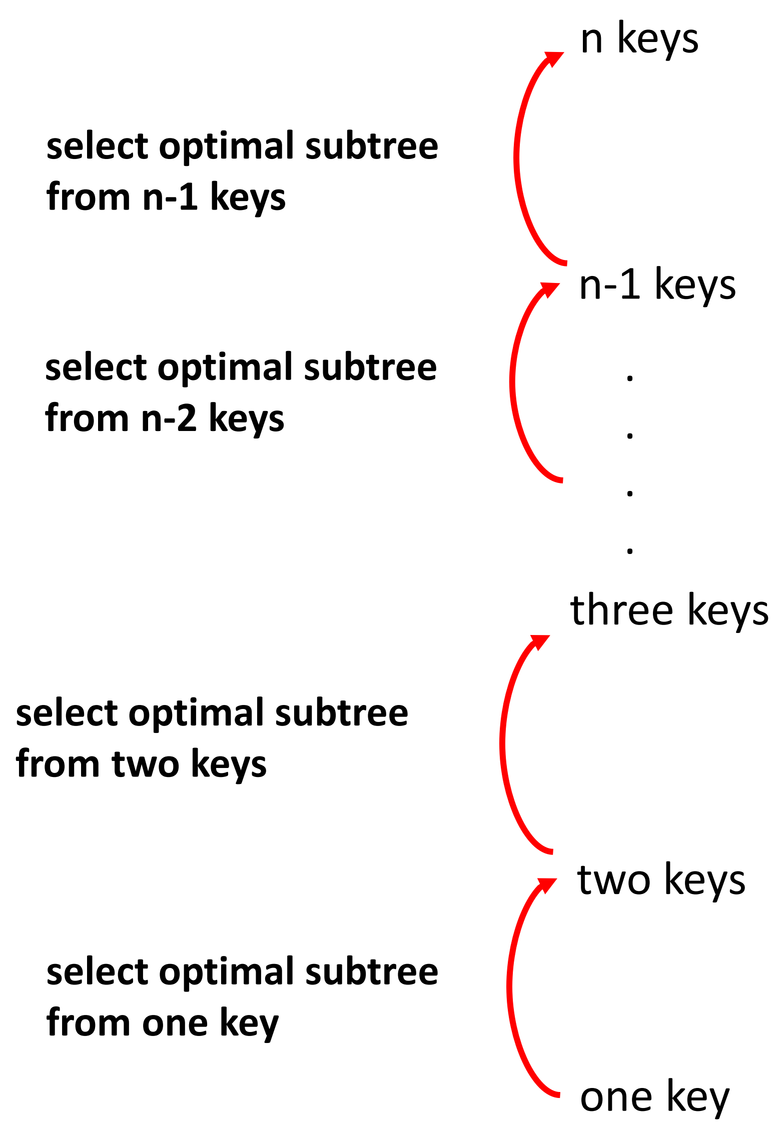
Cost (tree e) = 1.6

Optimal Binary Search Tree

Terminology	**Description**
C (i , j)	Cost of subtree t_{ij} that contains keys $\{a_{i+1},a_{i+2},\ldots a_j\}$ and dummy keys $\{d_i,d_{i+1},\ldots d_j\}$
w(i, j)	Sum of the probabilities (both p and q) of the keys $\{a_{i+1},a_{i+2,}\ \ldots..a_j\ \}$ and dummy keys $\{d_i,d_{i+1},\ldots d_j\}$
r(i, j)	Root of the subtree of keys $\{a_{i+1},a_{i+2},\ldots a_j\}$
Optimal substructure of the problem	we can construct an optimal solution to the problem from optimal solutions to subproblems.  need to select minimum k among keys $\{a_1,a_{i+2},\ldots a_n\}$ need to select minimum j among keys $\{a_1,a_{i+2},\ldots a_{k-1}\}$ a_k $\{a_1,\ldots a_{k-1}\}$ $\{a_{k+1},\ldots a_n\}$ a_j a_p need to select minimum p among keys $\{a_{k+1},\ldots a_n\}$ $\{a_1,\ldots a_{j-1}\}$ $\{a_{j+1},\ldots a_{k-1}\}$ a_i a_m

Optimal Binary Search Tree

Terminology	**Description**
Recurrence formula	$c(i,j) = \min_{i<k\leq j}\{c(i,k-1)+c(k,j)+p(k)+w(i,k-1)+w(k,j)\}$ $c(i,j) = \min_{i<k\leq j}\{c(i,k-1)+c(k,j)\}+w(i,j)$ $w(i,j) = p(j)+q(j)+w(i,j-1),$
Calculate the optimal value	c(0,n) Calculate the optimal value in a bottom-up method in an increasing order of m = (j- i) (m=1,2,3,….n)


n keys
select optimal subtree from n-1 keys
n-1 keys
select optimal subtree from n-2 keys
. . . .
three keys
select optimal subtree from two keys
two keys
select optimal subtree from one key
one key

