Dynamic Programming

Joy Mukherjee

Matrix Chain Multiplication

Four Matrices

A is of order 5 X 4

B is of order 4 X 3

C is of order 3 X 5

D is of order 5 X 2

If you calculate ABCD, what is the minimum number of basic multiplication needed?

A(r1 X c1)

B(r2 X c2)

Condition for Multiplication: c1 = r2

No of basic multiplication = r1 * c1 * c2

Order of AB(r1 X c2)

Matrix multiplication is associative

Matrix Chain Multiplication

Four Matrices
A is of order 5 X 4
B is of order 4 X 3
C is of order 3 X 5
D is of order 5 X 2
If you calculate ABCD, what is the minimum number of basic multiplication needed?

Option	Result
A(B(CD))	3*5*2 + 4*3*2 + 5*4*2 = 30 + 24 + 40 = 94
A((BC)D)	4*3*5 + 4*5*2 + 5*4*2 = 60 + 40 + 40 = 140
(AB)(CD)	5*4*3 + 3*5*2 + 5*3*2 = 60 + 30 + 30 = 120
(A(BC))D	4*3*5 + 5*4*5 + 5*5*2 = 60 + 100 + 50 = 210
((AB)C)D	5*4*3 + 5*3*5 + 5*5*2 = 60 + 75 + 50 = 185

Important Observation

Observation: The last multiplication for a sequence of n matrices can happen in (n-1) ways.

Opt(M1M2M3M4M5) = Min(R1, R2, R3, R4) where

R1 = Opt(M1) + Opt(M2M3M4M5) + M1.r * M1.c * M5.c

R2 = Opt(M1M2) + Opt(M3M4M5) + M1.r * M2.c * M5.c

R3 = Opt(M1M2M3) + Opt(M4M5) + M1.r * M3.c * M5.c

R4 = Opt(M1M2M3M4) + Opt(M5) + M1.r * M4.c * M5.c

Matrix Chain Multiplication

Four Matrices
A is of order 5 X 4
B is of order 4 X 3
C is of order 3 X 5
D is of order 5 X 2

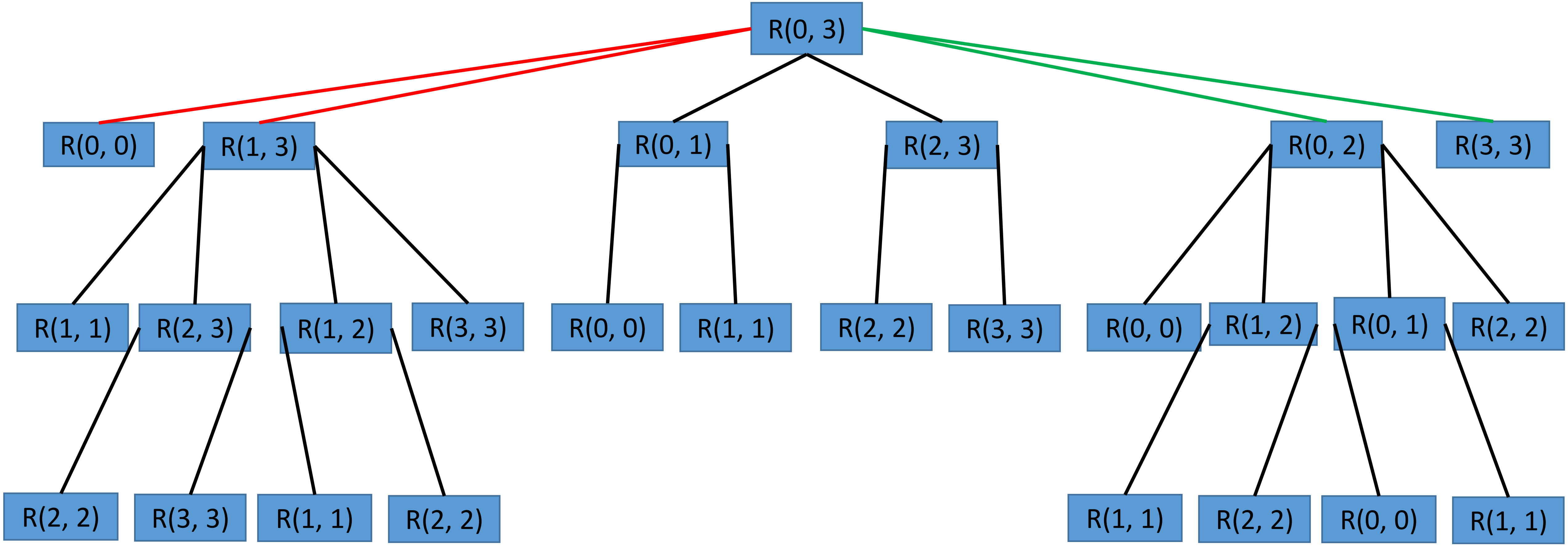
A((BC)D) = 4*3*5 (basic multiplication needed for X=BC, order of X = 4 X 5)

