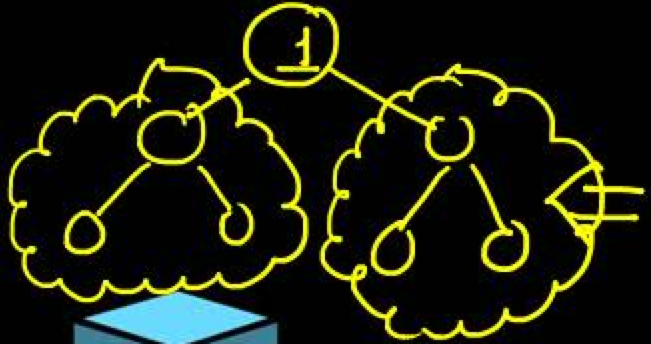


group

Original 2015

1. Question Related with shortest path Negative ~~end~~ edge
2. Lower bound on Heap Converting in to Heap.
3. Integer Multiplication

0/1 Knapsack

Knapsack.

M

object are given associated with profil.

maximize the profil.. capacity of Bag was the constrains

Problem Statement

Greedy Method — x — x — knapsack problem (Fractional knapsack)

Binary knapsack : ~~eiter~~ either object will be picked completely or it will Not be picked.

problem statement : find the filling of knapsack that maximize the ~~total~~ profit $\left(\sum_{i \in N} p_i\right)$

Constraints is : $\sum_{1 \le i \le n} w_i \le M$

$w_i = 0$ or $w_i = 1$

Problem Statement

- In this case
- Given: A set S of n items, with each item i having
  - w_i – a positive weight
  - p_i – a profit
- Goal: Choose items with maximum total profit but with weight at most W.
- If we are **not** allowed to take fractional amounts, then this is the **0/1 knapsack problem**.

Problem Statement

Problem Statement

- We need to find the filling of the knapsack such that maximum profit can be earned.

Problem Statement

- Let T denote the set of items we take
- Objective: maximize $\sum_{i \in T} p_i$
- Constraint: $\sum_{i \in T} w_i \leq \cancel{W}$ M

Example

- Capacity of the bag M = 8
- Number of object N =4
- The profit vector ,P={1,2,5,6}
- The weight vector, W={2,3,4,5}

Θ' ⇑

Brute force : How many
Combination i need to check

. optimal substructure
. Subproblem
. Overlapping Subproblem
. optimization.

Standard Dynamic programming approach is tabulation Method / Bottom up approach.

Memorization : Top down Recursive approach with table (array / Hashtable

→ Searching ⇐ Hashing

Brute Force

Maximum profit can be earned by inclusion or exclution of an object.

$\{O_1, O_2, O_3, O_4\}$ ⟶ All possible <u>subset</u>

$$n_{C_0} + n_{C_1} + n_{C_2} + n_{C_3} + \cdots + n_{C_n} = 2^n$$

Exponential tim Algorithm

Brute Force

- Checking all possible combination we need to check all Subsets i.e. 2^n

Dynamic Programming Solution Bottom Up Approach

Dynamic Programming

only one ~~object~~ copy of ~~eeo~~

Capacity of knapsack ⟶ max (

Increasing weight(→)		ob(→)	0	1	2	3	4	5	6	7	8
Pi	Wi	ob(i↓) 0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0	0	1	2	5	6			

Dynamic Programming

Increasing weight(→)		ob(→)	0	1	2	3	4	5	6	7	8
Pi	Wi	ob(i↓)	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6 P_k	5 w_k	4	0	0	1	2	5	6			

$$B[4,5] = \max\left(B[4-1, 5], \underbrace{B[4-1, 5-w_k] + P_k}_{0+6}\right)$$

$$(5, \; 0 + 6) = 6$$

Dynamic Programming

Increasing weight(→)		ob(→)	0	1	2	3	4	5	6	7	8
Pi	Wi	ob(i↓)	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6 p_k	5 w_k	4	0	0	1	2	5				

$$B[i, w] = \max\{B[i-1, w],\ B[i-1, w-w_k] + p_k\}$$

$$B[4,5] = \max\{B[3,5],\ B[3,5-5] + 6\} = \max\{5,\ 0+6\} = \max\{5,\ 6\} = 6$$

Dynamic Programming

Increasing weight(→)		ob(→)	0	1	2	3	4	5	6	7	8
Pi	Wi	ob(i↓)	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6 p_k	5 w_k	4	0	0	1	2	5	6	6		