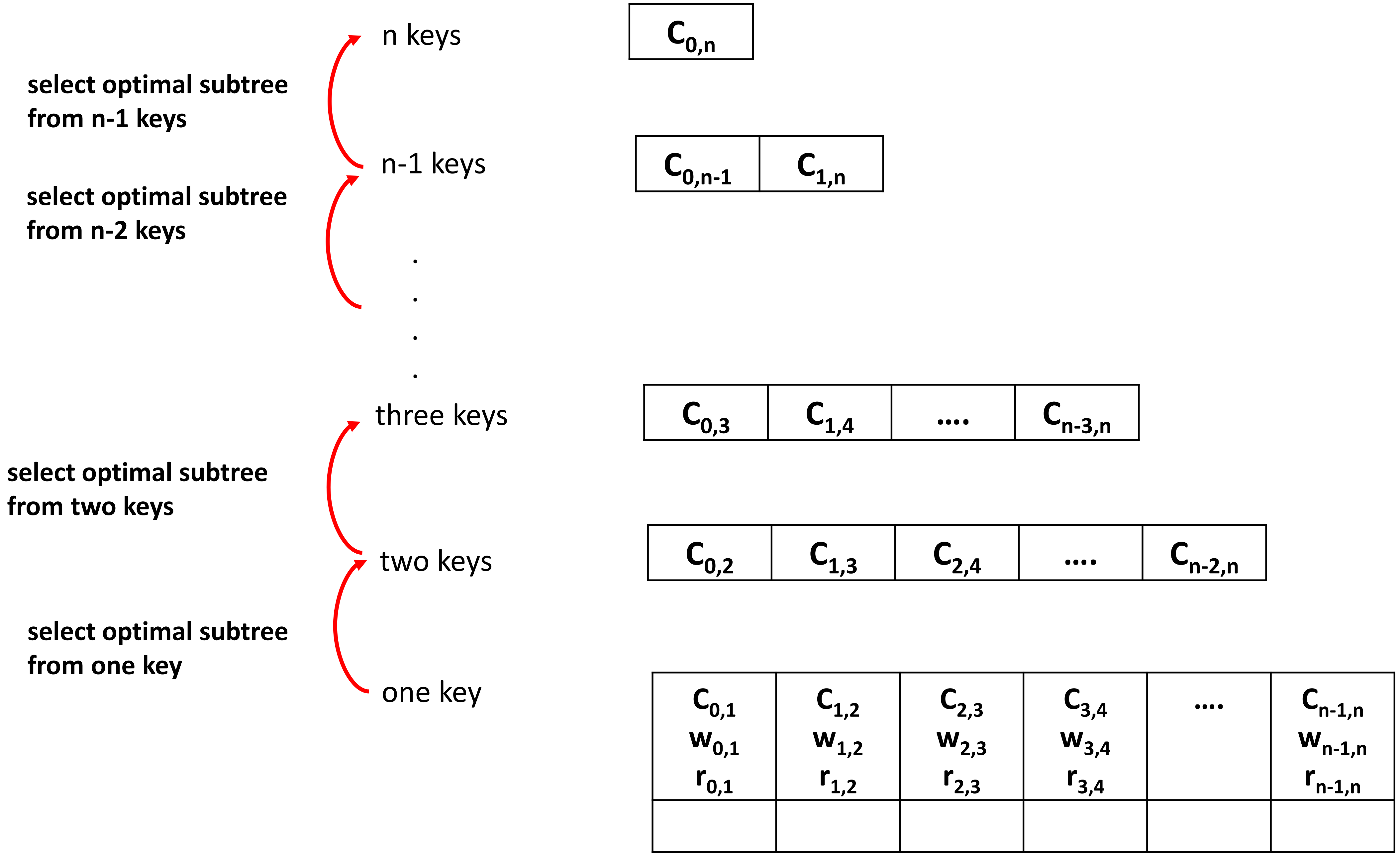
$w_{0,n}$ $C_{0,n}$ $r_{0,n}$

$w_{0,n-1}$ $C_{0,n-1}$ $r_{0,n-1}$	$w_{1,n}$ $C_{1,n}$ $r_{1,n}$

$w_{0,3}$ $C_{0,3}$ $r_{0,3}$	$w_{1,4}$ $C_{1,4}$ $r_{1,4}$	....	$w_{n-3,n}$ $C_{n-3,n}$ $r_{n-3,n}$

$w_{0,2}$ $C_{0,2}$ $r_{0,2}$	$w_{1,3}$ $C_{1,3}$ $r_{1,3}$	$w_{2,4}$ $C_{2,4}$ $r_{2,4}$	....	$w_{n-2,n}$ $C_{n-2,n}$ $r_{n-2,n}$

$w_{0,1}$ $C_{0,1}$ $r_{0,1}$	$w_{1,2}$ $C_{1,2}$ $r_{1,2}$	$w_{2,3}$ $C_{2,3}$ $r_{2,3}$	$w_{3,4}$ $C_{3,4}$ $r_{3,4}$	....	$w_{n-1,n}$ $C_{n-1,n}$ $r_{n-1,n}$


n keys
$C_{0,n}$
select optimal subtree from n-1 keys
n-1 keys
$C_{0,n-1}$ $C_{1,n}$
select optimal subtree from n-2 keys
. . . .
three keys
$C_{0,3}$ $C_{1,4}$ .... $C_{n-3,n}$
select optimal subtree from two keys
two keys
$C_{0,2}$ $C_{1,3}$ $C_{2,4}$ .... $C_{n-2,n}$
select optimal subtree from one key
one key
$C_{0,1}$ $w_{0,1}$ $r_{0,1}$
$C_{1,2}$ $w_{1,2}$ $r_{1,2}$
$C_{2,3}$ $w_{2,3}$ $r_{2,3}$
$C_{3,4}$ $w_{3,4}$ $r_{3,4}$
....
$C_{n-1,n}$ $w_{n-1,n}$ $r_{n-1,n}$

Let n =4 and (a1,a2,a3,a4)= {do, if, int , while }.
Let
p(1:4) = (3,3,1,1) and q(0:4)= (2,3,1,1,1)

$w(0,1) = p(1) + q(1) + w(0,0) = 3 + 3 + 2 = 8$

$c\ (0,1) = w\ (0,1) + \min\{c(0,0) + c(1,1)\}$

$= 8 + \min \{ 0 + 0\} = 8$

$r\ (0,1) = 1$

$w(1,2) = p(2) + q(2) + w(1,1) = 3 + 1 + 3 = 7$

$c\ (1,2) = w\ (1,2) + \min\{c(1,1) + c(2,2)\}$

$= 7 + \min \{ 0 + 0\} = 7$

$r\ (1,2) = 2$

$w(2,3) = p(3) + q(3) + w(2,2) = 1 + 1 + 1 = 3$

$c\ (2,3) = w\ (2,3) + \min\{c(2,2) + c(3,3)\}$

$= 3 + \min \{ 0 + 0\} = 3$

$r\ (2,3) = 3$

$w(3,4) = p(4) + q(4) + w(3,3) = 1 + 1 + 1 = 3$

$c\ (3,4) = w\ (3,4) + \min\{c(3,3) + c(4,4)\}$

$= 3 + \min \{ 0 + 0\} = 3$

$r\ (2,3) = 3$

j - i					
0	$w_{0,0} = 2$ $C_{0,0} = 0$ $r_{0,0} = 0$	$w_{1,1} = 3$ $C_{1,1} = 0$ $r_{1,1} = 0$	$w_{2,2} = 1$ $C_{2,2} = 0$ $r_{2,2} = 0$	$w_{3,3} = 1$ $C_{3,3} = 0$ $r_{3,3} = 0$	$w_{4,4} = 1$ $C_{4,4} = 0$ $r_{4,4} = 0$
1	$w_{0,1} = 8$ $C_{0,1} = 8$ $r_{0,1} = 1$	$w_{1,2} = 7$ $C_{1,2} = 7$ $r_{1,2} = 2$	$w_{2,3} = 3$ $C_{2,3} = 3$ $r_{2,3} = 3$	$w_{3,4} = 3$ $C_{3,4} = 3$ $r_{3,4} = 4$	
2	$w_{0,2}$ $C_{0,2}$ $r_{0,2}$	$w_{1,3}$ $C_{1,3}$ $r_{1,3}$	$w_{2,4}$ $C_{2,4}$ $r_{2,4}$		
3	$w_{0,3}$ $C_{0,3}$ $r_{0,3}$	$w_{1,4}$ $C_{1,4}$ $r_{1,4}$			
4	$w_{0,4}$ $C_{0,4}$ $r_{0,4}$				

Let n =4 and (a1,a2,a3,a4)= {do, if, int , while }.
Let
p(1:4) = (3,3,1,1) and q(0:4)= (2,3,1,1,1)