+ 4*5*2 (basic multiplication needed for Y=XD=(BC)D, order of Y = 4 X 2)

+ 5*4*2 (basic multiplication needed for Z=AY=A((BC)D), order of Z = 5 X 2)

= 140

Optimal Substructure



R(1, 3) = min(M1(M2M3), (M1M2)M3)

Optimal Substructure

Input: n matrices, each matrix M[i] has r rows and c columns

R(i, j) = 0 if i = j

R(i, j) = Minimum of {R(i, k) + R(k+1, j) + M[i].r * M[k].c * M[j].c } for all k = i to j-1

Minimum basic multiplications needed to multiply matrices from Mi to Mj

Order of matrix resulted from multiplying Mi to Mk   = M[i].r X M[k].c

Order of matrix resulted from multiplying Mk+1 to Mj = M[k+1].r X M[j].c

Execution

	M0	M1	M2	M3
r	5	4	3	5
c	4	3	5	2

R(i, j) = 0 if i = j

R(i, j) = Minimum of

{R(i, k) + R(k+1, j) + M[i].r * M[k].c * M[j].c } for all k = i to j-1

	0	1	2	3
0	0	M0M1	M02=Min(M0(M1M2), (M0M1)M2))	M03=Min(M0(M13), (M0M1)(M2M3), (M02)M3)
1		0	M1M2	M13=Min(M1(M2M3), (M1M2)M3))
2			0	M2M3
3				0

Execution

	M0	M1	M2	M3
r	5	4	3	5
c	4	3	5	2

R(i, j) = 0 if i = j

R(i, j) = Minimum of

{R(i, k) + R(k+1, j) + M[i].r * M[k].c * M[j].c } for all k = i to j-1

Min(M0(M1M2), (M0M1)M2))

	0	1	2	3
0	0	5*4*3 = 60	Min(60+5*4*5, 60+5*3*5)=135	Min(54+5*4*2, 60+54+5*3*2, 135+5*5*2) = 94
1		0	4*3*5 = 60	Min(30+4*3*2, 60+4*5*2)=54
2			0	3*5*2 = 30
3				0

Algorithm $O(n^3)$

```
int MCM(int n, Matrix m[])
{
        int i, k, l,  T[n][n];
        for(l = 0; l < n; l++) {
                for(i = 0; i < n-l; i++) {
                        j = i+l;
                        if(l == 0) T[i][j] = 0;
                        else {
                                T[i][j] = T[i][i] + T[i+1][j]+ m[i].r * m[i].c * m[j].c;
                                for(k = i+1; k < i+l; k++) // k denotes the intermediate break points
                                        if(T[i][j] > T[i][k] + T[k+1][j]+ m[i].r * m[k].c * m[j].c)
                                                T[i][j] = T[i][k] + T[k+1][j]+ m[i].r * m[k].c * m[j].c;
                        }
                }
        }
        return T[0][n-1];
}
```

Optimal Binary Search Tree

BST has three nodes 2, 5, 7 with frequency of search 30, 5, 50 respectively.