1
$\boxed{11}$
w = 4
$p_i \cdot 20$
w(11-4)
20

B[4,6] = max (B[3,6], B[3, 6-5] + 6) = (6, 0+6) = 6, 6 = 6

$B[i, w] = \max\{B[i-1, w],\ B[i-1, w-w_k] + p_k\}$

$B[4,6] = \max\{B[3,6],\ B[3,6-5]+6\} = \max\{6,\ 0+6\} = \max\{6,\ 6\} = 6$

Dynamic Programming

Increasing weight(→)		ob(→)	0	1	2	3	4	5	6	7	8
Pi	Wi	ob(i↓)	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6 p_k	5 w_k	4	0	0	1	2	5	6	6	7	

B[4,7] = max (B[3,7], B[3,7-5] + 6) = (7, 1+6) = (7,7) = 7

$B[i, w] = \max\{B[i-1, w],\ B[i-1, w-w_k] + p_k\}$

$B[4,7] = \max\{B[3,7],\ B[3,7-5] + 6\} = \max\{7,\ 1+6\} = \max\{7,\ 7\} = 7$

Dynamic Programming

Increasing weight(→)		ob(→)	0	1	2	3	4	5	6	7	8
Pi	Wi	ob(i↓)	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	$\boxed{5}$	6	7	7
6 p_k	5 w_k	4	0	0	1	2	5	$\boxed{6}$	6	7	8

$$\left(B[i,w] = \max\{B[i-1,w],\ B[i-1,w-w_k]+p_k\}\right)$$ — topdown Approach

$$B[4,8] = \max\{B[3,8],\ B[3,8-5]+6\} = \max\{7,\ 2+6\} = \max\{7,\ 8\} = 8$$

Structure of Optimality & Recursive Solution

$$B[i,w] = \begin{cases} \underline{B[i-1,w]} & , \underline{w} < \underline{w_i} \\ \max(\underbrace{B[i-1,w]}_{\text{optimal}}, B[i-1, w-w_i] + \underline{P_i}) & \underline{w} \end{cases}$$

$w = \underline{3}$

$w_k = \underline{4}$

w_i, P_i is the object in consideration

$\cdot\ \cancel{w_i = w_k}\ ,\ \underline{\cancel{P_k = P_i}}$

Structure of Optimality & Recursive Solution

$$B[i,w] = \begin{cases} B[i-1,w] & if\, w_k > w \\ \max\{B[i-1,w],\ B[i-1,w-w_k]+p_k\} & else \end{cases}$$

Example

Find the optimal solution for 0/1 Knapsack problem?

Capacity of the bag M = 11

Number of object, N = 5

Profit vector, P={1,6,18,22,28}

Weight vector, W={1,2,5,6,7}

Dynamic Programming

Increasing weight(→)		ob(→)	0	1	2	3	4	5	6	7	8	9	10	11
Pi	Wi	ob(i↓)	0	0	0	0	0	0	0	0	0	0	0	0
$\boxed{1}$	1	1	0	1	1	1	1	1	1	1	1	1	1	1
6	2	2	0	1	6	7	7	7	7	7	7	7	7	7
18	5	3	0	1	6	7	7	18	19	24	25	25	25	25
22	6	4	0	1	6	7	7	18	22	$\boxed{24}$	28	29	29	40
28	7	5	0	1	6	7	4	5	22	28	29	34	35	40

$B[5,7] = \max(\underset{24}{B[4,7]},\ B[4,7-7] + 28)$

$40,\ 28+7$

$29,\ 28+7$

$29,\ 28+6$

$29,\ 34$

Weight Limit (i):	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$W_1 = 1$ $V_1 = 1$	0	1	1	1	1	1	1	1	1	1	1	1
$W_2 = 2$ $V_2 = 6$	0	1	6	7	7	7	7	7	7	7	7	7
$W_3 = 5$ $V_3 = 18$	0	1	6	7	7	18	19	24	25	25	25	25
$W_4 = 6$ $V_4 = 22$	0											
$W_5 = 7$ $V_5 = 28$	0											

$$B[k, w] = \begin{cases} B[k-1, w] & \text{if } w_k > w \\ \max\{B[k-1, w],\ B[k-1, w - w_k] + b_k\} & \text{else} \end{cases}$$

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	6
2	7	28	4
3	4	20	5 ←
4	2	24	12 ←

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}.

