Dynamic Programming

By
Prof. Shaik Naseera
Department of CSE
JNTUA College of Engg., Kalikiri

Objectives

- Knapsack Problem
- Reliability Design
- Matrix Chain Multiplication
- Optimal Binary Search Tree
- Previous Gate Questions

- Dynamic Programming is an algorithm design method that can be used when the solution to a problem can be viewed as the result of a sequence of decisions.

Principle of optimality

- An optimal sequence of decisions has the property that whatever the initial state and decisions are, the remaining decisions must constitute an optimal decision sequence with regard to the state resulting from the first decision.

0/1 knapsack problem

- Consider a knapsack instance n=3, (p1,p2,p3)=(1,2,5), (w1,w2,w3)=(2,3,4) and m=6.
- S^0={(0,0)} ; S_1^0 ={(1,2)}
- S^1={(0,0),(1,2)} S_1^1 ={(2,3),(3,5)}
- S^2={(0,0),(1,2),(2,3),(3,5)} S_1^2 ={(5,4),(6,6),(7,7),(8,9)}
- S^3={(0,0),(1,2),(2,3),(5,4),(6,6),(7,7),(8,9)}
- (3,5) is eliminated as a part of merge-purge rule

Merge-purge Rule

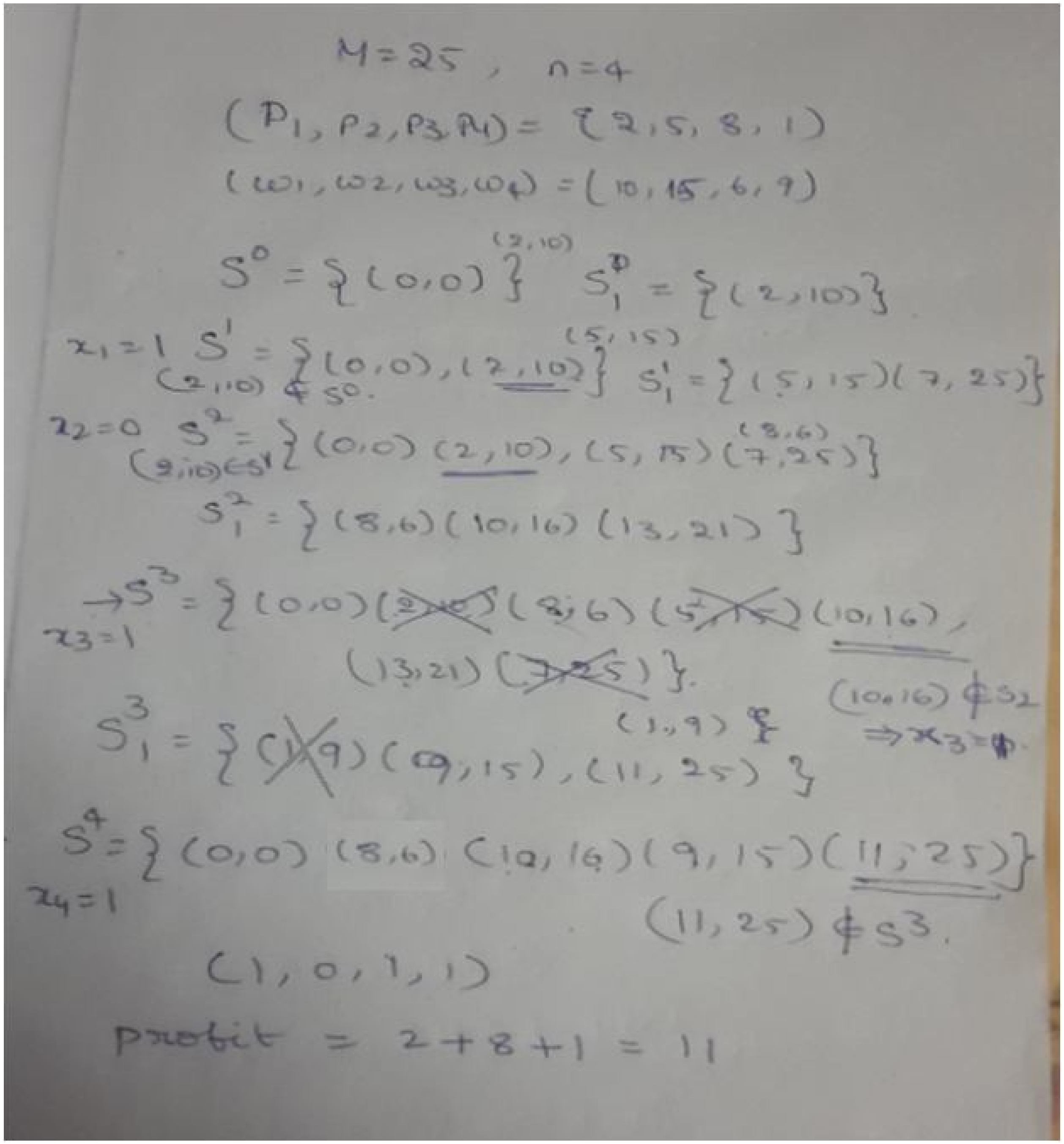
- Note that if S^{i+1} contains two pairs Let (P_i,W_i) and (P_j,W_j) such that

  if $(P_i \leq P_j)$ and $W_i \geq W_j$ then (P_i, W_i) can be discarded (purged).

Optimal solution

- M=6
- (6,6) is the tuple to meet the maximum capacity of the bag.
- (6,6)$\notin S^2$ therefore x3=1
- (6,6) is obtained from (1,2)
- (1,2)$\in S^1$ therefore x2=0
- (1,2) $\notin S^0$ therefore x1=1
- The optimal solution is (1, 0, 1) and the profit is 6

- N=4, m=25 (p1,p2,p3,p4)=(2,5,8,1) (w1,w2,w3,w4)=(10,15,6,9) obtain the optimal solution using dynamic programming



$M = 25$, $n = 4$

$(P_1, P_2, P_3, P_4) = \{2, 5, 8, 1)$

$(w_1, w_2, w_3, w_4) = (10, 15, 6, 9)$

$S^0 = \{(0,0)\}^{(2,10)}$ $S^0_1 = \{(2,10)\}$

$x_1 = 1$ $S^1 = \{(0,0), (\underline{2,10})\}^{(5,15)}$ $S^1_1 = \{(5,15)(7,25)\}$

$(2,10) \notin S^0.$

$x_2 = 0$ $S^2 = \{(0,0)\ (\underline{2,10}), (5,15)\overset{(8,6)}{(7,25)}\}$

$(2,10) \in S^1$

$S^2_1 = \{(8,6)(10,16)(13,21)\}$

$\rightarrow S^3 = \{(0,0)(\cancel{2,10})(8,6)(\cancel{5,15})(\underline{10,16}),$

$x_3 = 1$

$(13,21)(\cancel{7,25})\}.$

$(10,16) \notin S_2$

$\Rightarrow x_3 = 1.$

$(1,9) \notin$

$S^3_1 = \{(\cancel{1,9})(9,15), (11,25)\}$

$S^4 = \{(0,0)\ (8,6)\ (10,16)(9,15)(\underline{11,25})\}$

$x_4 = 1$

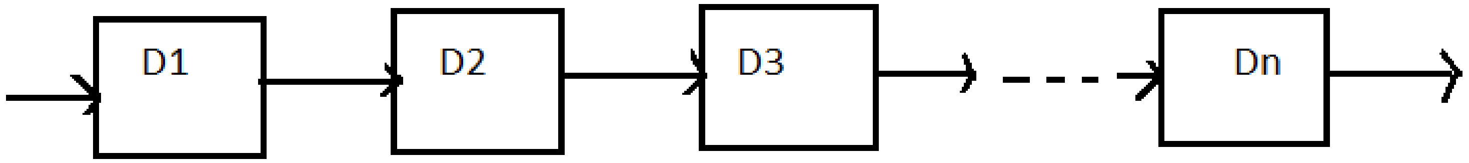
$(11,25) \notin S^3.$

$(1, 0, 1, 1)$

profit $= 2 + 8 + 1 = 11$

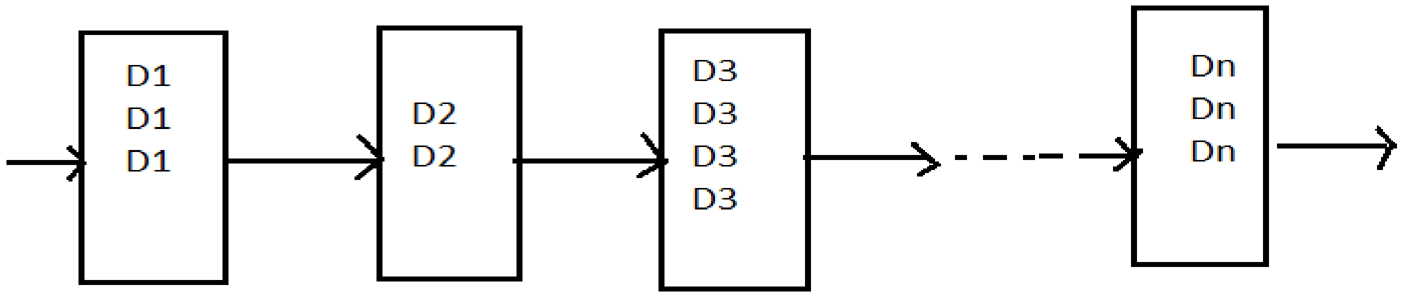
Reliability Design

- The problem is to design a system that is composed of several devices connected in series.



- Let r_i be the reliability of D_i . The reliability of the entire system is $\prod r_i$.
- Even if the individual devices are very reliable, the reliability of the entire system may not be very good.
- Ex:- n=10, r_i =0.99 then $\prod r_i$ =0.904.
- Hence it is desirable to duplicate the devices.

- Multiple copies of the same device type are connected in parallel through the use of switching circuits.



- The switching circuits determine which devices in any given group are functioning properly and make use of one such device at each stage.

- If stage i contains m_i copies of device D_i then the probability that all m_i have a malfunction is $(1-r_i)^{mi}$.
- The reliability of stage i becomes $(1-(1-r_i)^{mi})$.
- Let $\Phi_i(m_i)$ be the reliability of the stage i. Then $\Phi_i(m_i)=(1-(1-r_i)^{mi})$.
- $\prod\Phi_i(m_i)$ where $1 \le i \le n$ be the reliability of the entire system.
- The problem is to use device duplication to maximize the reliability.
- The maximization is to be carried out under cost constraint.

- Let c_i be the cost of device i.
- Let c be the cost of entire system (budget).
- The problem is to

$$\text{Maximize} \prod_{1 \le i \le n} \Phi_i (m_i)$$

Subject to $\Sigma\, c_i m_i \le c$, $1 \le i \le n$, $m_i \ge 1$.

- The number of devices that can be duplicated at each stage is

$$u_i = \left\lfloor (c + ci - \sum_{n}^{1} cj)/ci \right\rfloor$$

Q. Design a three stage system with device types D1, D2 and D3. The costs are \$30, \$15 and \$20 respectively. Reliability is 0.9, 0.8 and 0.5. The total cost of the system must not be more than c=\$105.