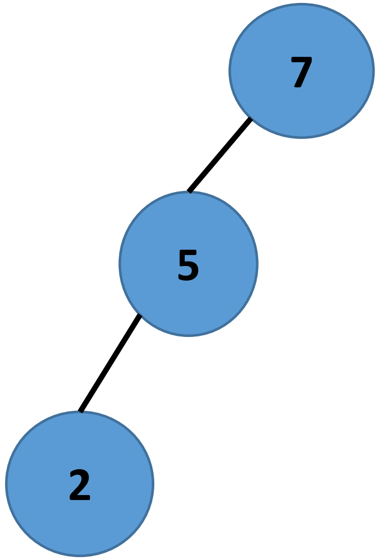
Level of root is 1.

Search cost of a BST node a = level(a) X frequency(a).

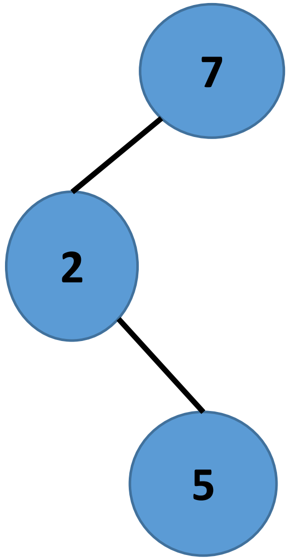
Objective: Construct a BST of all keys such that the total search cost is minimum.

Optimal Binary Search Tree

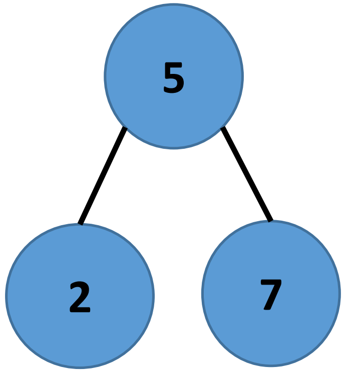
BST has three nodes 2, 5, 7 with frequency of search 30, 5, 50 respectively.
Level of root is 1.
Search cost of a BST node a = level(a) X frequency(a).
Objective: Construct a BST of all keys such that the total search cost is minimum.



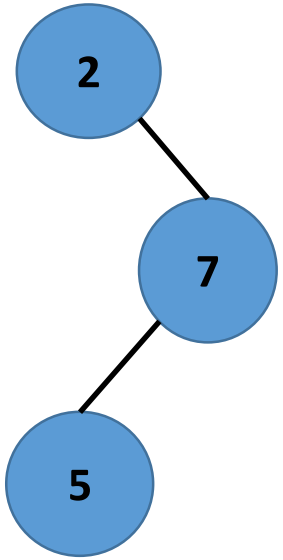
50*1 + 5*2 + 30*3
= 150



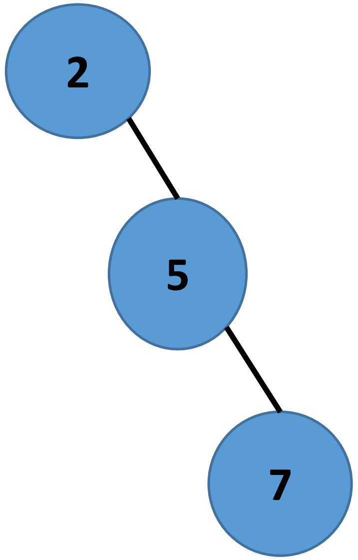
50*1 + 30*2 + 5*3
= 125 (Minimum)



5*1 + 30*2 + 50*2 =
165



30*1 + 50*2 + 5*3
= 145



30*1 + 5*2 + 50*3
= 190

Recursion

Any key can be a root (index of the root).