M = 5

The value of V_{opt} - V_{greedy} is ______ $60 - 44 = \boxed{16}$

$V_{optimal}$ – 60

V_{greedy} = 44

agree with ye object

w. 2

P = 24

M = 9

I can't pick object 1

M = 9

w = 4

P = ~~24~~ 20

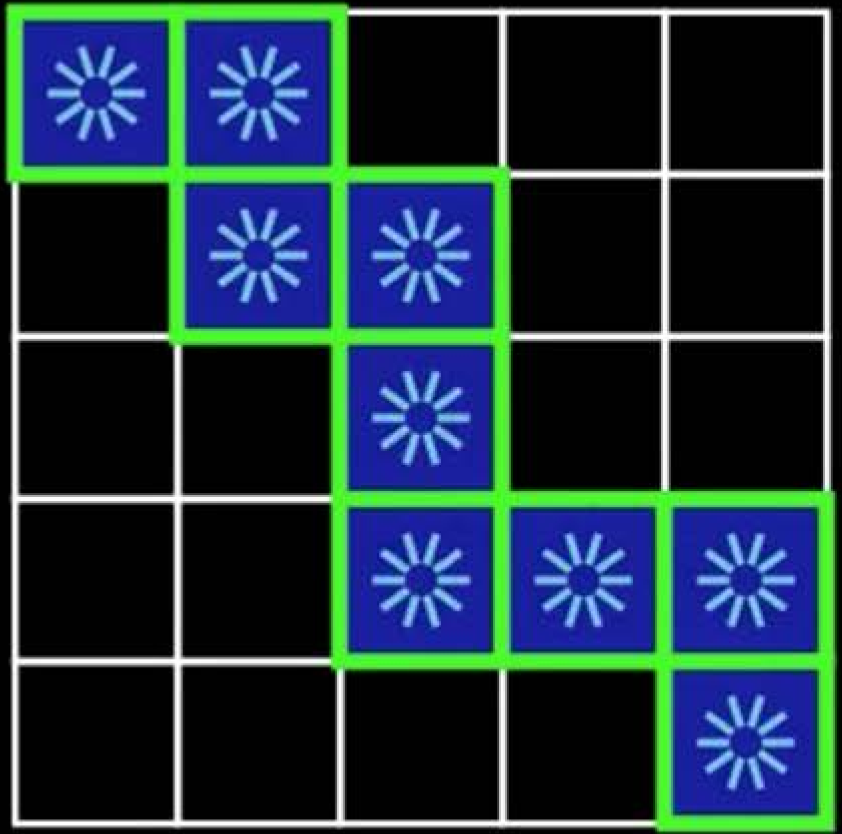
M = 9 - 4 = 5

profit 20 + 24 = 44

0/1 Knapsack Algorithm

Algorithm

```
Dynamic-0-1-knapsack (n, m, w[], p[])
B[n+1,m+1]
for i = 0 to n do
     for j = 0 to m do
        if (i =0 or j=0)
              B[i][j]=0;
        else        6      8
           if ( w[i] ≤ j)
              B[i][j]=max{B[i-1,j], B[i-1][j-w[i]]+p[i]}
           else
              B[i][j] = B[i-1][j];
     }
```



Longest Common Subsequence

Longest Common Subsequence

DNA matching

DNA structure of an organism is made of 4 bases called as

- Adenine
- guanine
- Cyctocine
- Thymine

- Any DNA structure is defind as: A sequence of A, G, C, T

Longest Common Subsequence

- ***Biological applications often need to compare the DNA of two (or more) different organisms***. A strand of DNA consists of a string of molecules called ***bases***, where the possible bases are adenine, guanine, cytosine, and thymine.

Strand of DNA

- A string of A,G,C,T Reprents DNA for a orgarism.