$w(0,2) = p(2) + q(2) + w(0,1) = 3 + 1 + 8 = 12$

$c(0,2) = w(0,2) + \min\{c(0,0) + c(1,2), c(0,1) + c(2,2)\}$

$= 12 + \min\{0 + 7, 8 + 0\} = 19$

$r(0,2) = 1$

$w(1,3) = p(3) + q(3) + w(1,2) = 1 + 1 + 7 = 9$

$c(1,3) = w(1,3) + \min\{c(1,1) + c(2,3), c(1,2) + c(3,3)\}$

$= 7 + \min\{0 + 3, 7 + 0\} = 12$

$r(1,2) = 2$

$w(2,4) = p(4) + q(4) + w(2,3) = 1 + 1 + 3 = 5$

$c(2,4) = w(2,4) + \min\{c(2,2) + c(3,4), c(2,3) + c(4,4)\}$

$= 5 + \min\{0 + 3, 3 + 0\} = 8$

$r(2,4) = 3$

j - i					
0	$w_{0,0} = 2$ $C_{0,0} = 0$ $r_{0,0} = 0$	$w_{1,1} = 3$ $C_{1,1} = 0$ $r_{1,1} = 0$	$w_{2,2} = 1$ $C_{2,2} = 0$ $r_{2,2} = 0$	$w_{3,3} = 1$ $C_{3,3} = 0$ $r_{3,3} = 0$	$w_{4,4} = 1$ $C_{4,4} = 0$ $r_{4,4} = 0$
1	$w_{0,1} = 8$ $C_{0,1} = 8$ $r_{0,1} = 1$	$w_{1,2} = 7$ $C_{1,2} = 7$ $r_{1,2} = 2$	$w_{2,3} = 3$ $C_{2,3} = 3$ $r_{2,3} = 3$	$w_{3,4} = 3$ $C_{3,4} = 3$ $r_{3,4} = 4$	
2	$w_{0,2} = 12$ $C_{0,2} = 19$ $r_{0,2} = 1$	$w_{1,3} = 9$ $C_{1,3} = 12$ $r_{1,3} = 2$	$w_{2,4} = 5$ $C_{2,4} = 8$ $r_{2,4} = 3$		
3	$w_{0,3}$ $C_{0,3}$ $r_{0,3}$	$w_{1,4}$ $C_{1,4}$ $r_{1,4}$			
4	$w_{0,4}$ $C_{0,4}$ $r_{0,4}$				

Let n =4 and (a1,a2,a3,a4)= {do, if, int , while }.
Let
p(1:4) = (3,3,1,1) and q(0:4)= (2,3,1,1,1)

$w(0,3) = p(3) + q(3) + w(0,2) = 1 + 1 + 12 = 14$

$c(0,3) = w(0,3) + \min\{c(0,0) + c(1,3),$
$\qquad c(0,1) + c(2,3),$
$\qquad c(0,2) + c(3,3)\}$
$= 14 + \min\{0 + 12, 8 + 3, 19+0\} = 25$

$r(0,3) = 2$

$w(1,4) = p(4) + q(4) + w(1,3) = 1 + 1 + 9 = 11$
$c(1,4) = w(1,4) + \min\{c(1,1) + c(2,4),$
$\qquad c(1,2) + c(3,4),$
$\qquad c(1,3) + c(4,4)\}$
$= 11 + \min\{0 + 8, 7 + 3, 12+0\} = 19$
$r(1,4) = 2$

j - i					
0	$w_{0,0} = 2$ $C_{0,0} = 0$ $r_{0,0} = 0$	$w_{1,1} = 3$ $C_{1,1} = 0$ $r_{1,1} = 0$	$w_{2,2} = 1$ $C_{2,2} = 0$ $r_{2,2} = 0$	$w_{3,3} = 1$ $C_{3,3} = 0$ $r_{3,3} = 0$	$w_{4,4} = 1$ $C_{4,4} = 0$ $r_{4,4} = 0$
1	$w_{0,1} = 8$ $C_{0,1} = 8$ $r_{0,1} = 1$	$w_{1,2} = 7$ $C_{1,2} = 7$ $r_{1,2} = 2$	$w_{2,3} = 3$ $C_{2,3} = 3$ $r_{2,3} = 3$	$w_{3,4} = 3$ $C_{3,4} = 3$ $r_{3,4} = 4$	
2	$w_{0,2} = 12$ $C_{0,2} = 19$ $r_{0,2} = 1$	$w_{1,3} = 9$ $C_{1,3} = 12$ $r_{1,3} = 2$	$w_{2,4} = 5$ $C_{2,4} = 8$ $r_{2,4} = 3$		
3	$w_{0,3} = 14$ $C_{0,3} = 25$ $r_{0,3} = 2$	$w_{1,4} = 11$ $C_{1,4} = 19$ $r_{1,4} = 2$			
4	$w_{0,4}$ $C_{0,4}$ $r_{0,4}$				

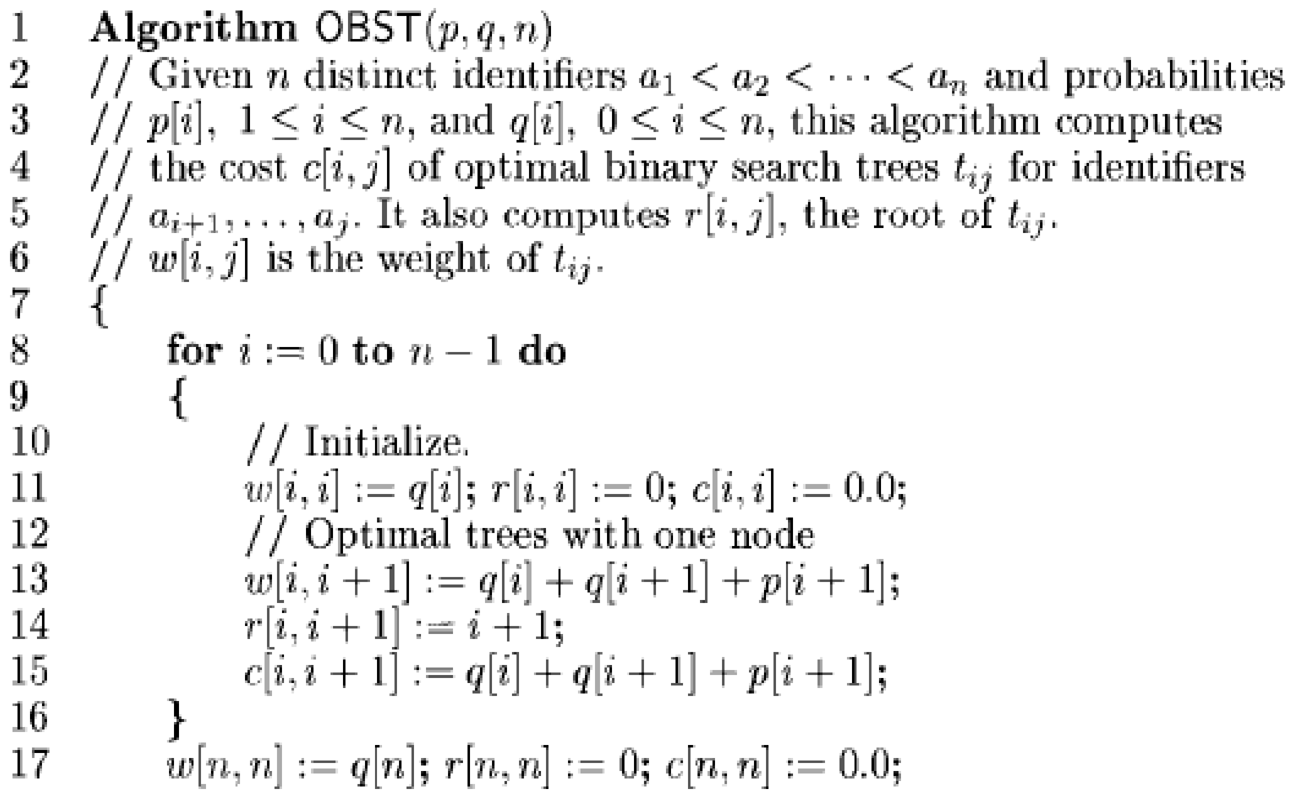
Let n =4 and (a1,a2,a3,a4)= {do, if, int , while }.
Let
p(1:4) = (3,3,1,1) and q(0:4)= (2,3,1,1,1)