Ans.

$$u1= \left\lfloor (105+30-65)/30 \right\rfloor = \lfloor 2.33 \rfloor = 2$$

$$u2= \left\lfloor (105+15-65)/15 \right\rfloor = \lfloor 3.66 \rfloor =3$$

$$u3= \left\lfloor (105+20-65)/20 \right\rfloor = \lfloor 3.25 \rfloor =3$$

$$u_{i=} \left\lfloor (c+ci - \sum_{n}^{1} cj)/ci \right\rfloor$$

The Maximum number of units that each stage contain is u_i With respect to the allowed budget

System Design:

$$S^0 = \{(1,0)\}$$

$$S_1^1 = \{(0.9, 30)\} \quad S_2^1 = \{(0.99, 60)\}$$

$$S^1 = \{(0.9, 30), (0.99, 60)\}$$

$$S_1^2 = \{(0.72, 45), (0.792, 75)\}$$

$$S_2^2 = \{(0.864, 60), (0.9504, 90)\}$$

1-(1-0.8)^2=0.96*0.9=0.864
1-(1-0.8)^3=0.992*0.9=0.8928

$$S_3^2 = \{(0.8928, 75), (0.98208, 105)\}$$

$$S^2 = \{(0.72, 45), (0.864, 60), (0.8928, 75)\}$$

$$S_1^3 = \{(0.36, 65), (0.432, 80), (0.4464, 95)\}$$

$$S_2^3 = \{(0.54, 85), (0.648, 100)\}$$

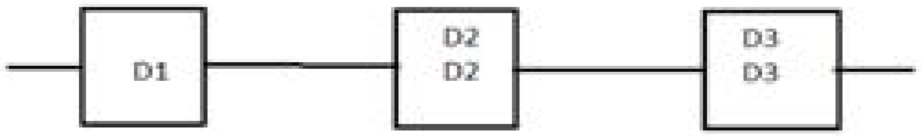
1-(1-0.5)^2=0.75*0.72=0.54
1-(1-0.5)^3=0.875*0.72=0.63

$$S_3^3 = \{(0.63, 105)\}$$

$$S^3 = \{(0.36, 65), (0.432, 80), (0.54, 85), (0.648, 100)\}$$

m1=1, m2=2 and m3=2

C=\$105, c1=\$30, c2=\$15, c3=\$20



m1=1, m2=2, m3=2 and reliability is 0.648 and cost is \$100.

Matrix Chain Multiplication

Problem Definition

- Input: a chain of matrices to be multiplied

- Output: a parenthesizing of the chain

- Objective: minimize number of steps

needed for the multiplication

Matrix Chain Multiplication

- Given : a chain of matrices $\{A_1, A_2, \ldots, A_n\}$.
- Once all pairs of matrices are *parenthesized*, they can be multiplied by using the standard algorithm as a sub-routine.
- A product of matrices is ***fully parenthesized*** if it is either a single matrix or the product of two fully parenthesized matrix products, surrounded by parentheses. [Note: since matrix multiplication is associative, all parenthesizations yield the same product.]

- Matrix multiplication is **associative** , e.g.,

$$A_1A_2A_3 = (A_1A_2)A_3 = A_1(A_2A_3),$$

so parenthenization does not change result.

- To compute the number of scalar multiplications necessary, we must know:
  - Algorithm to multiply two matrices
  - Matrix dimensions

Matrix Chain Multiplication cont.

- For example, if the chain of matrices is {A, B, C, D}, the product A, B, C, D can be fully parenthesized in 5 distinct ways:

  (*A* (*B* (*C D*))),
  (*A* ((*B C*) *D*)),
  ((*A B*) (*C D*)),
  ((*A* (*B C*)) *D*),
  (((*A B*) *C*) *D*).

$2nc_n/n+1$, substitute n=n-1
$\Rightarrow 2(n-1)c_{n-1}/n$
Let n=4, $6c_3/4 = (6*5*4/3*2*1)/4 = 5$ possible parenthesization

- The way the chain is parenthesized can have a dramatic impact on the cost of evaluating the product.

Matrix Chain Multiplication Optimal Parenthesization

- Example: A[30][35], B[35][15], C[15][5]
  minimum of A*B*C
  A*(B*C) = 30*35*5 + 35*15*5 = 7,585
  (A*B)*C = 30*35*15 + 30*15*5 = 18,000
- How to optimize:
  – Brute force – look at every possible way to parenthesize : $\Omega(4^n/n^{3/2})$
  – Dynamic programming – time complexity of $\Omega(n^3)$ and space complexity of $\Theta(n^2)$.

- The time to compute C is dominated by the number of scalar multiplications.
- To illustrate the different costs incurred by different paranthesization of a matrix product.

Example: Consider three matrices $\mathbf{A_{10\times100}}$, $\mathbf{B_{100\times5}}$, **and** $\mathbf{C_{5\times50}}$ There are 2 ways to parenthesize

((AB)C) = $D_{10\times5} \cdot C_{5\times50}$

$$\left.\begin{array}{l}\mathbf{AB \Rightarrow 10\cdot100\cdot5=5{,}000\ \text{scalar multiplications}} \\ \mathbf{DC \Rightarrow 10\cdot5\cdot50=2{,}500\ \text{scalar multiplications}}\end{array}\right\}$$

Total: 7,500

(A(BC)) = $A_{10\times100} \cdot E_{100\times50}$

$$\left.\begin{array}{l}\mathbf{BC \Rightarrow 100\cdot5\cdot50=25{,}000\ \text{scalar multiplications}} \\ \mathbf{AE \Rightarrow 10\cdot100\cdot50=50{,}000\ \text{scalar multiplications}}\end{array}\right\}$$

Total: 75,000

Matrix Chain Multiplication cont..

- Example: consider the chain A_1, A_2, A_3, A_4 of 4 matrices
  - Let us compute the product $A_1A_2A_3A_4$
- 5 different orderings = 5 different parenthesizations

1. $(A_1(A_2(A_3A_4)))$
2. $(A_1((A_2A_3)A_4))$
3. $((A_1A_2)(A_3A_4))$
4. $((A_1(A_2A_3))A_4)$
5. $(((A_1A_2)A_3)A_4)$

Algorithm..

Matrix-Multiply(A,B)

m*p , p*q matrices produce m*q matrix

```
1.If columns[A]!=rows[B]
2.  then error "incomplete dimensions"
3.  else for i ← 1 to rows[A]
4.        do for j ← 1 to columns[B]
5.            do C[i,j] ← 0
6.               for k ← 1 to columns[A]
7.                  Do C[i,j]=C[i,j]+A[i,k]*B[k,j]
8. Return C
```

- The time to compute C is dominated by the number of scalar multiplications.
- To illustrate the different costs incurred by different paranthesization of a matrix product.

Example: Consider three matrices $\mathbf{A}_{10\times100}$, $\mathbf{B}_{100\times5}$, **and** $\mathbf{C}_{5\times50}$ There are 2 ways to parenthesize

((AB)C) = $D_{10\times5} \cdot C_{5\times50}$

AB $\Rightarrow$ 10·100·5=5,000 scalar multiplications

DC $\Rightarrow$ 10·5·50 =2,500 scalar multiplications

Total: 7,500

(A(BC)) = $A_{10\times100} \cdot E_{100\times50}$

BC $\Rightarrow$ 100·5·50=25,000 scalar multiplications

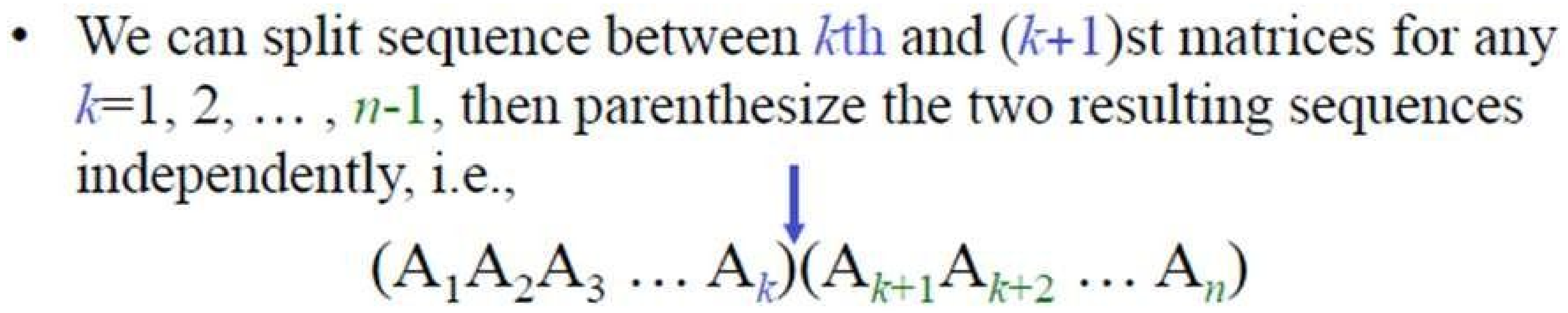
AE $\Rightarrow$ 10·100·50 =50,000 scalar multiplications

Total: 75,000

- Matrix-chain multiplication problem
  - ✓ Given a chain $A_1, A_2, \ldots, A_n$ of n matrices, where for $i=1, 2, \ldots, n$, matrix A_i has dimension $p_{i-1} \times p_i$
  - ✓ Parenthesize the product $A_1 A_2 \ldots A_n$ such that the total number of scalar multiplications is minimized

Counting the Number of Parenthesizations

- Before solving by Dynamic programming exhaustively check all paranthesizations.
- P(n) : paranthesization of a sequence of n matrices



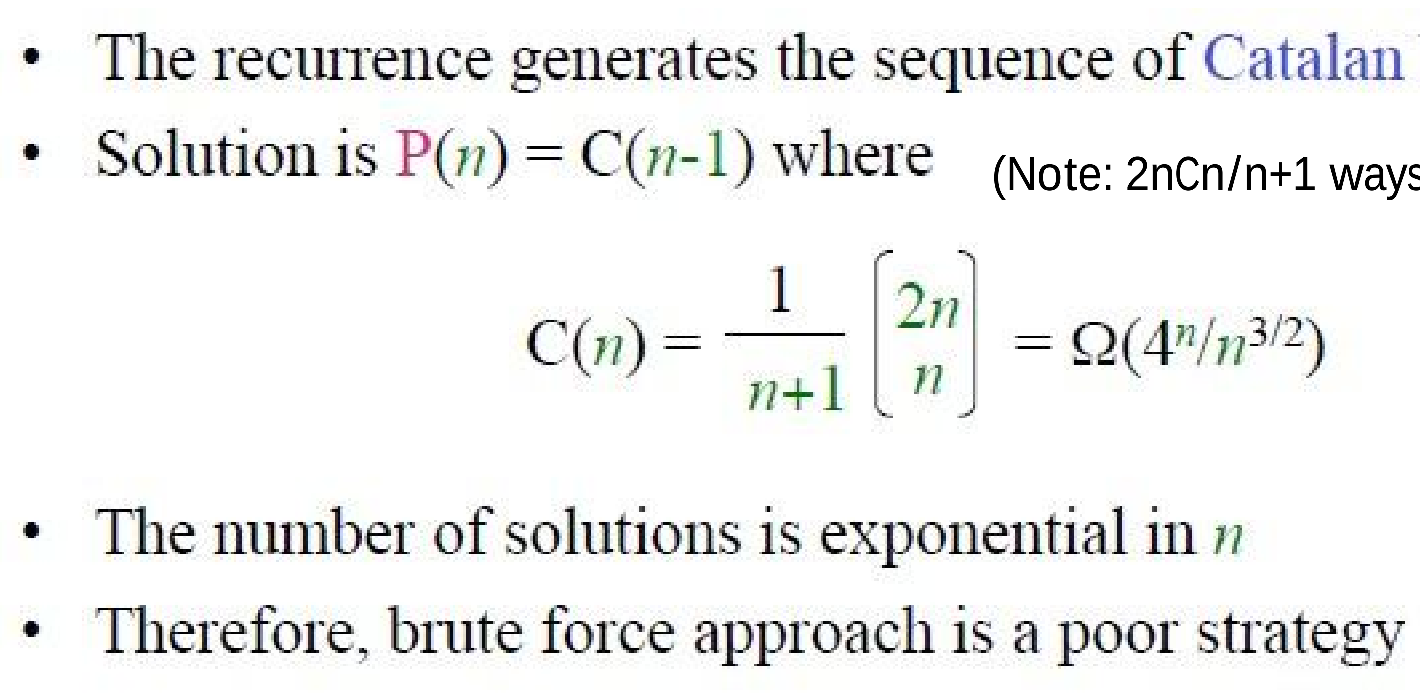
- We can split sequence between kth and $(k+1)$st matrices for any $k=1, 2, \ldots, n-1$, then parenthesize the two resulting sequences independently, i.e.,

$$(A_1A_2A_3 \ldots A_k)\overset{\downarrow}{}(A_{k+1}A_{k+2} \ldots A_n)$$

- We obtain the recurrence

$$P(n) = \begin{cases} 1 & \text{if } n = 1 \\ \sum_{k=1}^{n-1} P(k)p(n-k) & \text{if } n \geq 2 \end{cases}$$

Parenthesize P(k) and P(n-k) recursively



- The recurrence generates the sequence of Catalan Numbers
- Solution is $P(n) = C(n-1)$ where

$$C(n) = \frac{1}{n+1}\binom{2n}{n} = \Omega(4^n/n^{3/2})$$

- The number of solutions is exponential in n
- Therefore, brute force approach is a poor strategy

(Note: 2nCn/n+1 ways of parenthesizing.)