Given 2 DNA structure we want to find the Longest common Subsequence in 2 DNA. that decides How Similar two specisis are

Longest Common Subsequence

- Representing each of these bases by its initial letter, we can express a strand of DNA as a string over the finite set {A; C; G; T}.

Longest Common Subsequence

- For example, the DNA of one organism may be S1 = ACCGGTCGAGTGCGCGGAAGCCGGCCGAA, and the DNA of another organism may be
- S2 = GTCGTTCGGAATGCCGTTGCTCTGTAAA.
- One reason to compare two strands of DNA is to determine how "similar" the two strands are, as some measure of how closely related the two organisms are.

Subsequence

X ={A; B;C; B;D;A;B}
1 2 3 4 5 6 7

- Subsequence
- Common Subsequence

Subsequence is ~~the~~ character in ~~inch~~ increasing order of index.

Subsequence is - ABC, ABD (1 4 5), AAB (1 6 7)

Subsequence

- Characters in increasing order of indices
- For example, Z = {B; C; D; B} is a subsequence of

  X ={A; B;C; B;D;A;B} with corresponding index sequence

  {2; 3; 5; 7}

Subsequence

- Characters in increasing order of indices
- For example, Z = {B; C; D; B} is a subsequence of

  X ={A; B;C; B;D;A;B} with corresponding index sequence

  {2; 3; 5; 7}

Common Subsequence

- Given two sequences X and Y , we say that a sequence Z is a common subsequence of X and Y if Z is a subsequence of both X and Y . For example, if X = {A; B;C; B;D;A;B} and Y = {B;D;C;A; B;A},

Common Subsequence

Question : find common Subsequence of Length 4

- Length 4:

~~bdab~~
bcab
BCBA

X = {A, B, C, B, D, A, B}

Y = {B D C A, B, A}

Common Subsequence

- Given two sequences X and Y , we say that a sequence Z is a common subsequence of X and Y if Z is a subsequence of both X and Y . For example, if X = {A; B;C; B;D;A;B} and
  Y = {B;D;C;A; B;A},

- Length 4: BCAB, BDAB, BCBA

Common Subsequence

- Given two sequences X and Y , we say that a sequence Z is a common subsequence of X and Y if Z is a subsequence of both X and Y . For example, if X = {A; B;C; B;D;A;B} and Y = {B;D;C;A; B;A},
- **Length 3:** The sequence {B;C;A} is a common subsequence of both X and Y . The sequence {B;C;A} is not a longest common subsequence (LCS) of X and Y , however, since it has length 3 and the sequence {B;C; B;A}, which is also common to both X and Y , has length 4.
- **Length 4:** The sequence {B; C; B; A} is an LCS of X and Y , as is the sequence {B; D; A; B}, since X and Y have no common subsequence of length 5 or greater.

Problem Statement

Given two subsequence, $X = x_1 x_2 x_3 ... x_n$ and $Y = y_1 y_2 y_3 .. y_m$, the problem is to find the length maximum common subsequence.

Problem Statement:

- In the longest-common-subsequence problem, we are given two sequences X = {x1; x2;….. ; xm} and Y = {y1; y2; …..; yn} and wish to find a maximum length common subsequence of X and Y .

Gate 2014 Set-I

- Consider two strings A = "qpqrr" and B = "pqprqrp". Let x be the length of the longest common subsequence (not necessarily contiguous) between A and B and let y be the number of such longest common subsequences between A and B. Then $x + 10y =$ ___.

$4 + 10 \times 3 = \underline{34}$

A = qpqrr

B = pqprqrp

pqrr ✓

qqrrq

qprr ✓

qpqr ✓

Legth 5 is Not possible

Legth as 4
NO. as 3

Smaller: Every Algorithmic problem is aptitude question

Structure of Optimality & Recursive Solution

Hop down

- LCS("ABBA$\boxed{B}$", "ACBAB") 4

— there Last alphabet matches?

Subproblem — then where you will search?