$w(0,4) = p(4) + q(4) + w(0,3) = 1 + 1 + 14 = 16$

$$\begin{aligned} c(0,4) &= w(0,4) + \min\{c(0,0) + c(1,4), \\ &\qquad c(0,1) + c(2,4), \\ &\qquad c(0,2) + c(3,4), \\ &\qquad c(0,3) + c(4,4)\} \\ &= 16 + \min\{0 + 19, 8 + 8, 19+3, 25+0\} = 32 \end{aligned}$$

$r(0,4) = 2$

j - i					
0	$w_{0,0} = 2$ $C_{0,0} = 0$ $r_{0,0} = 0$	$w_{1,1} = 3$ $C_{1,1} = 0$ $r_{1,1} = 0$	$w_{2,2} = 1$ $C_{2,2} = 0$ $r_{2,2} = 0$	$w_{3,3} = 1$ $C_{3,3} = 0$ $r_{3,3} = 0$	$w_{4,4} = 1$ $C_{4,4} = 0$ $r_{4,4} = 0$
1	$w_{0,1} = 8$ $C_{0,1} = 8$ $r_{0,1} = 1$	$w_{1,2} = 7$ $C_{1,2} = 7$ $r_{1,2} = 2$	$w_{2,3} = 3$ $C_{2,3} = 3$ $r_{2,3} = 3$	$w_{3,4} = 3$ $C_{3,4} = 3$ $r_{3,4} = 4$	
2	$w_{0,2} = 12$ $C_{0,2} = 19$ $r_{0,2} = 1$	$w_{1,3} = 9$ $C_{1,3} = 12$ $r_{1,3} = 2$	$w_{2,4} = 5$ $C_{2,4} = 8$ $r_{2,4} = 3$		
3	$w_{0,3} = 14$ $C_{0,3} = 25$ $r_{0,3} = 2$	$w_{1,4} = 11$ $C_{1,4} = 19$ $r_{1,4} = 2$			
4	$w_{0,4} = 16$ $C_{0,4} = 32$ $r_{0,4} = 2$				



```
1   Algorithm OBST($p, q, n$)
2   // Given $n$ distinct identifiers $a_1 < a_2 < \cdots < a_n$ and probabilities
3   // $p[i]$, $1 \le i \le n$, and $q[i]$, $0 \le i \le n$, this algorithm computes
4   // the cost $c[i, j]$ of optimal binary search trees $t_{ij}$ for identifiers
5   // $a_{i+1}, \ldots, a_j$. It also computes $r[i, j]$, the root of $t_{ij}$.
6   // $w[i, j]$ is the weight of $t_{ij}$.
7   {
8       for $i := 0$ to $n - 1$ do
9       {
10          // Initialize.
11          $w[i, i] := q[i]$; $r[i, i] := 0$; $c[i, i] := 0.0$;
12          // Optimal trees with one node
13          $w[i, i+1] := q[i] + q[i+1] + p[i+1]$;
14          $r[i, i+1] := i + 1$;
15          $c[i, i+1] := q[i] + q[i+1] + p[i+1]$;
16      }
17      $w[n, n] := q[n]$; $r[n, n] := 0$; $c[n, n] := 0.0$;
```

```
18.  for m:=2 to n do
19.      for i:=0 to n-m do // Find Optimal trees with m node
20.      {
21.              j := i+m;
22.              w[i,j]:= w[i,j-1]+ p[j] +q[j];
23.              minval =  999 ;                              n*(n-m) *n
24.              for $l$ :=i+1 to j {
25.                  if c[i,$l$ -1]+c[$l$,j] < minval {
26.                      minval=c[i,$l$ -1]+c[$l$ , j];
27.                      k = $l$;
28.                  }
29.              }
30.              c[i,j]:= w[i , j]+c[i,k-1]+c[k];
31.              r[I,j]:=k;
32.      }
33.      write (c[0,n],w[0,n],r[0,n]);
34. }
```

Time complexity :O(n^3)

1. Use function OBST (Algorithm 5.5) to compute $w(i,j)$, $r(i,j)$, and $c(i,j)$, $0 \leq i < j \leq 4$, for the identifier set $(a_1, a_2, a_3, a_4) =$ (**cout, float, if, while**) with $p(1) = 1/20$, $p(2) = 1/5$, $p(3) = 1/10$, $p(4) = 1/20$, $q(0) = 1/5$, $q(1) = 1/10$, $q(2) = 1/5$, $q(3) = 1/20$, and $q(4) = 1/20$. Using the $r(i,j)$'s, construct the optimal binary search tree.