1.The Structure of an Optimal Parenthesization

Step 1: Characterize the structure of an optimal solution

- $A_{i..j}$: matrix that results from evaluating the product
  $A_i\, A_{i+1}\, A_{i+2} \ldots A_j$

- An optimal parenthesization of the product $A_1A_2 \ldots A_n$
  - Splits the product between A_k and A_{k+1}, for some $1 \leq k < n$

    $$(A_1A_2A_3 \ldots A_k) \cdot (A_{k+1}A_{k+2} \ldots A_n)$$

  - i.e., first compute $A_{1..k}$ and $A_{k+1..n}$ and then multiply these two

- The cost of this optimal parenthesization :
  (Cost of computing $A_{1..k}$ + Cost of computing $A_{k+1..n}$ + Cost of multiplying $A_{1..k} \cdot A_{k+1..n}$)

Optimal (sub)structure:

- Suppose that optimal parenthesization of $A_{i,j}$ splits between A_k and A_{k+1}.
- Then, parenthesizations of $A_{i,k}$ and $A_{k+1,j}$ must be optimal, too

(otherwise, enhance overall solution — subproblems are independent!).

➢ Construct optimal solution:

1. split into subproblems (using optimal split!),
2. parenthesize them optimally,
3. combine optimal subproblem solutions.

Example

Let us consider three matrices namely A1, A2 and A3 each of dimensions (2 * 3) , (3 * 4) and (4 * 2)
For example the matrices be parenthasized in the following manner.
((A1*A2)*A3)

```
((A1   *   A2)  *  A3 )
  2 * 3     3 * 4       4 * 2  = 0 (since A3 is single matrix, the value is 0)

      2 * 3* 4 = 24

             2  * 4 * 2 = 16        => p0, p2,p3
```

Hence the common formula can be derived from the above example as follows.
Let us consider the cost of the matrix as C[i,j] and the dimensions as 'P'. Now for the above example the dimensions are termed as

```
2 * 3       3  *  4         4  *  2
p0  p1      p1   p2         p2    p3
```

The cost matrix is taken as C[1,2] + C[3,3] + (p0 * p2 * p3) , where we consider

$$\underbrace{C[\overset{}{1}_{i},\,2_{k}]}\ +\ C[\,3_{k+1}\ ,\ 3_{j}\,] + (p0_{p_{i-1}} * p2_{p_k} * p3_{p_j})$$

$$C[i,j] = \min_{i \le k \le j} \{\, c[i,k] + c[k+1,j] + p_{i-1} * p_k * p_j \,\}$$

2. Recursively def. value of opt. solution

•Let m[i, j] denote **minimum number of scalar multiplications** needed to compute Ai;j = Ai*Ai+1….Aj (full problem: m[1, n]).

Recursive definition of m[i, j]:

• if i = j, then

m[i, j] = m[i, i] = 0 (Ai,i = Ai, no mult. needed).

- if $i < j$, assume optimal split at k, $i \leq k < j$. $A_{i,k}$ is $p_{i-1} \times p_k$ and $A_{k+1,j}$ is $p_k \times p_j$, hence

$$m[i,j] = m[i,k] + m[k+1,j] + p_{i-1} \cdot p_k \cdot p_j.$$

P is the dimension of the matrix

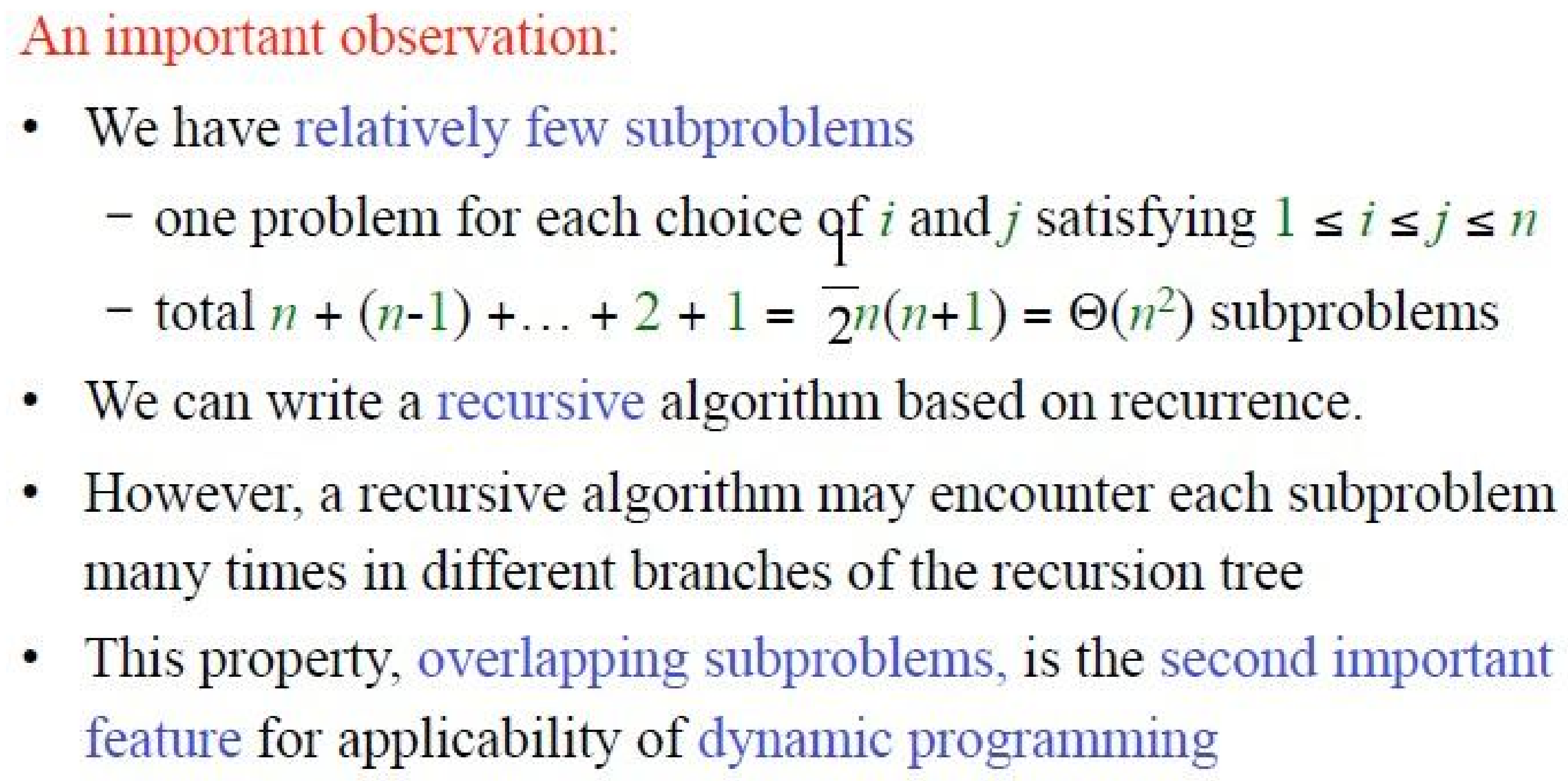
- $m_{ij} = m_{ik} + m_{k+1,j} + p_{i\text{-}1} \times p_k \times p_j$
  - We do not know k, but there are j-i possible values for k; $k = i, i+1, i+2, \ldots, j-1$

$$m[i,j] = \begin{cases} 0 & \text{if } i = j \\ \min_{i \le k < j}\{m[i,k] + m[k+1,j] & \text{if } i < j \\ +p_{i-1} \cdot p_k \cdot p_j\} & \end{cases}$$

- We also keep track of optimal splits:

$$s[i,j] = k \;\Leftrightarrow m[i,j] = m[i,k] + m[k+1,j] + p_{i-1} \cdot p_k \cdot p_j$$

4.Computing the Optimal Cost



An important observation:

- We have relatively few subproblems
  - one problem for each choice of i and j satisfying $1 \leq i \leq j \leq n$
  - total $n + (n-1) + \ldots + 2 + 1 = \frac{1}{2}n(n+1) = \Theta(n^2)$ subproblems
- We can write a recursive algorithm based on recurrence.
- However, a recursive algorithm may encounter each subproblem many times in different branches of the recursion tree
- This property, overlapping subproblems, is the second important feature for applicability of dynamic programming

Computing the Optimal Cost(cont..)

Compute the value of an optimal solution in a bottom-up fashion

- matrix A_i has dimensions $p_{i-1} \times p_i$ for $i = 1, 2, \ldots, n$
- the input is a sequence $\langle p_0, p_1, \ldots, p_n \rangle$ where length[p] = $n + 1$

Procedure uses the following auxiliary tables:

- $m[1\ldots n, 1\ldots n]$: for storing the $m[i, j]$ costs
- $s[1\ldots n, 1\ldots n]$: records which index of k achieved the optimal cost in computing $m[i, j]$

Algorithm for Computing the Optimal Costs

```
MATRIX-CHAIN-ORDER($p$)
    $n \leftarrow$ length[$p$] -1
    for $i \leftarrow 1$ to $n$ do
        $m[i, i] \leftarrow 0$
    for $\ell \leftarrow 2$ to $n$ do
        for $i \leftarrow 1$ to $n - \ell + 1$ do
            $j \leftarrow i + \ell - 1$
            $m[i, j] \leftarrow \infty$
            for $k \leftarrow i$ to $j$-1 do
                $q \leftarrow m[i, k] + m[k+1, j] + p_{i-1} p_k p_j$
                if $q < m[i, j]$ then
                    $m[i, j] \leftarrow q$
                    $s[i, j] \leftarrow k$
    return $m$ and $s$
```

Example:

$$A_1 \quad 30\times 35 \quad = p_0 \times p_1$$
$$A_2 \quad 35\times 15 \quad = p_1 \times p_2$$
$$A_3 \quad 15\times 5 \quad = p_2 \times p_3$$
$$A_4 \quad 5\times 10 \quad = p_3 \times p_4$$
$$A_5 \quad 10\times 20 \quad = p_4 \times p_5$$
$$A_6 \quad 20\times 25 \quad = p_5 \times p_6$$

m	1	2	3	4	5	6
1	0	15750	7875	9375	11875	15125
2		0	2625	4375	7125	10500
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0

s	1	2	3	4	5	6
1		1	1	3	3	3
2			2	3	3	3
3				3	3	3
4					4	5
5						5
6						

Sample Calculation

$m[2,5]=$

min{

$m[2,2]+m[3,5]+p_1p_2p_5=0+2500+35\times15\times20$
$=13000,$

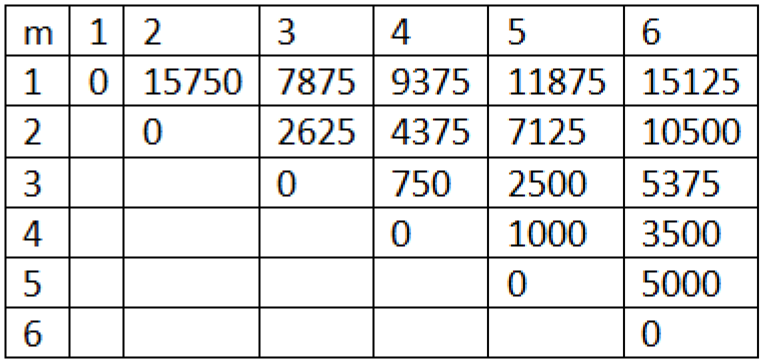
$m[2,3]+m[4,5]+p_1p_3p_5=2625+1000+35\times5\times$
$20=7125,$