$\Downarrow$ 3

LCS(ABBA, ACBA) + $\boxed{1}$ ← you got match

$\Downarrow$ 2

$\boxed{ACB}$ CBD

LCS(ABB, ACB) + 1 max(LCS(ACB, CB), LCS(AC, CBD))

1 $\Downarrow$

LCS(AD, AC) + 1

LCS(AB, A)

1 1

max(LCS(AB, A), LCS(A, AC))

max(LCS(A, A), LCS(AD, ∅))

Structure of Optimality & Recursive Solution

$$LCS[X_{\underline{i}}, Y_j] = \begin{cases} \underline{0} & \text{if } i = 0 \text{ or } j = 0 \\ LCS[X_{i-1}, Y_{j-1}] + \underline{1}, & \text{if } \underline{X_i = Y_j} \\ \max(LCS[X_{i-1}, Y], LCS[X_i, Y_{j-1}]) & \text{if } X_i \neq Y_j \end{cases}$$

~~No~~ string length is 0

$\underline{i, j}$ Last $\underline{\text{index}}$

Structure of Optimality & Recursive Solution

$$c[i;\ j] = \begin{cases} 0 & if\ i = 0\ \ or\ \ j = 0 \\ c[i-1, j-1] + 1 & i, j > 0\ \ and\ \ x_i = y_i \\ \max(c[i, j-1], c[i-1, j]) & i, j > 0\ \ and\ \ x_i \neq y_i \end{cases}$$

Structure of Optimality & Recursive Solution

- Let $X = \{x_1; x_2; \ \ldots.. \ ; x_m\}$ and $Y = \{y_1; y_2; \ \ldots\ldots. \ ; y_n\}$ be sequences, and let $Z = \{z_1; z_2; \ \ldots\ldots; z_k\}$ e any LCS of X and Y.

- 1. If $x_m = y_n$, then $z_k = x_m = y_n$ and z_{k-1} is an LCS of X_{m-1} and Y_{n-1}.

- 2. If $x_m \neq y_n$, then $z_k \neq x_m$ implies that Z is an LCS of X_{m-1} and Y.

- 3. If $x_m \neq y_n$, then $z_k \neq y_n$ implies that Z is an LCS of X_m and Y_{n-1}.

Computing the length of an LCS

LCS-Length(X, Y)

~~Length~~ direction

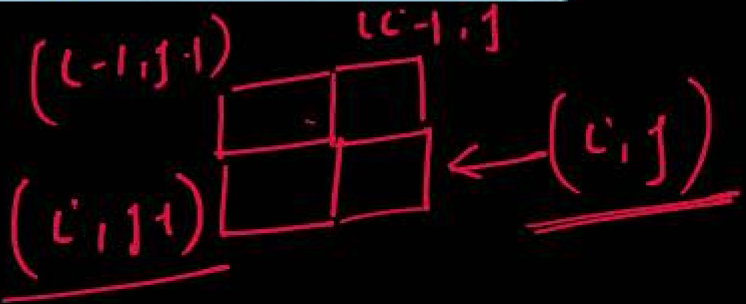
1. m = X. length
2. n = Y. length
3. let b[1.. m. 1.. n] and $\underbrace{\text{c[0.. m, 0.. n]}}_{\text{Length}}$ be new tables

direction

4. for i = 1 to m
5.   c[i,0] = 0
6. for j = 0 to n
7.   c[0,j] = 0

~~8. for i = 1 to m~~

Computing the length of an LCS




```
for i = 1 to m
    for y = 1 to n
        if xi == yj
        c[i,j] = c[i-1,j -1] + 1
        b[i,j]= "↖"
        elseif c[i − 1,j]≥ c[i,j − 1]
        c[i,j] = c[i-1,j]
        b[i.j] = "↑"
        else c[i,j] = c[i,j − 1]
        b[i,j] = "←"
      return c and b
```

Match

defaul for equality is upword in Algorithm

Example

Dynamic Programming

Bottom up approach

with one

The Length of maximum Common Subsequence is 4

BCAB

BDAB

BCBA

CAB

BDAB

	J	0	1	2	3	4	5	6
i		Yi	B	D	C	A	B	A
0	Xi	0	0	0	0	0	0	0
1	A	0	0↑ ←	0↑ ←	0↑ ←	1↖	1 ←	1↖
2	B	0	1↖	1 ←	1 ←	1↑ ←	2↖	2 ←
3	C	0	1↑	1↑ ←	2↖	2 ←	2↑ ←	2↑ ←
4	B	0	1↖	1↑ ←	2↑	2↑ ←	3↖	3 ←
5	D	0	1↑	2↖	2↑ ←	2↑ ←	3↑	3↑ ←
6	A	0	1↑	2↑	2↑ ←	3↖	3↑ ←	4↖
7	B	0	1↖	2↑	2↑ ←	3↑	4↖	4↑ ←