0/1 knapsack problem

- Given n objects a knapsack or bag. Object *i* has a weight w_i and the knapsack has a capacity m. If an object *i* is placed into the knapsack, then a profit p_i is earned. The ***objective*** is to obtain a filling of the knapsack that ***maximizes the total profit earned*** . Since the knapsack capacity is m, we require the total weight of all chosen objects to be at most m. Formally, the problem can be stated as

$$\text{maximize} \sum_{1 \le i \le n} p_i x_i$$

$$\text{subject to} \sum_{1 \le i \le n} w_i x_i \le m$$

Where $x_i =$ 0 or 1

The profits and weights are positive real numbers

Terminology	**Description**
$f_{\mathbf{n}}$ (m)	The value of optimal solution to knapsack(1,n,m) where n is the number of objects and m is the knapsack capacity
Optimal substructure of the problem	<table><tr><th>Items</th><th>Optimal value</th><th>Description</th></tr><tr><td>0</td><td>$f_{\mathbf{0}}$ (1), $f_{\mathbf{0}}$ (2), $f_{\mathbf{0}}$ (3),… $f_{\mathbf{0}}$ (m)</td><td>Initialization</td></tr><tr><td>1</td><td>$f_{\mathbf{1}}$ (1), $f_{\mathbf{1}}$ (2), $f_{\mathbf{1}}$ (3),… $f_{\mathbf{1}}$ (m)</td><td>Optimal value of including item 1</td></tr><tr><td>2</td><td>$f_{\mathbf{2}}$ (1), $f_{\mathbf{2}}$ (2), $f_{\mathbf{2}}$ (3),… $f_{\mathbf{2}}$ (m)</td><td>Optimal value of including all possibilities of item 1,and 2 i.e., {1},{2},{1,2}</td></tr><tr><td>3</td><td>$f_{\mathbf{3}}$ (1), $f_{\mathbf{3}}$ (2), $f_{\mathbf{3}}$ (3),… $f_{\mathbf{3}}$ (m)</td><td>Optimal value of including all possibilities of item 1, 2, and 3</td></tr><tr><td>.</td><td>.</td><td></td></tr><tr><td>n</td><td>$f_{\mathbf{n}}$ (1), $f_{\mathbf{n}}$ (2), $f_{\mathbf{n}}$ (3),… $f_{\mathbf{n}}$ (m)</td><td>Optimal value of including all possibilities of item 1, 2, … n</td></tr></table>
Recurrence formula	$f_{\mathbf{n}}(m) = \max \{ f_{\mathbf{n-1}}(m) , f_{\mathbf{n-1}}(m-w_{\mathbf{n}}) + p_{\mathbf{n}} \}$
Calculate the optimal value	$f_{\mathbf{n}}$ (m) where n is number of objects and m is the knapsack capacity

- A solution to the knapsack problem can be obtained by making a sequence of decisions on the variables $x_1, x_2, x_3, \ldots. x_n$. A decision on variable x_i involves determining which of the values *0 or 1* is to be assigned to it. Let us assume that decisions on the x_i are made in the order $x_n, x_{n-1}, .., x_2, x_1$. Following a decision on x_n , we may be in one of two possible states :
  - the capacity remaining in the knapsack is m and no profit has accrued or
  - the capacity remaining is $m - w_n$ and a profit of p_n has accrued .
- It is clear that the remaining decisions $x_{n-1}, \ldots x_1$ must be optimal with respect to the problem state resulting from the decision on x_n. Otherwise , $x_n, \ldots. x_1$ will not be optimal

Consider the knapsack instance n=3 and knapsack capacity m=6
(w1,w2,w3)= (2,3,4), (p1,p2,p3)= (11,15,12) .

```
if $w_i > w$
    $K[i,w]=K[i-1,w]$
else
    $K[i,w] = \max \{ K[i-1,w], K[i-1,w-w_i] + p_i \}$
```

$$
\begin{aligned}
K[1,2] &= \max\{ K[1-1,2], K[1-1, 2-2] + 11 \} \\
&= \max\{ K[0,2], K[0, 0] + 11 \} \\
&= \max\{ 0,11\} = 11
\end{aligned}
$$

$$
\begin{aligned}
K[1,3] &= \max\{ K[1-1,3], K[1-1, 3-2] + 11 \} \\
&= \max\{ K[0,3], K[0, 1] + 11 \} \\
&= \max\{ 0,11\} = 11
\end{aligned}
$$

$$
\begin{aligned}
K[1,4] &= \max\{ K[1-1,4], K[1-1, 4-2] + 11 \} \\
&= \max\{ K[0,4], K[0, 2] + 11 \} \\
&= \max\{ 0,11\} = 11
\end{aligned}
$$

$$
\begin{aligned}
K[1,5] &= \max\{ K[1-1,5], K[1-1, 5-2] + 11 \} \\
&= \max\{ K[0,5], K[0, 3] + 11 \} \\
&= \max\{ 0,11\} = 11
\end{aligned}
$$

$$
\begin{aligned}
K[1,6] &= \max\{ K[1-1,6], K[1-1, 6-2] + 11 \} \\
&= \max\{ K[0,5], K[0, 4] + 11 \} \\
&= \max\{ 0,11\} = 11
\end{aligned}
$$

K

			w						
p_i	w_i	i	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0	0	0
11	2	1	0	0	11	11	11	11	11
15	3	2	0						
12	4	3	0						

Consider the knapsack instance n=3 and knapsack capacity m=6
(w1,w2,w3)= (2,3,4), (p1,p2,p3)= (11,15,12) .

```
if $w_i > w$
    $K[i,w]=K[i-1,w]$
else
    $K[i,w] = \max \{ K[i-1,w], K[i-1,w-w_i] + p_i \}$
```

$$\begin{aligned} K[2,3] &= \max\{ K[2-1,3], K[2-1, 3-3] + p_2 \} \\ &= \max\{ K[1,3], K[1, 0] + 15 \} \\ &= \max\{ 11,15\} = 15 \end{aligned}$$

$$\begin{aligned} K[2,4] &= \max\{ K[2-1,4], K[2-1, 4-3] + p_2 \} \\ &= \max\{ K[1,4], K[1, 1] + 15 \} \\ &= \max\{ 11,15\} = 15 \end{aligned}$$

$$\begin{aligned} K[2,5] &= \max\{ K[2-1,5], K[2-1, 5-3] + p_2 \} \\ &= \max\{ K[1,5], K[1, 2] + 15 \} \\ &= \max\{ 11,26\} = 26 \end{aligned}$$

$$\begin{aligned} K[2,6] &= \max\{ K[2-1,6], K[2-1, 6-2] + p_2 \} \\ &= \max\{ K[1,6], K[1, 4] + 15 \} \\ &= \max\{ 11,26\} = 26 \end{aligned}$$

K

			w						
p_i	w_i	i	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0	0	0
11	2	1	0	0	11	11	11	11	11
15	3	2	0	0	11	15	15	26	26
12	4	3	0						

0

Consider the knapsack instance n=3 and knapsack capacity m=6
(w1,w2,w3)= (2,3,4), (p1,p2,p3)= (11,15,12) .

```
if $w_i > w$
    $K[i,w]=K[i-1,w]$
else
    $K[i,w] = \max\{ K[i-1,w], K[i-1,w-w_i] + p_i \}$
```

$K[3,4] = \max\{ K[3-1,4], K[3-1, 4-4] + p_3 \}$
$= \max\{ K[2,4], K[2, 0] + 12 \}$
$= \max\{ 15,12\} = 15$

$K[3,5] = \max\{ K[3-1,5], K[3-1, 5-4] + p_3 \}$
$= \max\{ K[2,5], K[2, 1] + 12 \}$
$= \max\{ 26,12\} = 26$

$K[3,6] = \max\{ K[3-1,6], K[3-1, 6-4] + p_3 \}$
$= \max\{ K[2,6], K[2, 2] + 12 \}$
$= \max\{ 26,23\} = 26$

K

			w						
p_i	w_i	i	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0	0	0
11	2	1	0	0	11	11	11	11	11
15	3	2	0	0	11	15	15	26	26
12	4	3	0	0	11	15	15	26	26

0

Consider the knapsack instance n=3 and knapsack capacity m=6 (w1,w2,w3)= (2,3,4), (p1,p2,p3)= (11,15,12) .