OPT(F[], start, end, level)

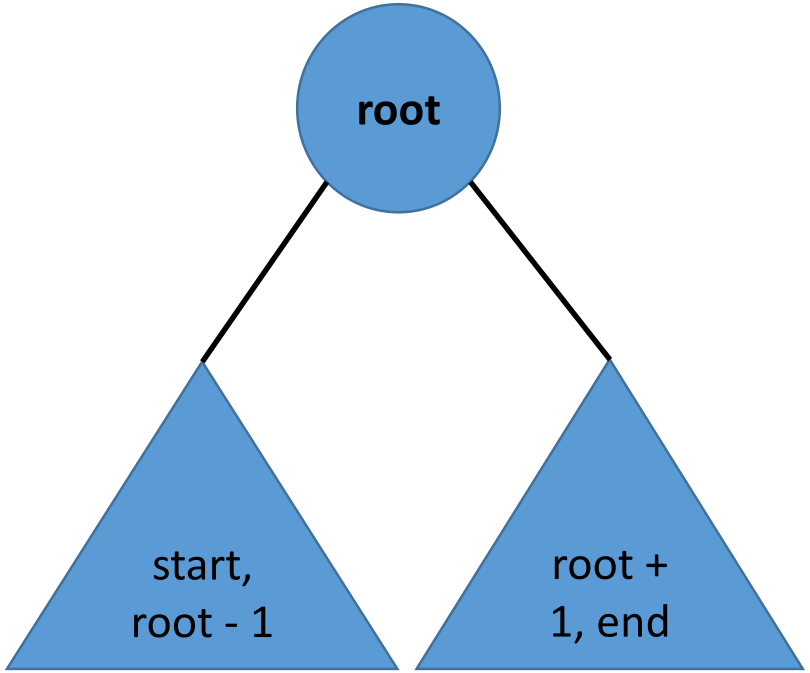
= 0 if start > end

= F[start] * level if start = end

= $\text{Minimum}_{\textbf{root = start to end}}$ { F[root] * level

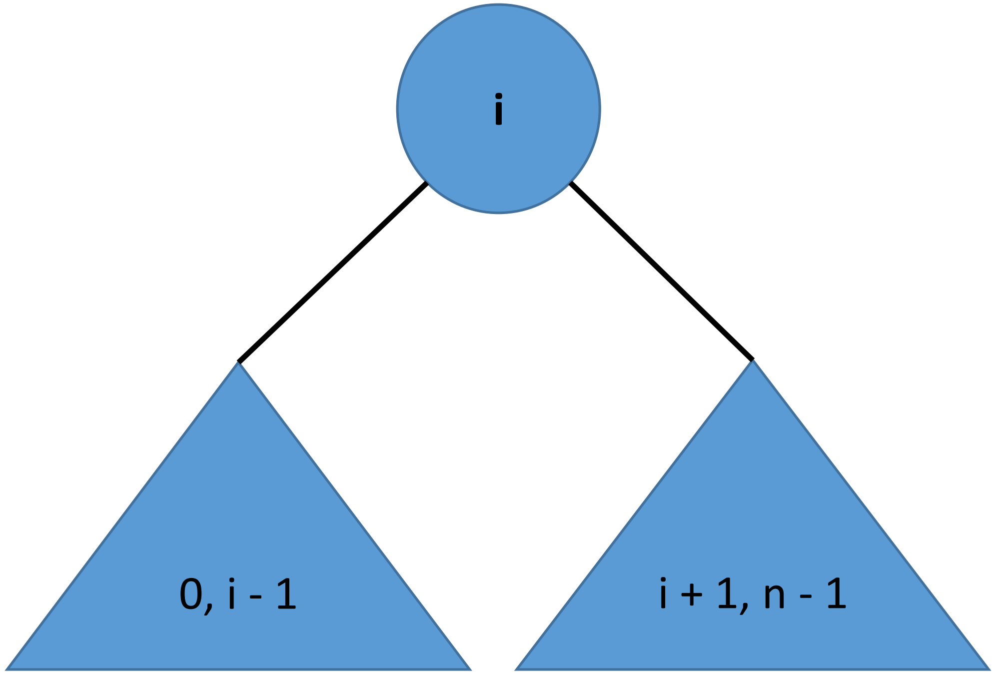
+ OPT(F, start, root-1, level+1)

+ OPT(F, root+1, end, level+1) }



Recursion

Any key can be a root (read as "index of the root").