$m[2,4]+m[5,5]+p_1p_4p_5=4375+0+35\times10\times20$
$=11374$

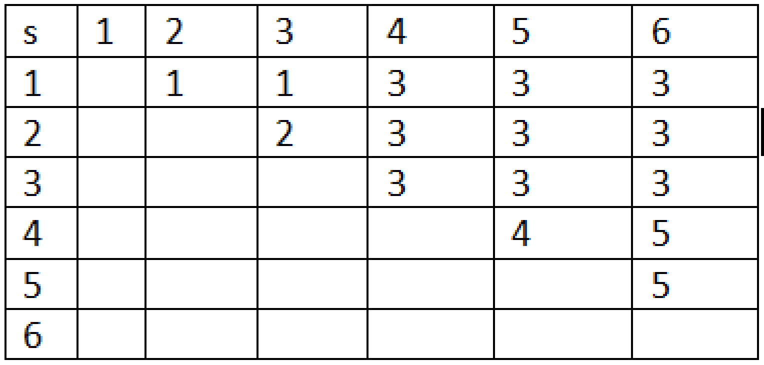
}

$=7125$

$$C[i,j] = \min_{i \le k \le j} \{ c[i,k] + c[k+1,j] + p_{i-1} * p_k * p_j \}$$



m	1	2	3	4	5	6
1	0	15750	7875	9375	11875	15125
2		0	2625	4375	7125	10500
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0



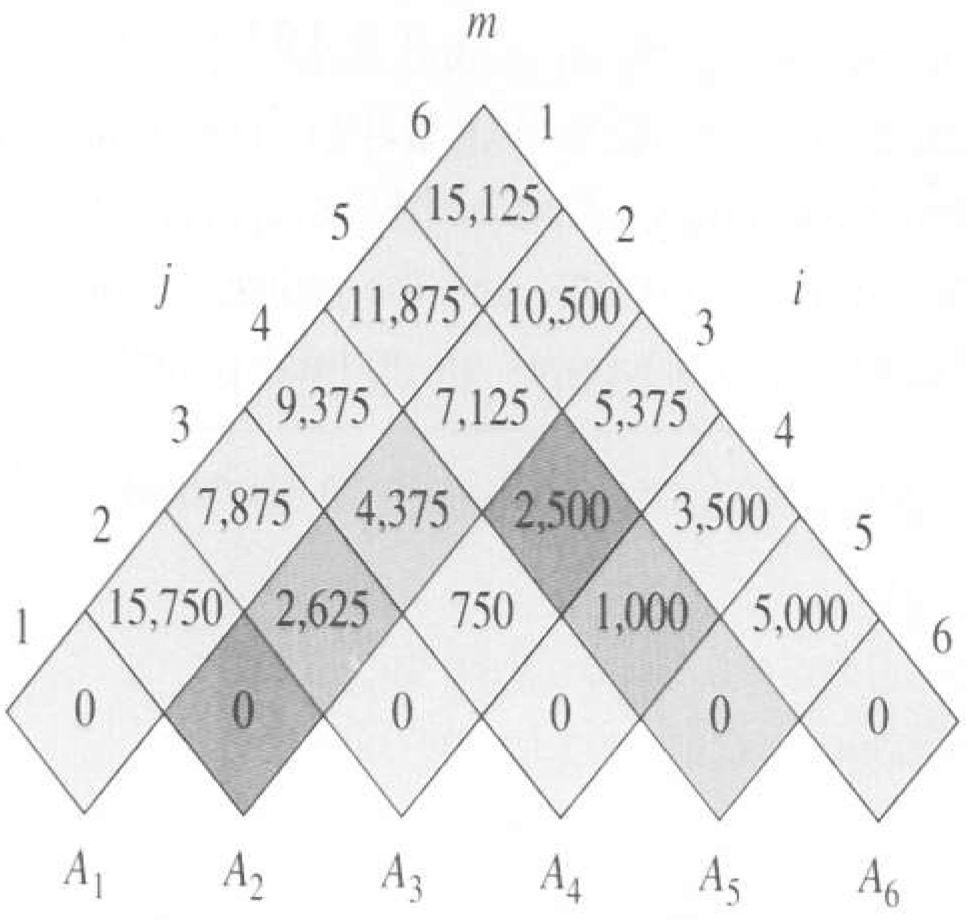
s	1	2	3	4	5	6
1		1	1	3	3	3
2			2	3	3	3
3				3	3	3
4					4	5
5						5
6						

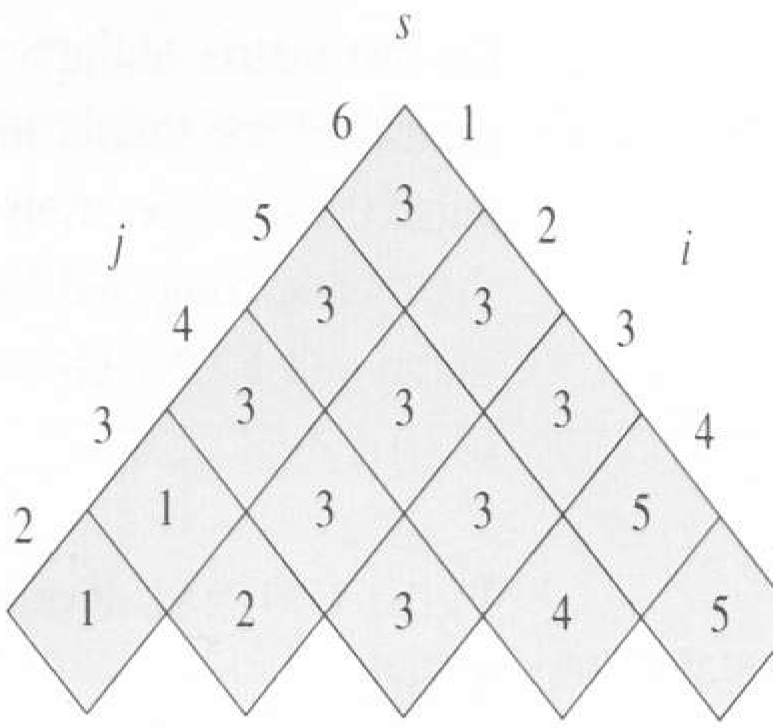
A1 A2 A3 A4 A5 A6

(A1 A2 A3) A4 A5 A6

(A1 (A2 A3)) ((A4 A5) A6)

The m and s table computed by MATRIX-CHAIN-ORDER for n=6





A1 A2 A3 A4 A5 A6

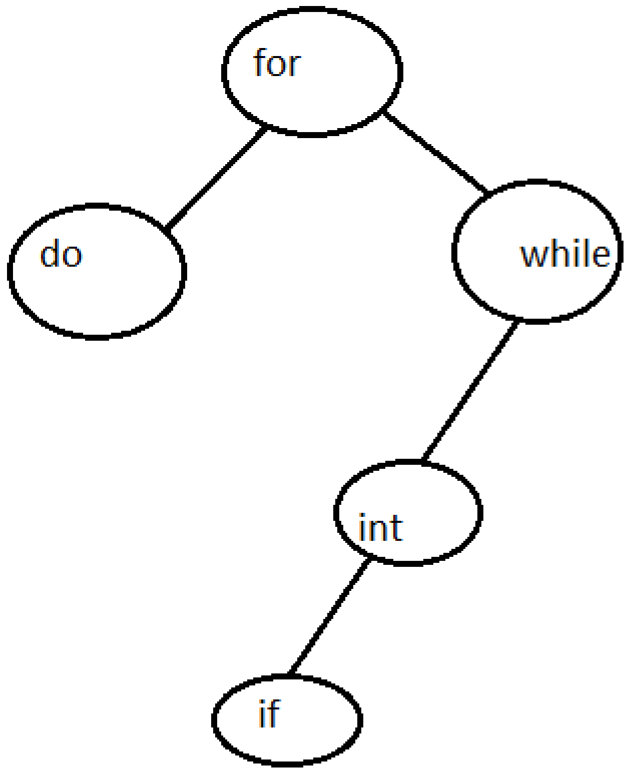
(A1 A2 A3) A4 A5 A6

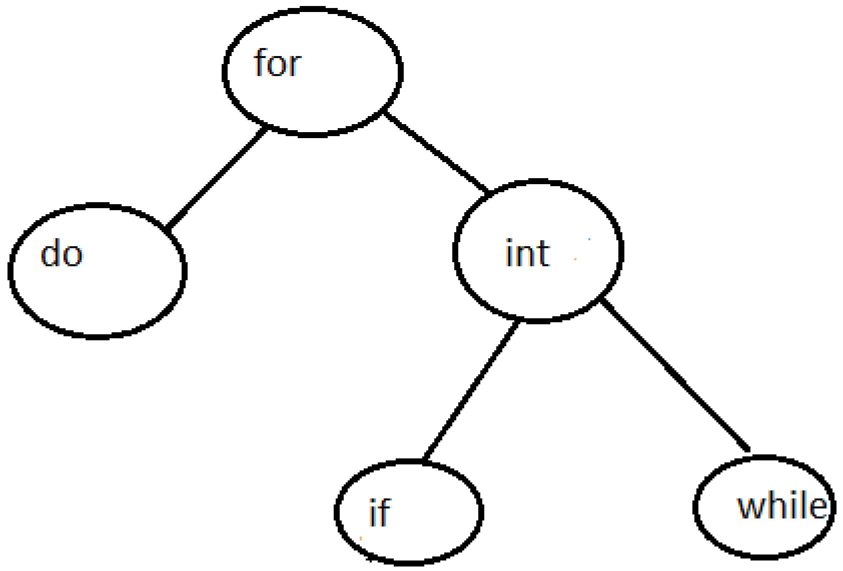
(A1 (A2 A3)) ((A4 A5) A6)

Optimal Binary Search Trees (OBST)

- Binary Search Tree : The values present in the left subtree are less than root value and the values present in the right subtree are greater than the root value.
- For a given set of identifies, we can construct more than one binary search tree.
- The binary search trees exhibit different performance characteristics.

For successful search cases



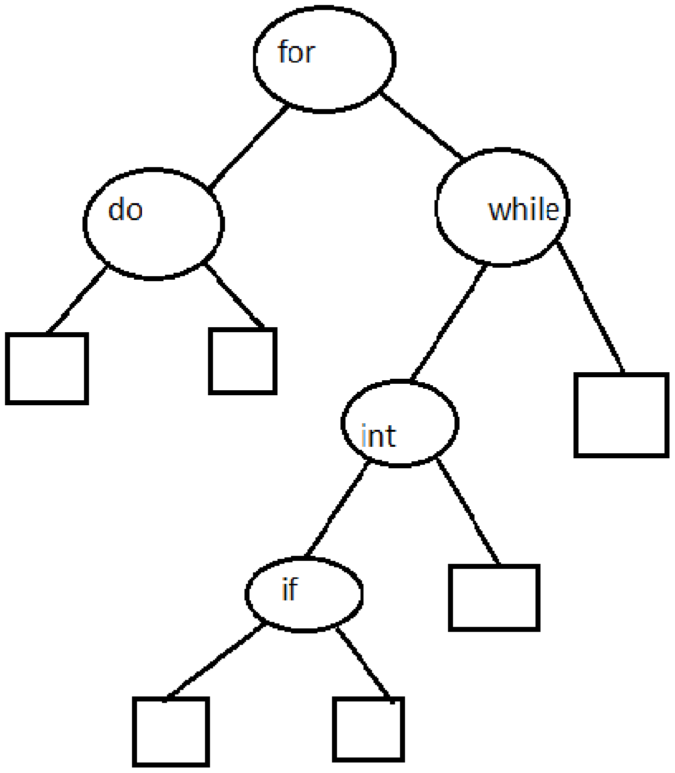


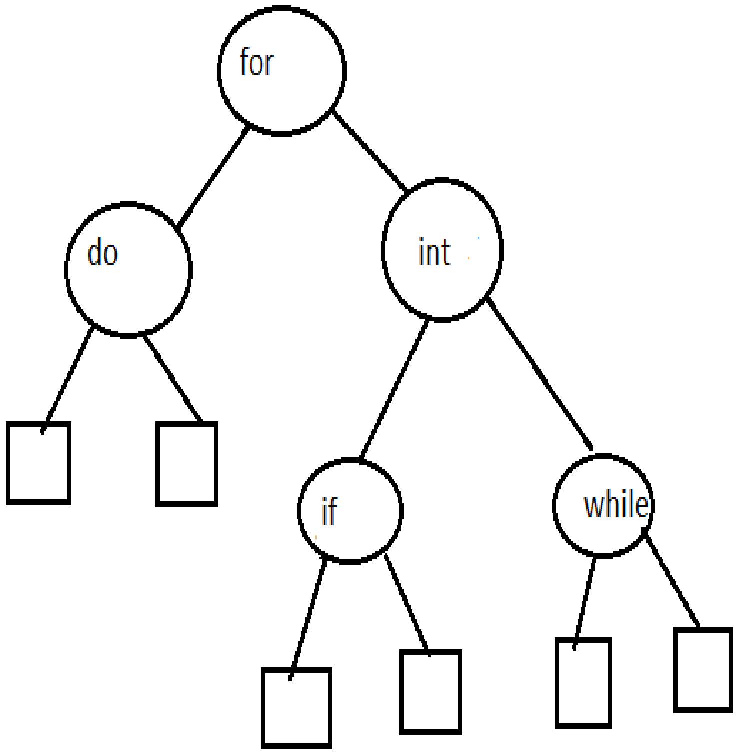
(1+2+2+3+4)/5=12/5

(1+2+2+3+3)/5=11/5

- In general situation, we expect different identifiers to be searched for with different frequencies (probabilities).
- There may be unsuccessful search cases also.
- Let {a1,a2 …,an} with a1<a2<…<an.
- Let p(i) be the probability of with which each a_i is searched.
- Then, Σp(i) where $1 \le i \le n$ is the probability of successful search.
- Let q(i) be the probability that the identifier x is searched such that $a_i < x < a_{i+1}$.
- Then, Σq(i) where $0 \le i \le n$ is the probability of unsuccessful search.
- $\sum_{1 \le i \le n} p(i) + \sum_{0 \le i \le n} q(i) = 1$

- To obtain a cost function for binary search tree, it is useful to add a fictitious node in place of empty sub tree in the search tree.