K

			w						
p_i	w_i	i	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0	0	0
11	2	1	0	0	11	11	11	11	11
15	3	2	0	0	11	15	15	26	26
12	4	3	0	0	11	15	15	26	26

X1=1 since 11 doesn't belong to previous row.

X2=1 since 26 doesn't belong to previous row.

X3=0 since 26 belongs to previous row i.e., profit is not obtained because of item 4

The solution vector X is
X= (x_1,x_2,x_3)=(1,1,0)

Analysis

- The table filling method will take complexity of O(nW).
- For suppose if the weights of the objects are real numbers, then it is very difficult to calculate for all possible weights between i ≤ w ≤ i+1. so, this method is inefficient.

			w								
p_i	w_i	i	0	1	1.1	1.2	… …	2	2.01	…..	……
0	0	0	0	0	0	0	….	0	0		
11	2.3	1	0								
15	3.25	2	0								
12	4.5	3	0								

- So, we use another method known as ordered set with merging and purging

0/1 knapsack (using Merging and Purging)- Method-2

Consider the knapsack instance n=3 and knapsack capacity m=6
(w1,w2,w3)= (2,3,4), (p1,p2,p3)= (1,2,5) .

(P,W) –profit and weight obtained not including item 1

(P,W) –profit and weight obtained by including item 1

(P,W) –profit and weight obtained by including item 1 and 2 possibilities

(P,W) –profit and weight obtained by including item 1, 2,and 3 possibilities

$$\begin{tikzcd}
S^0 = \{(0,0)\} \arrow[d] & S_1^0 = \{(1,2)\} \arrow[dl] \\
S^1 = \{(0,0),(1,2)\} \arrow[d] & S_1^1 = \{(2,3),(3,5)\} \arrow[dl] \\
S^2 = \{(0,0),(1,2),(2,3),(3,5)\} \arrow[d] & S_1^2 = \{(5,4),(6,6),(7,7),(8,9)\} \arrow[dl] \\
S^3 = \{(0,0),(1,2),(2,3),(5,4),(6,6),(7,7),(8,9)\} &
\end{tikzcd}$$

(3,5) is purged because it is dominating tuple

1. Generate the sets $S^i, 0 \leq i \leq 4$ (Equation 5.16), when $(w_1, w_2, w_3, w_4) = (10, 15, 6, 9)$ and $(p_1, p_2, p_3, p_4) = (2, 5, 8, 1)$.

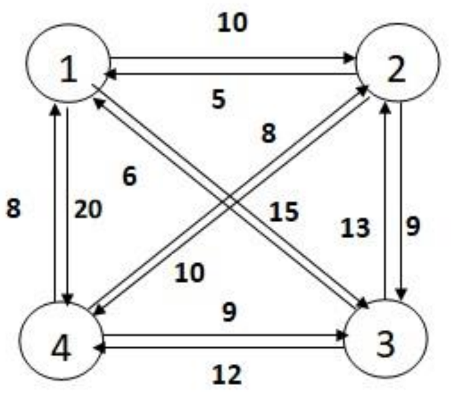
Traveling SalesPerson(TSP) Problem

- Given a graph G= (V,E) and a tour is a directed simple cycle that includes every vertex in V.
- Solving the TSP problem is to find a tour of the given graph with minimum cost.
- TSP is a permutation problem, i.e., if a Graph G contains n-vertices then the possible tours are n! .
- By applying Dynamic Programming technique ,the time complexity of TSP problem reduces from **$O(n!)$** to **$O(n^2 2^n)$.**
- Let G=(V,E) be a directed graph with edge cost matrix C.

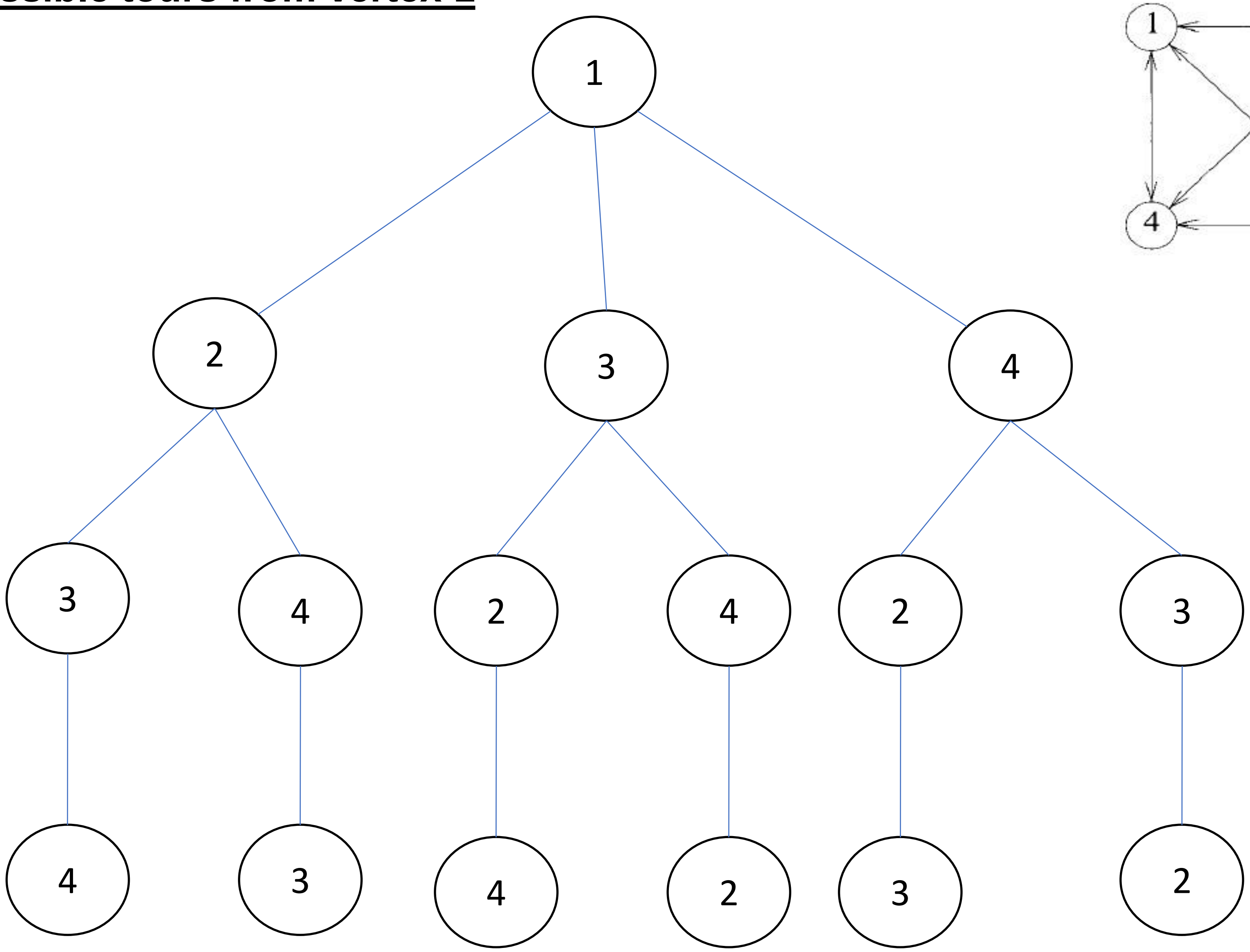
  The cost matrix is constructed as follows