OPT(F[], 0, n-1, 1)

$= \text{Minimum}_{i = 0 \text{ to } (n-1)} \{ F[i] * 1 + \text{OPT}(F, 0, i-1, 2) + \text{OPT}(F, i+1, n-1, 2) \}$

OPT(F[], 0, i-1, 2)

$= \text{Minimum}_{j = 0 \text{ to } (i-1)} \{ F[j] * 2 + \text{OPT}(F, 0, j-1, 3) + \text{OPT}(F, j+1, i-1, 3) \}$

OPT(F[], i+1, n-1, 2)

$= \text{Minimum}_{j = i+1 \text{ to } (n-1)} \{ F[j] * 2 + \text{OPT}(F, i+1, j-1, 3) + \text{OPT}(F, j+1, n-1, 3) \}$

Observation

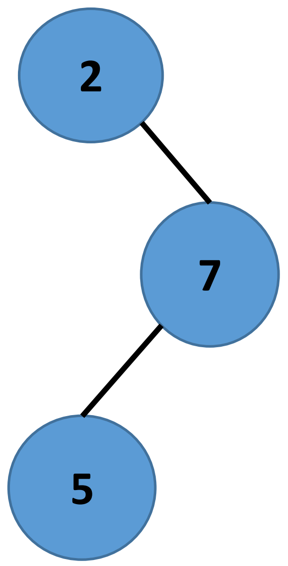
- Value of Key does not matter.
- The search cost of i-th level key

  = i X frequency

  = frequency added each time it

  passes a level

BST has three nodes 2, 5, 7 with frequency of search 30, 5, 50 respectively.



Total Search cost = 30*1 + 50*2 + 5*3 = 145
= (30 + 50 + 5) + (50 + 5) + 5
= Total Search cost through level 1+ Total Search cost through level 2+Total Search cost through level 3

Idea: To calculate search cost at a given level in O(1) time, we calculate the prefix sum for the F[]
Before the DP for OPT BST starts, we need to calculate prefix sum (levelSum) for the F[] in O(n) time.

Modified Recursion

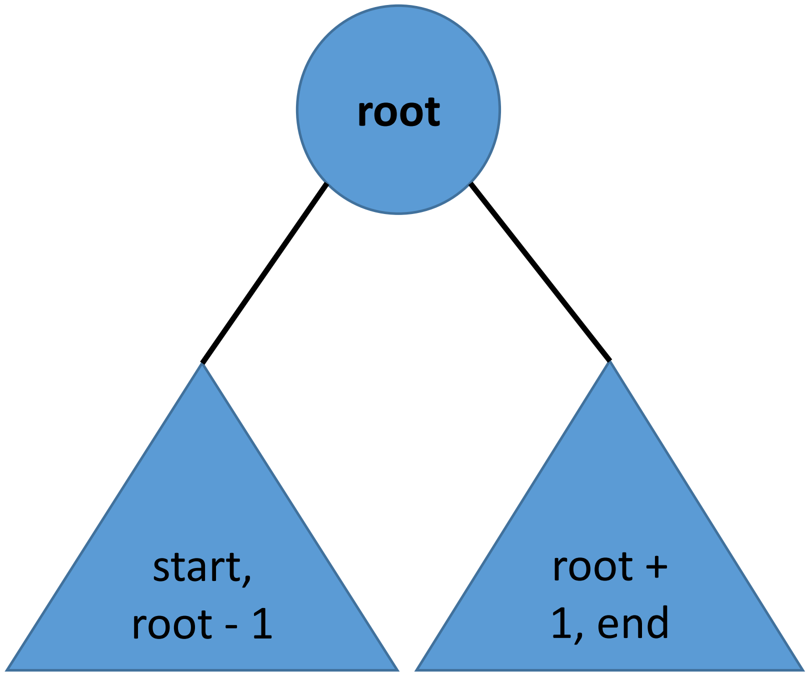
OPT(F[], start, end, levelSum)