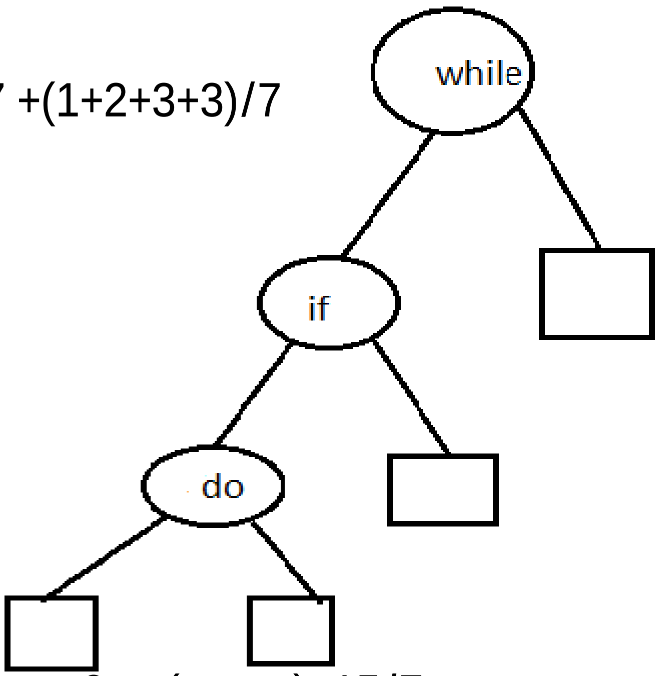
- Successful search terminate at level *l*, unsuccessful search terminates at node level-1.
- The formula for the expected cost of the binary search tree is

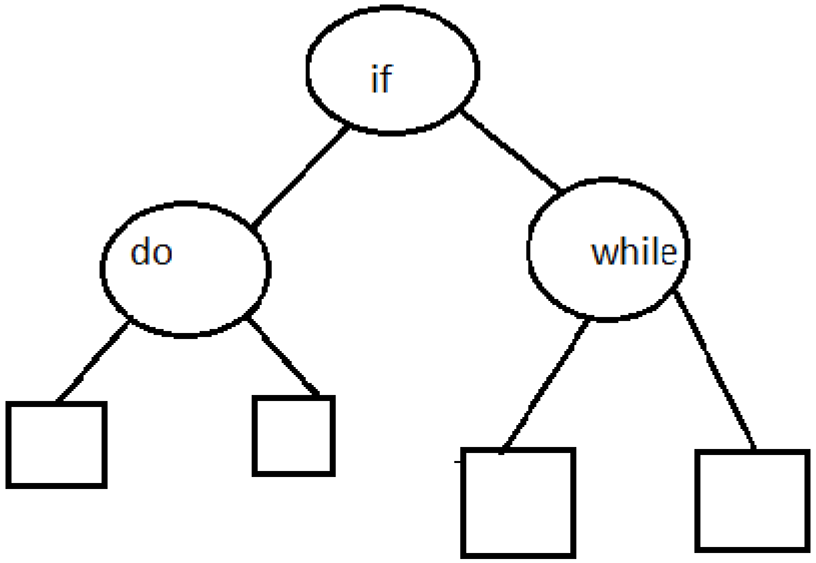
$$\sum_{1 \le i \le n} p(i) * level(a_i) + \sum_{0 \le i \le n} q(i) * (level(E_i)-1)$$

Possible binary search trees

- equal probability
- (a1,a2,a3)=(do, if, while)   P(i)=q(i)=1/7

=(1+2+3)/7 +(1+2+3+3)/7
=15/7

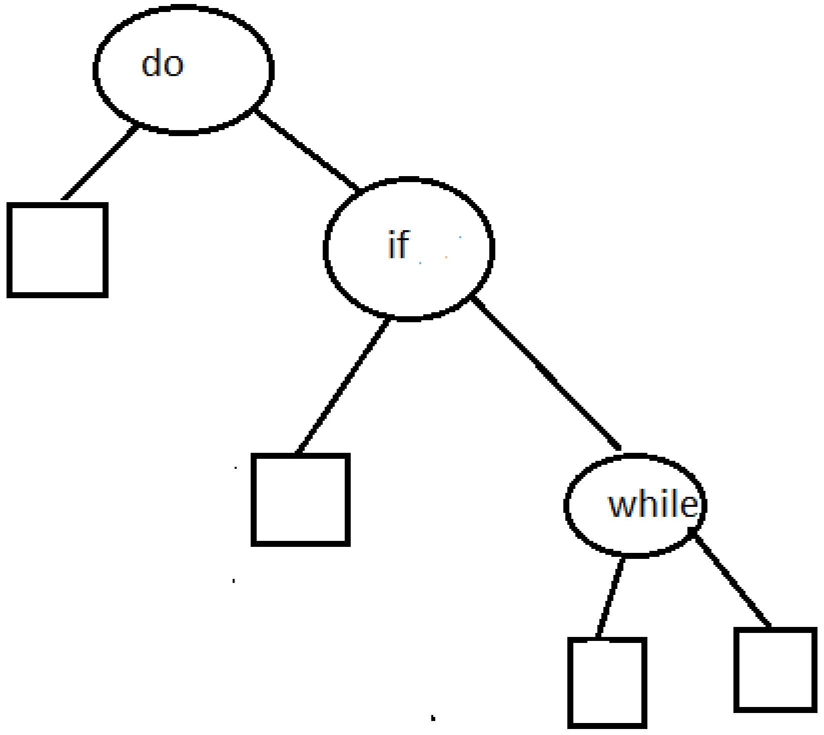


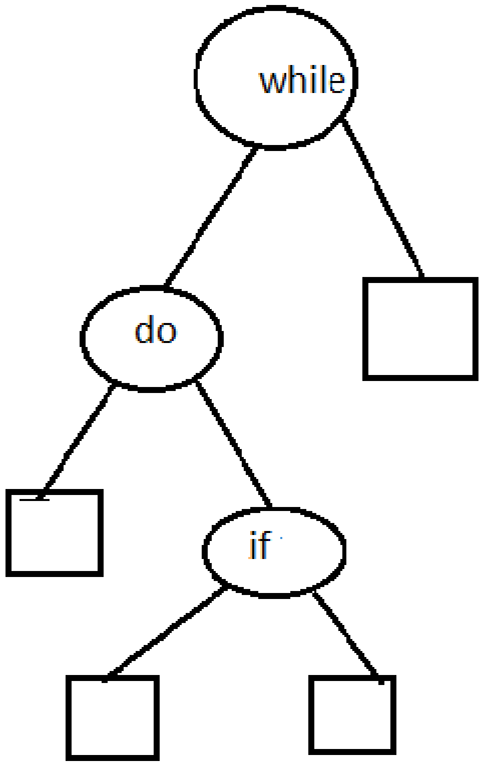


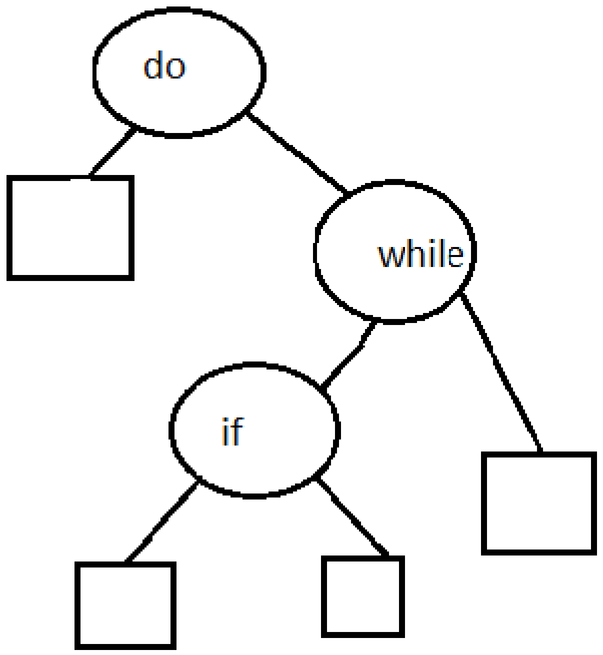
Cost(tree a)=15/7

cost(tree b)=13/7
=((1+2+2)+(2+2+2+2))/7

- Cost(tree c)=15/7 cost(tree d)=15/7



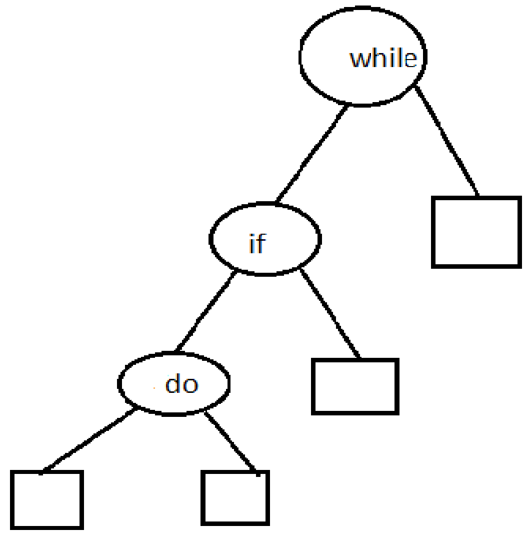




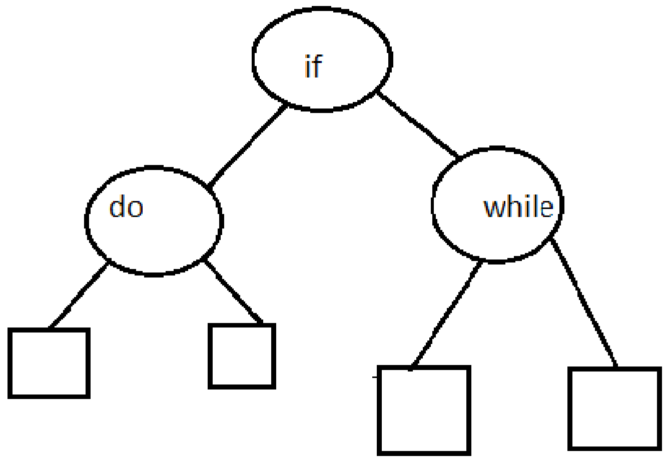
Cost(tree e)=15/7

Possible binary search trees

- Un-equal probability (a1,a2,a3)=(do, if, while)
- P(1)=0.5, P(2)=0.1, P(3)=0.05
- q(0)=0.15, q(1)=0.1, q(2)=0.05, q(3)=0.05