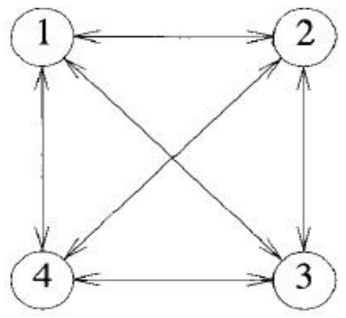
$$c_{ij} = 0 \quad \text{if } i=j$$
$$= \text{cost of the edge } \langle i,j\rangle \in E$$
$$= \infty \quad \langle i,j\rangle \notin E$$

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 10 & 15 & 20 \\ 2 & 5 & 0 & 9 & 10 \\ 3 & 6 & 13 & 0 & 12 \\ 4 & 8 & 8 & 9 & 0 \end{array}$$



Possible tours from vertex 1





TSP problem – recursive formulation

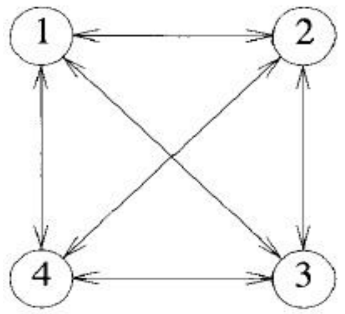
- Let us assume the tour starts and ends at vertex 1.(In general, we can starts and ends with any vertex)
- Let g(i,S) be the length of a shortest path starting at vertex i, going through all vertices in S, and terminating at vertex 1.

$$g(1, V - \{1\}) = \min_{2 \le k \le n} \{c_{1k} + g(k, V - \{1, k\})\}$$

The optimal length of the tour starts at vertex 1 and goes through all vertices V excluding vertex 1 i.e., S={2,3,4,…n}. In general , for any vertex i

$$g(i, S) = \min_{j \in S}\{c_{ij} + g(j, S - \{j\})\}$$

Problem Formulation



$$g(1, V - \{1\}) = \min_{2 \le k \le n} \{c_{1k} + g(k, V - \{1, k\})\}$$

- The optimal length tour of the graph G is represented as g(1,{2,3,4})

***g (1,{2,3,4})** - form vertex 1 going to all vertices {2,3,4} exactly once in any order and back to vertex 1*

g(1,{2,3,4}) = min { c_{12} + g (2,{3,4}) , c_{13} + g (3,{2,4}), c_{14} + g (4,{2,3})}

g (2,{3,4})** - from vertex 2 going to all vertices {3,4} exactly once in any order and similarly for **g (3,{2,4})** and **g (4,{2,3})

g (2,{3,4}) = min {{ c_{23} + g (3,{4}) , c_{24} + g (4,{3}) }
g (3,{2,4}) = min {{ c_{32} + g (2,{4}) , c_{34} + g (4,{2}) }
g (4,{2,3}) = min {{ c_{42} + g (2,{3}) , c_{43} + g (3,{2}) }

g (3,{4})** - from vertex 3 going to vertex {4} exactly once and similarly for **g (3,{2}), g (4,{2}), g (4,{3}) , g (3,{2}),** and **g (3,{4}).

g (3,{4}) = min { c_{34} + g (4,Φ)} **g(4,{3})= min {c_{43} + g (3,Φ)}**
g (2,{4}) = min { c_{24} + g (4,Φ)} **g(4,{2})= min {c_{42} + g (2,Φ)}**
g (2,{3}) = min { c_{23} + g (3,Φ)} **g(3,{2})= min {c_{32} + g (2,Φ)}**