= 0 if start > end

= F[start] if start = end

$$= \text{Minimum}_{\textbf{root = start to end}}\{\text{ levelSum[end]} - \text{levelSum[start-1]}$$
$$+ \text{ OPT(F, start, root-1, levelSum)}$$
$$+ \text{ OPT(F, root+1, end, levelSum) }\}$$

Note that levelSum[end] - levelSum[start-1] = F[start] + F[start-1]+…+F[end]



Key		2	8	9	20	99
Frequency		8	1	5	2	10
LevelSum (Prefix Sum)	0	8	9	14	16	26

i = 3

OPT(F[], 1, 5, levelSum[]) = $\text{Minimum}_{i = 1 \text{ to } 5}$\{ 26 + OPT(F, 1, i-1, levelSum) + OPT(F, i+1, 5, levelSum) \}

OPT(F, 1, i-1, levelSum) = $\text{Minimum}_{j = 1 \text{ to } i-1}$\{levelSum[i-1] – levelSum[0] + OPT(F, 1, j-1, levelSum) + OPT(F, j+1, i-1, levelSum) \}

OPT(F, i+1, 5, levelSum) = $\text{Minimum}_{j = i+1 \text{ to } 5}$\{levelSum[5] – levelSum[i] + OPT(F, i+1, j-1, levelSum) + OPT(F, j+1, 5, levelSum) \}

Execution

	0	1 (123)	2(128)	3(4095)
F	0	30	5	50
L	0	30	35	85

	1	2	3
1	30	Min{0+OPT(2,2), OPT(1,1)+0} + 35)	Min{0+OPT(2,3), OPT(1,1)+OPT(3,3), OPT(1,2)+0} + (L[3]-L[0])
2	0	5	Min{0+OPT(3,3), OPT(2,2)+0} + (L[3]-L[1])
3	0	0	50

OPT(F[], start, end, L[]) = 0 if start > end

= F[start] if start = end

= $\text{Minimum}_{\text{root = start to end}}$ { L[end] – L[start-1] = F[start] + F[start-1]+…+F[end]

+ OPT(F, start, root-1, L)

+ OPT(F, root+1, end, L) }

Execution

	0	1	2	3
F	0	30	5	50
L	0	30	35	85

	1	2	3
1	30	Min{5, 30} + 35 = 40	Min{60, 30+50, 40} + 85 = 125
2	0	5	Min{50, 5} + 55 = 60
3	0	0	50

OPT(F[], start, end, L[]) = 0 if start > end

= F[start] if start = end

= $\text{Minimum}_{\text{root = start to end}}$ { L[end] – L[start-1] = F[start] + F[start-1]+…+F[end]

+ OPT(F, start, root-1, L)

+ OPT(F, root+1, end, L) }

Algorithm

```
int OPT(int n, int F[])
{
	int i, j, l, k, cost[n][n];
	S[0] = F[0];
	for(i = 1; i < n; i++) S[i] = F[i] + S[i-1];
	for (l = 1; l < n; l++) {
		for (i = 0; i < n-l; i++) {
			j = i+l;
			int fsum = (i == 0) ? S[j] : S[j] - S[i-1];
			cost[i][j] = cost[i+1][j] + fsum;
			for (k=i+1; k<=j; k++) {
				int c = cost[i][k-1] + ((k < j)? cost[k+1][j]:0) + fsum;
				if (c < cost[i][j])
					cost[i][j] = c;
			}
		}
	}
	return cost[0][n-1];
}
```