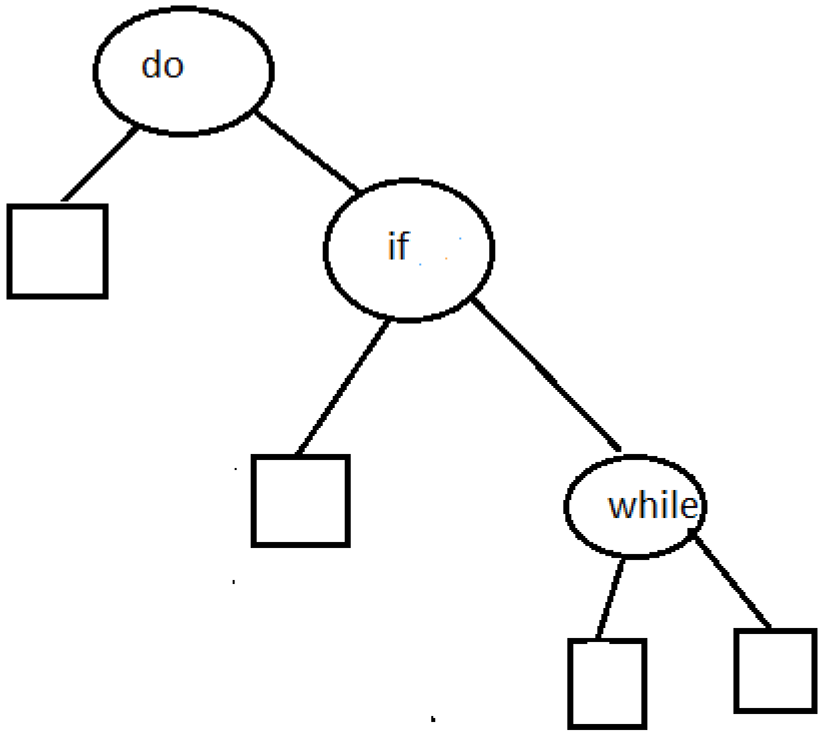


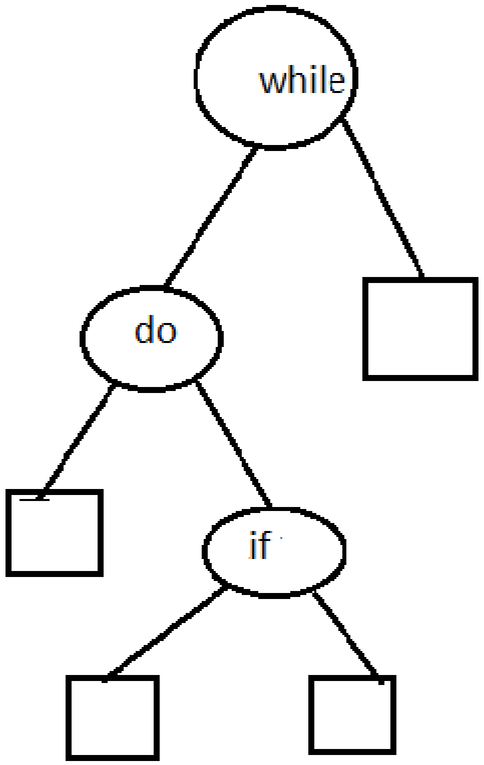
Cost(tree a)=2.65

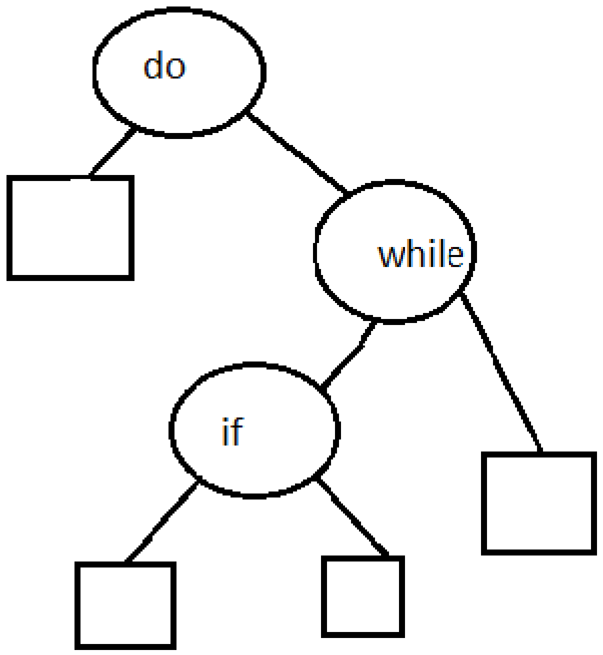


cost(tree b)=1.9

- Cost(tree c)=1.5 cost(tree d)=2.05



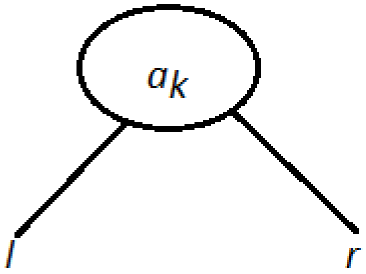




Cost(tree e)=1.6

OBST using Dynamic Programming

- An optimal binary search tree with root $a_{k.}$
- *Let* $w(i, j) = q(i)+\Sigma(q(l)+p(l)),\ i+1 \leq l \leq j.$
- *Expected cost* $= p(k)+cost(l)+cost(r)+w(0,k-1)+w(k,n)$



- $C(i, j)=\min\{cost(i, k-1)+cost(k, j)+p(k)+w(i,k-1)+w(k,n)\}$ is minimum

w(i, j) is for external nodes

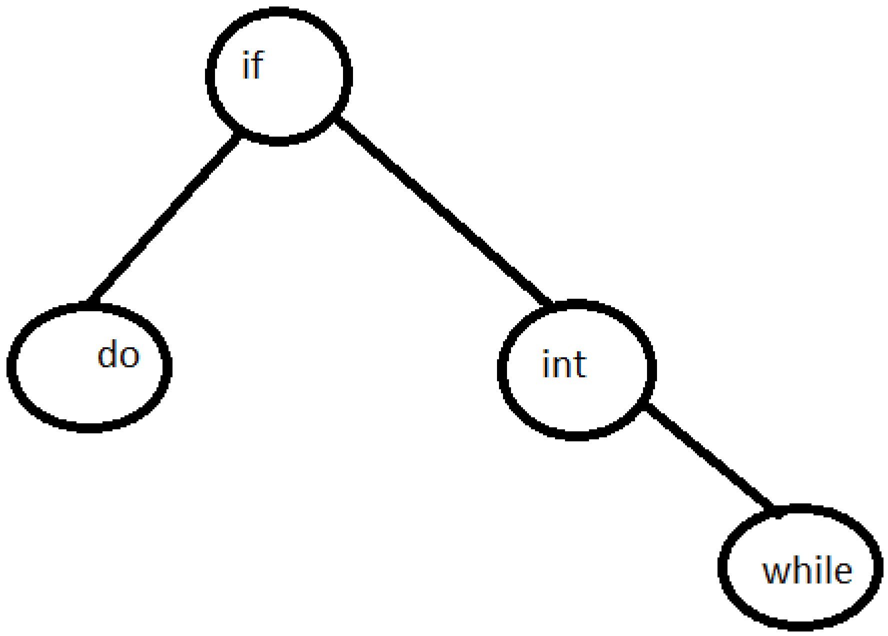
- $c(i, j) = \min_{i < k \leq j}\{c(i, k-1) + c(k, j)\} + w(i, j)\}$
- $w(i, j) = p(j) + q(j) + w(i, j-1)$
- $r(i, j)$ = k that minimizes $c(i, j)$

- $w(i, i) = q(i)$
- $c=(i, i)=0$
- $r(i, i) = 0$

Construct OBST for n=4, (a1,a2,a3,a4)=(do,if,int,while) p(1:4)=(3,3,1,1) and q(0:4)=(2,3,1,1,1).

$w_{00}=2$ $c_{00}=0$ $r_{00}=0$	$w_{11}=3$ $c_{11}=0$ $r_{11}=0$	$w_{22}=1$ $c_{22}=0$ $r_{22}=0$	$w_{33}=1$ $c_{33}=0$ $r_{33}=0$	$w_{44}=1$ $c_{44}=0$ $r_{44}=0$
$w_{01}=8$ $c_{01}=8$ $r_{01}=1$	$w_{12}=7$ $c_{12}=7$ $r_{12}=2$	$w_{23}=3$ $c_{23}=3$ $r_{23}=3$	$W_{34}=3$ $C_{34}=3$ $r_{34}=4$	
$w_{02}=12$ $c_{02}=19$ $r_{02}=1$	$w_{13}=9$ $c_{13}=12$ $r_{13}=2$	$w_{24}=5$ $c_{24}=8$ $r_{24}=3$		
$w_{03}=14$ $c_{03}=25$ $r_{03}=2$	$w_{14}=11$ $c_{14}=19$ $r_{14}=2$			
$w_{04}=16$ $c_{04}=32$ $r_{04}=2$				

OBST for given example



OBST Algorithm

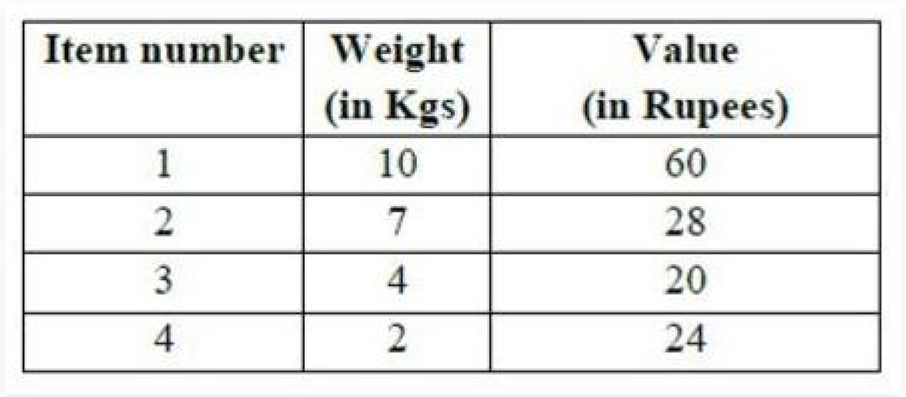
- obst.docx

Previous Gate Questions

1.

GATE | GATE CS 2018 | Question 60

Consider the weights and values of items listed below. Note that there is only one unit of each item.



Item number	Weight (in Kgs)	Value (in Rupees)
1	10	60
2	7	28
3	4	20
4	2	24

The task is to pick a subset of these items such that their total weight is no more than 11 Kgs and their total value is maximized. Moreover, no item may be split. The total value of items picked by an optimal algorithm is denoted by V_{opt}. A greedy algorithm sorts the items by their value-to-weight ratios in descending order and packs them greedily, starting from the first item in the ordered list. The total value of items picked by the greedy algorithm is denoted by V_{greedy}.

The value of $V_{opt} - V_{greedy}$ is ______ .

Note –This was Numerical Type question.