g (4,Φ) = c_{41} **g(3,Φ) = c_{31}** **g(2,Φ) = c_{21}**

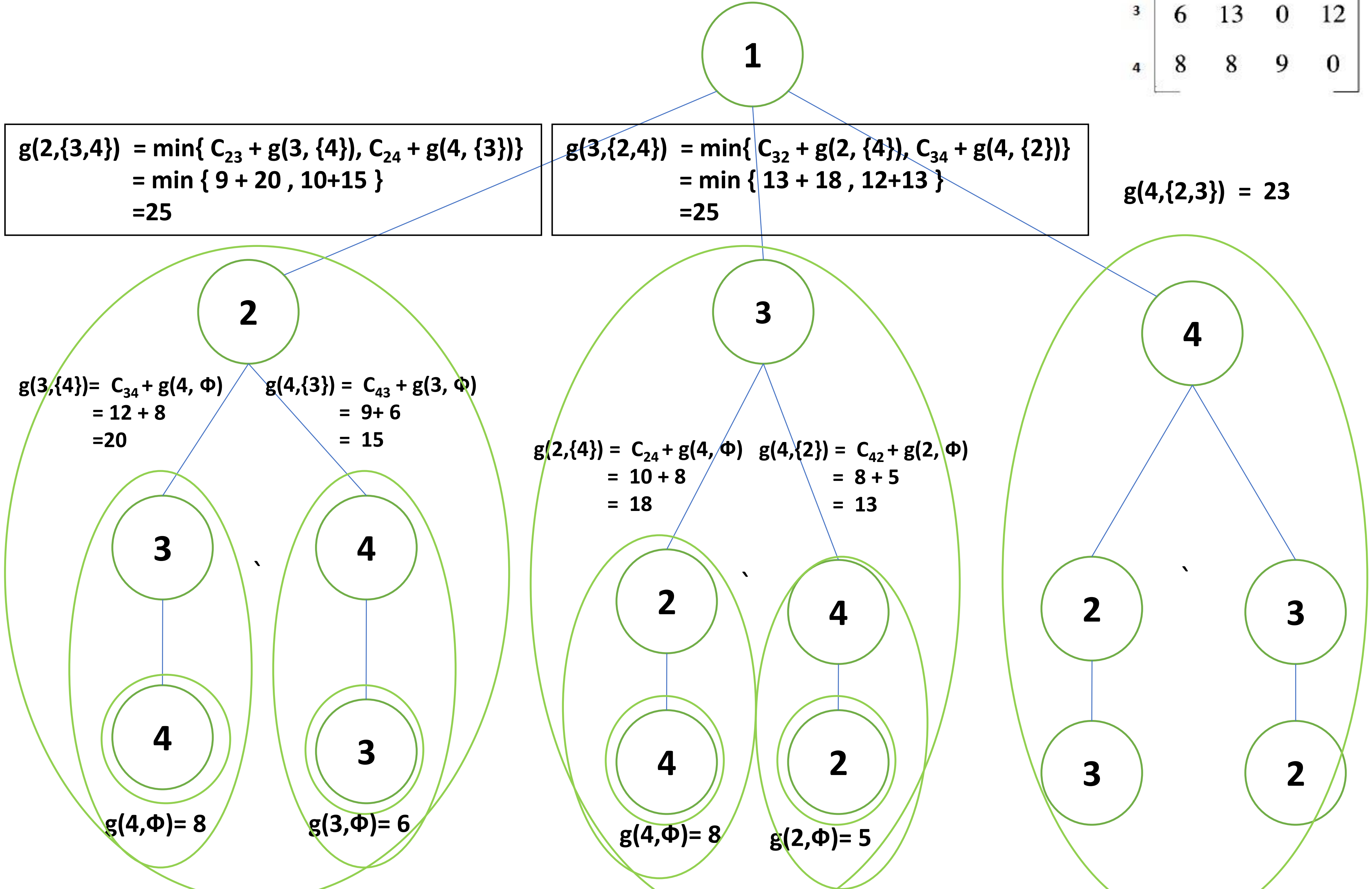
$$
\begin{aligned}
g(1,\{2,3,4\}) &= \min\{ C_{12} + g(2,\{3,4\}),\ C_{13} + g(3,\{2,4\}),\ C_{14} + g(4,\{2,3\})\} \\
&= \min\{ 10+25,\ 15+25,\ 20+23 \} \\
&= 35
\end{aligned}
$$

$$
\begin{array}{c|cccc}
 & 1 & 2 & 3 & 4 \\
\hline
1 & 0 & 10 & 15 & 20 \\
2 & 5 & 0 & 9 & 10 \\
3 & 6 & 13 & 0 & 12 \\
4 & 8 & 8 & 9 & 0
\end{array}
$$

$$
\begin{aligned}
g(2,\{3,4\}) &= \min\{ C_{23} + g(3,\{4\}),\ C_{24} + g(4,\{3\})\} \\
&= \min\{ 9 + 20,\ 10+15 \} \\
&= 25
\end{aligned}
$$

$$
\begin{aligned}
g(3,\{2,4\}) &= \min\{ C_{32} + g(2,\{4\}),\ C_{34} + g(4,\{2\})\} \\
&= \min\{ 13 + 18,\ 12+13 \} \\
&= 25
\end{aligned}
$$

$g(4,\{2,3\}) = 23$



$$
\begin{aligned}
g(3,\{4\}) &= C_{34} + g(4,\Phi) \\
&= 12 + 8 \\
&= 20
\end{aligned}
$$

$$
\begin{aligned}
g(4,\{3\}) &= C_{43} + g(3,\Phi) \\
&= 9 + 6 \\
&= 15
\end{aligned}
$$

$$
\begin{aligned}
g(2,\{4\}) &= C_{24} + g(4,\Phi) \\
&= 10 + 8 \\
&= 18
\end{aligned}
$$

$$
\begin{aligned}
g(4,\{2\}) &= C_{42} + g(2,\Phi) \\
&= 8 + 5 \\
&= 13
\end{aligned}
$$

$g(4,\Phi) = 8$ $g(3,\Phi) = 6$ $g(4,\Phi) = 8$ $g(2,\Phi) = 5$

$g(1,\{2,3,4\}) = \min \{ C_{12} + g(2,\{3,4\}), C_{13} + g(3,\{2,4\}), C_{14} + g(4,\{2,3\})\}$
$= \min \{ 10+25, 15 + 25 , 20+23 \}$
$= 35$

$g(2,\{3,4\}) = \min\{ C_{23} + g(3, \{4\}), C_{24} + g(4, \{3\})\}$
$= \min \{ 9 + 20 , 10+15 \}$
$= 25$

$g(3,\{2,4\}) = \min\{ C_{32} + g(2, \{4\}), C_{34} + g(4, \{2\})\}$
$= \min \{ 13 + 18 , 12+13 \}$
$= 25$

$g(4,\{2,3\} = \min\{ C_{42} + g(2, \{3\}), C_{43} + g(3, \{2\})\}$
$= \min \{ 8 + 15 , 9+18 \}$
$= 23$

$g(3,\{4\}) = C_{34} + g(4, \Phi)$
$= 12 + 8 = 20$
$g(4,\{3\}) = C_{43} + g(3, \Phi)$
$= 9 + 6 = 15$
$g(2,\{4\}) = C24 + g(4, \Phi)$
$= 10 + 8 = 18$

$g(4,\{2\}) = C_{42} + g(2, \Phi)$
$= 8 + 5 = 13$
$g(2,\{3\}) = C_{23} + g(3, \Phi)$
$= 9 + 6 = 15$
$g(3,\{2\}) = C_{32} + g(2, \Phi)$
$= 13 + 5 = 18$

$g(4,\Phi) = 8$ $g(3,\Phi) = 15$ $g(2,\Phi) = 18$

Optimal tour

J(1,{2,3,4}) =2

Path: 1-- >2

J(2,{3,4}) = 4

Path: 1-- >2 --> 4

J(4,{3}) = 3

Path: 1-- >2 --> 4 --> 3

Path: 1-- >2 --> 4 --> 3 --> 1

Applications of Traveling Salesperson Problem

1. A post van is to pick up mails from mail boxes located at different sites .

   It can be modelled as TSP problem by considering sites as cities and distance between sites(in kms) as edge cost. The route taken by the postal van is a **tour**.

2. A robotic arm to tighten nuts on some piece of machinery in an assembly line. It can be modelled as TSP problem by considering **nuts** as cities and arm movement as edge cost. The path of the arm is a **tour** of the graph with nuts as vertices and **minimal time needed by the robotic arm** to complete its task.

3. A Production environment in which several commodities are manufactured on the same set of machines

   It can be modelled as TSP problem by considering commodities as cities and change over cost between commodity i and commodity j as edge cost. The tour in this problem is the sequence of change of machines to produce commodities (**which sequence is minimum change over cost**).