(A) 16

(B) 8

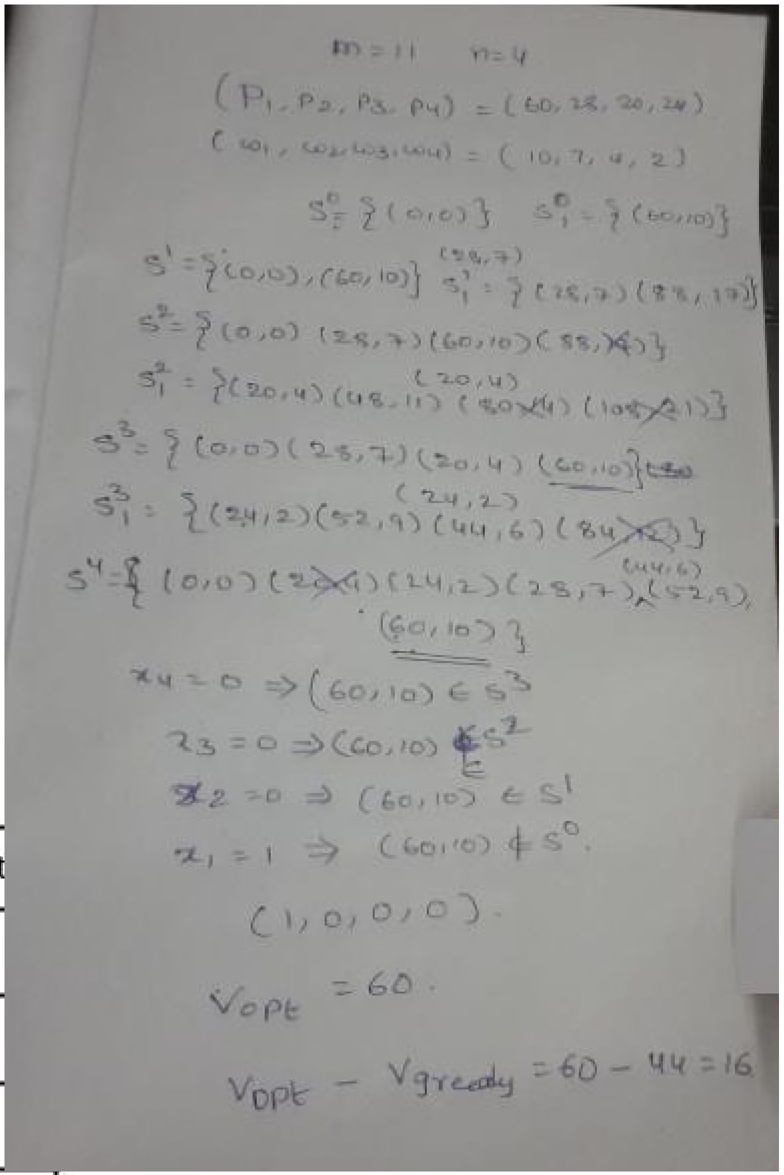
(C) 44

(D) 60

Answer: (A)

Explanation:

Item No	Weight	Value	Value/Weight
1	10	60	6
2	7	28	4
3	4	20	5
4	2	24	12



$m = 11 \quad n = 4$

$(P_1, P_2, P_3, P_4) = (60, 28, 20, 24)$

$(w_1, w_2, w_3, w_4) = (10, 7, 4, 2)$

$S^0 = \{(0,0)\} \quad S^0_1 = \{(60,10)\}$

$S^1 = \{(0,0),(60,10)\} \quad \overset{(28,7)}{S^1_1} = \{(28,7)(88,17)\}$

$S^2 = \{(0,0)(28,7)(60,10)(\cancel{88,17})\}$

$S^2_1 = \overset{(20,4)}{\{(20,4)(48,11)(\cancel{80,14})(\cancel{108,21})\}}$

$S^3 = \{(0,0)(28,7)(20,4)\underline{(60,10)}\}$ [illegible]

$S^3_1 = \overset{(24,2)}{\{(24,2)(52,9)(44,6)(\cancel{84,12})\}}$

$S^4 = \{(0,0)(\cancel{20,4})(24,2)(28,7),\overset{(44,6)}{\wedge}(52,9),$
$\underline{\underline{(60,10)}}\}$

$x_4 = 0 \Rightarrow (60,10) \in S^3$

$x_3 = 0 \Rightarrow (60,10) \overset{}{\underset{\in}{\notin}} S^2$

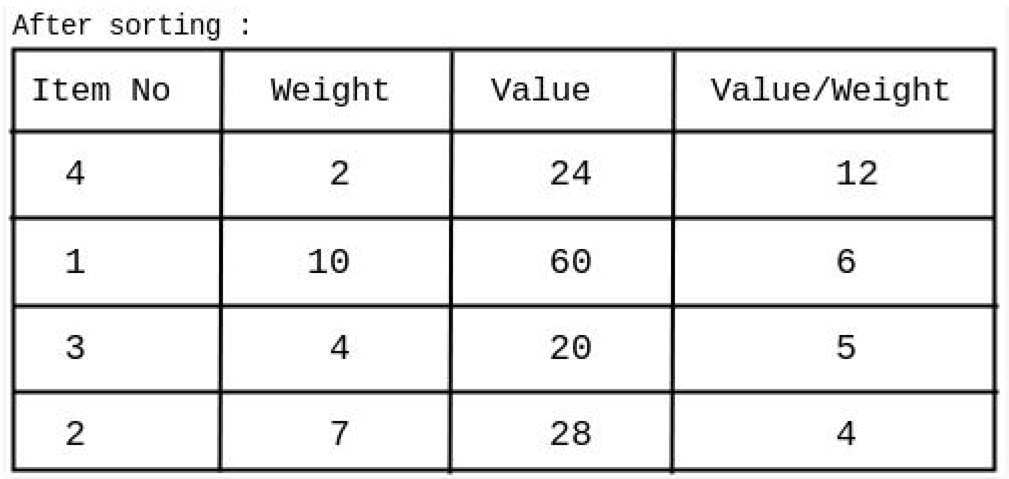
$\cancel{x}_2 = 0 \Rightarrow (60,10) \in S^1$

$x_1 = 1 \Rightarrow (60,10) \notin S^0.$

$(1, 0, 0, 0).$

$V_{opt} = 60.$

$V_{opt} - V_{greedy} = 60 - 44 = 16$



After sorting :

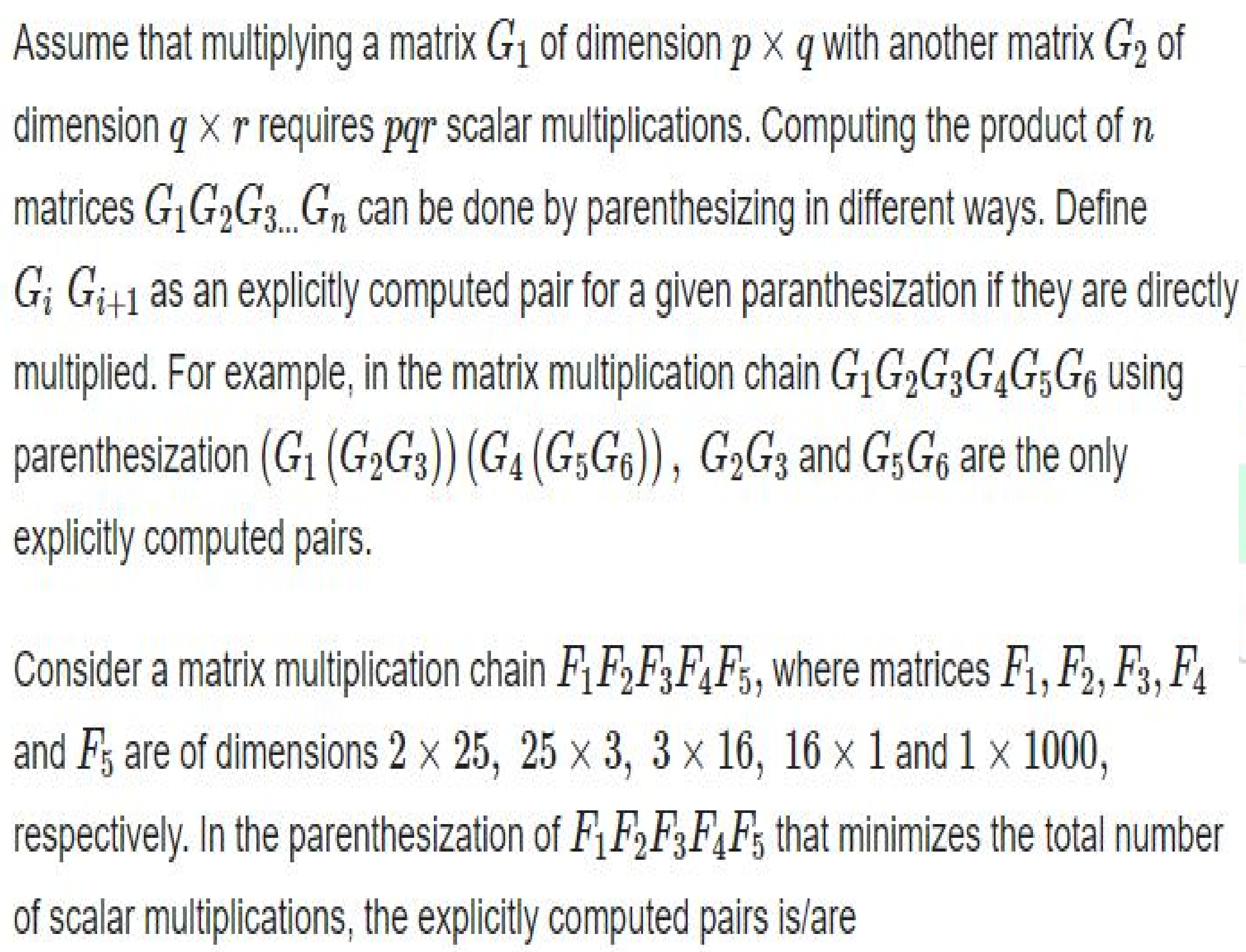
Item No	Weight	Value	Value/Weight
4	2	24	12
1	10	60	6
3	4	20	5
2	7	28	4

First we will pick item_4 (Value weight ratio is highest). Second highest is item_1, but cannot be picked because of its weight. Now item_3 shall be picked. item_2 cannot be included because of its weight.

Therefore, overall profit by V_{greedy} = 20+24 = 44

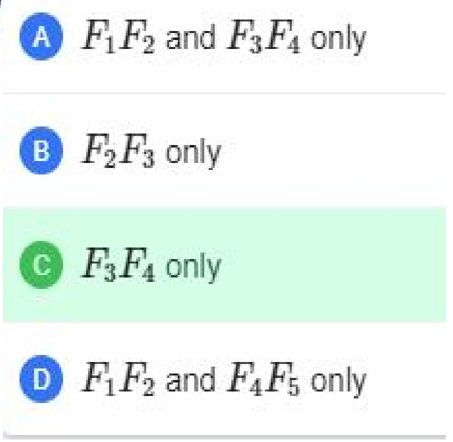
GATE CSE 2018

2.



Assume that multiplying a matrix G_1 of dimension $p \times q$ with another matrix G_2 of dimension $q \times r$ requires pqr scalar multiplications. Computing the product of n matrices $G_1 G_2 G_{3...} G_n$ can be done by parenthesizing in different ways. Define $G_i \; G_{i+1}$ as an explicitly computed pair for a given paranthesization if they are directly multiplied. For example, in the matrix multiplication chain $G_1G_2G_3G_4G_5G_6$ using parenthesization $(G_1(G_2G_3))(G_4(G_5G_6))$, G_2G_3 and G_5G_6 are the only explicitly computed pairs.

Consider a matrix multiplication chain $F_1F_2F_3F_4F_5$, where matrices F_1, F_2, F_3, F_4 and F_5 are of dimensions $2 \times 25,\; 25 \times 3,\; 3 \times 16,\; 16 \times 1$ and 1×1000, respectively. In the parenthesization of $F_1F_2F_3F_4F_5$ that minimizes the total number of scalar multiplications, the explicitly computed pairs is/are



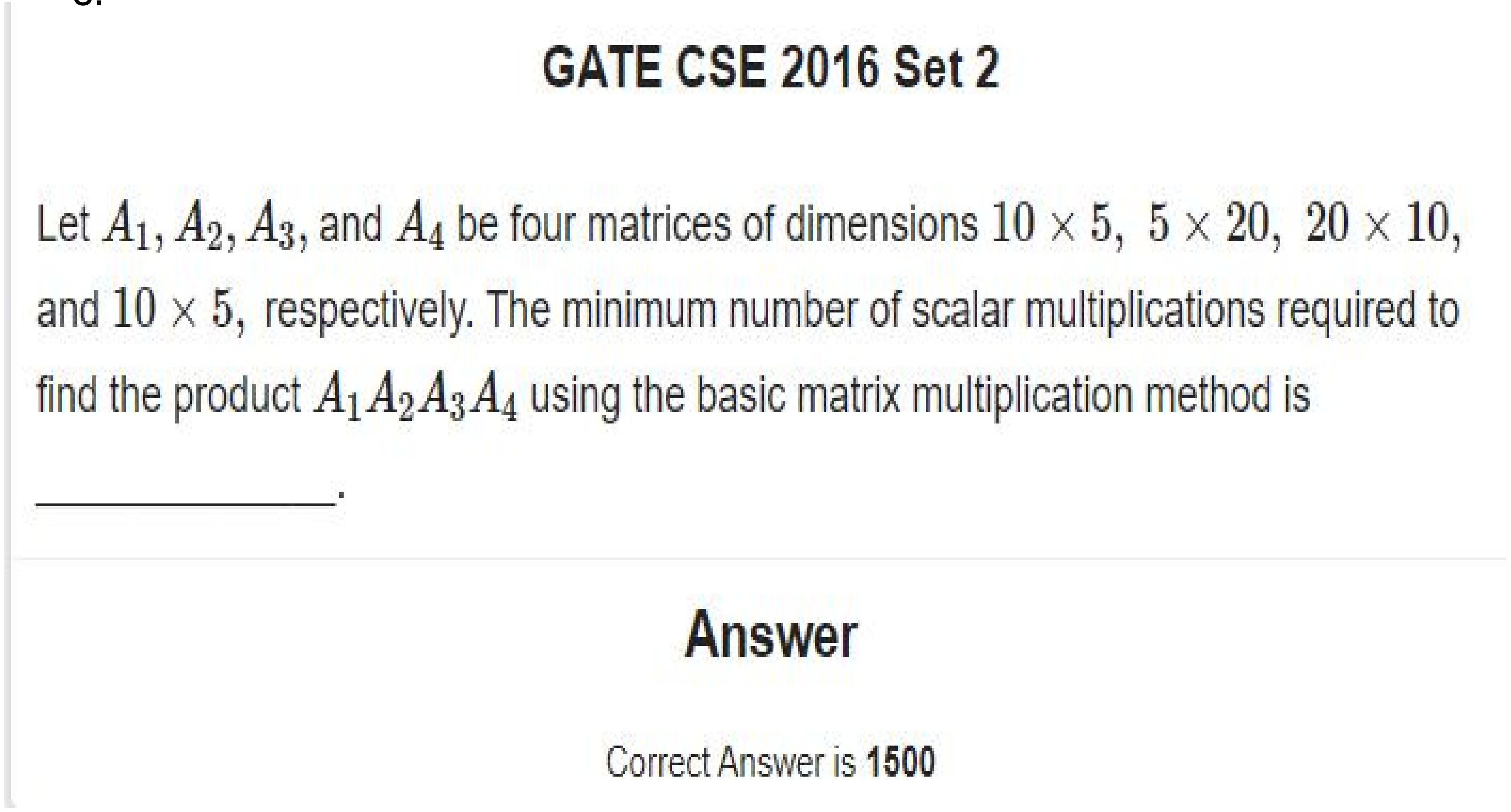
A F_1F_2 and F_3F_4 only

B F_2F_3 only

[✓ C F_3F_4 only]

D F_1F_2 and F_4F_5 only

3.


GATE CSE 2016 Set 2

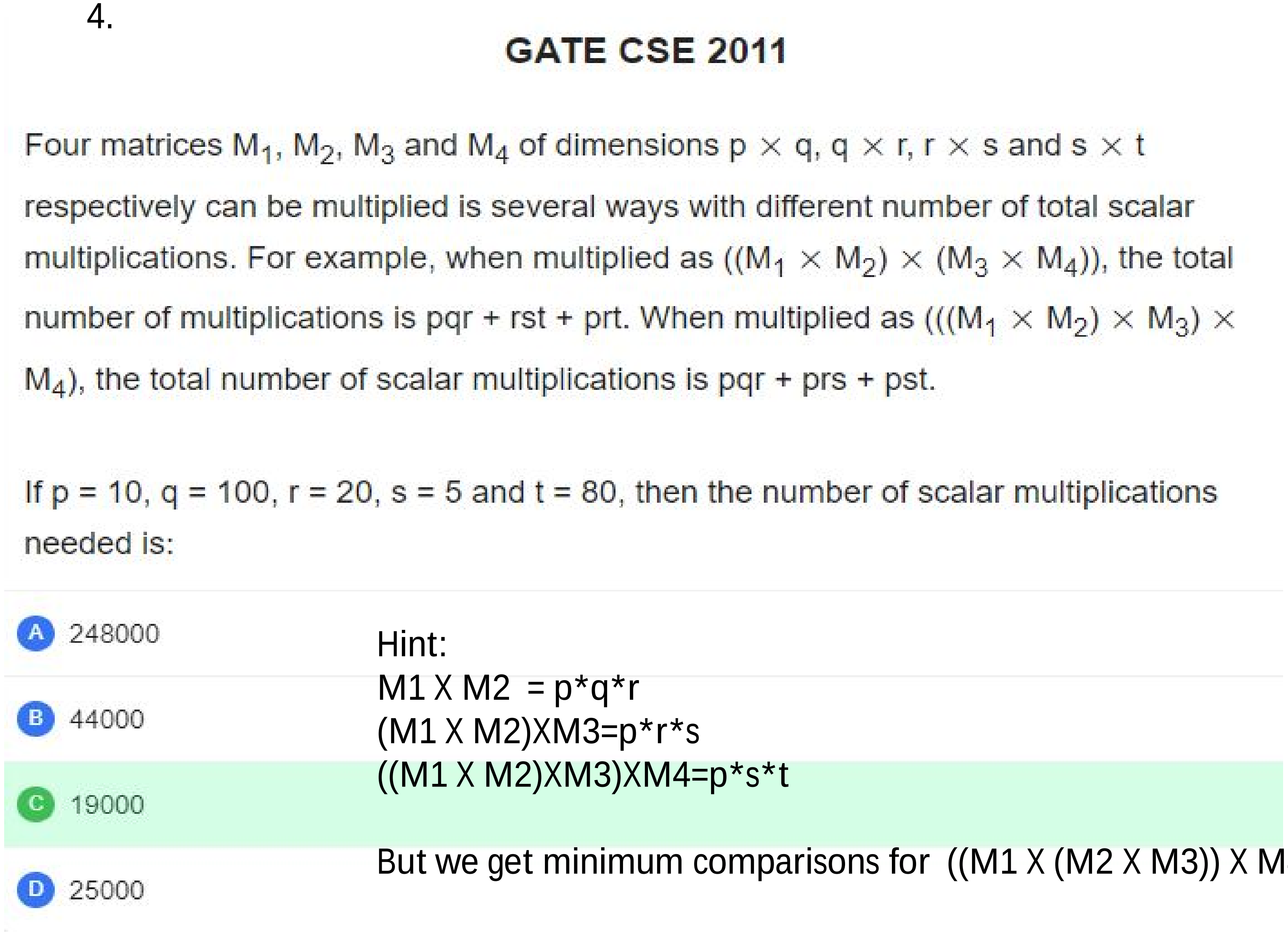
Let A_1, A_2, A_3, and A_4 be four matrices of dimensions $10 \times 5,\ 5 \times 20,\ 20 \times 10$, and 10×5, respectively. The minimum number of scalar multiplications required to find the product $A_1A_2A_3A_4$ using the basic matrix multiplication method is ______________.

Answer

Correct Answer is **1500**

((A1A2)A3)A4 = ((A1(A2A3))A4) = (A1A2)(A3A4) = **A1((A2A3)A4)** = A1(A2(A3A4)).

4.



GATE CSE 2011

Four matrices M_1, M_2, M_3 and M_4 of dimensions $p \times q$, $q \times r$, $r \times s$ and $s \times t$ respectively can be multiplied is several ways with different number of total scalar multiplications. For example, when multiplied as $((M_1 \times M_2) \times (M_3 \times M_4))$, the total number of multiplications is pqr + rst + prt. When multiplied as $(((M_1 \times M_2) \times M_3) \times M_4)$, the total number of scalar multiplications is pqr + prs + pst.

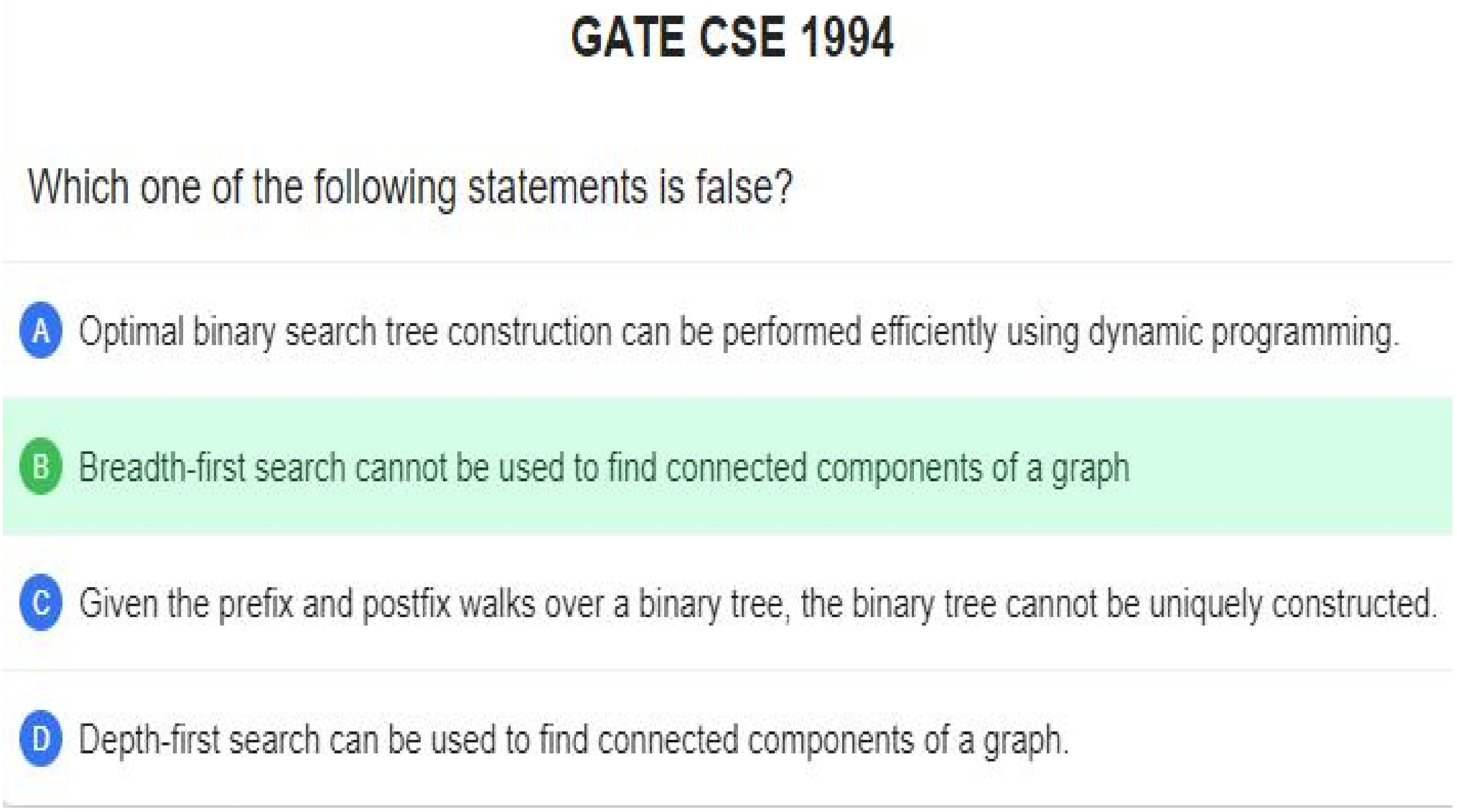
If p = 10, q = 100, r = 20, s = 5 and t = 80, then the number of scalar multiplications needed is:

- A 248000
- B 44000
- [✓ C 19000]
- D 25000

Hint:
M1 X M2 = p*q*r
(M1 X M2)XM3=p*r*s
((M1 X M2)XM3)XM4=p*s*t

But we get minimum comparisons for ((M1 X (M2 X M3)) X M4).

5.



GATE CSE 1994

Which one of the following statements is false?

- A Optimal binary search tree construction can be performed efficiently using dynamic programming.
- [✓ B] Breadth-first search cannot be used to find connected components of a graph
- C Given the prefix and postfix walks over a binary tree, the binary tree cannot be uniquely constructed.
- D Depth-first search can be used to find connected components of a graph.

Thank You