Matrix Chain Multiplication

Matrix Multiplication

- Iterative algorithm
- Recursive version

- Chain of compitable matrices given

Matrix Multiplication

- On multiplying two Matrices of size $p \times q$ and $q \times r$, number of scalar matrix multiplication possible is

$$p \times q \times r$$

Matrix Chain Multiplication

- consider the problem of a chain $\{A_1. A_2 A_3\}$ of three matrices. Suppose that the dimensions of the matrices are 10×100, 100×5, and 5×50, respectively. What is the number of scalar product for different parenthesization. How many ways to we can do the parenthesization.

A_1 A_2 A_3

10×100 100×5 5×50

$(A_1 (A_2 A_3))$

$((A_1 A_2) A_3)$

How many different ways we can multiply

$A_1, A_2 A_3$

Matrix Chain Multiplication

- 10×100, 100×5, and 5×50,

$$(A_1 \cdot (A_2 \cdot A_3))$$

$$((A_1 \cdot A_2) A_3)$$

$$n \times m \quad m \times p \quad - \quad (n \times m \times p)$$

No. of Scalar multiplication

$$(n \times m \times p)$$

Matrix Chain Multiplication

- Our next example of dynamic programming is an algorithm that solves the problem of matrix-chain multiplication. We are given a sequence (chain) $\{A_1;\ A_2; \ldots.\ ; A_n\}$ of n matrices to be multiplied, and we wish to compute the product $A_1;\ A_2; \ldots.\ ; A_n$:

What is the number of scalar multiplication

- $((A_1A_2).A_3)$ (I) — $\underbrace{10\times100,\ 100\times5}$ 5×50

  $10\times100\times5 = 5000$ multiplying $A_1 A_2$

  the dimension $A_1 \cdot A_2 = 10\times5$

  $(A_1 A_2) \cdot \cancel{5\times50}\ A_3$

  $10\times5 \qquad 5\times50 \Rightarrow$

  cost $10\times5\times50 = 2500$

  total $5000+2500 = 7500$

- $(A_1(A_2.A_3))$

$\Downarrow$

Cost of $A_2 A_3$

$100\times5\times50 = 25000$

Size dimension - 100×50

$A_1 (A_2 A_3)$

$\cancel{10\times100} \quad 100\times50 =$

cost $10\times100\times50 = 50000$

total $= 50,000+25000 = 75000$

Solution

- If we multiply according to the parenthesization $((A_1A_2).A_3)$ we perform 10 • 100 • 5 = 5000 scalar multiplications to compute the 10 x 5 matrix product A_1A_2, plus another 10 . 5 . 50 = 2500 scalar multiplications to multiply this matrix by *A3*, for a total of 7500 scalar multiplications.

Solution

Optimal

- If instead we multiply according to the parenthesization $(A_1(A_2.A_3))$ we perform 100 • 5 • 50 = 25,000 scalar multiplications to compute the 100 x 50 matrix product A_2A_3, *plus* another 10 • 100 • 50 = 50,000 scalar multiplications to multiply A_1 by this matrix, for a total of 75,000 scalar multiplications. Thus, computing the product according to the first parenthesization is 10 times faster.

Fully parenthesized

- How many different ways we can parenthesized $A_1 A_2 A_3 A_4$

Counting the number of parenthesizations

10^9

$$\left(\underline{23|45} \times 3652\right)$$

n

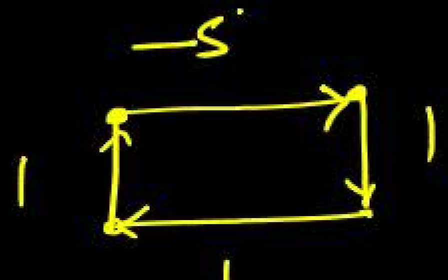
$A = m = n/2$

$$(23 \times \underline{10}^2 + 45)\,(36 \times 10^2 + 52)$$

$$T(n) = 4T(n/2) +$$

$$(\underbrace{w \times 10^m + x)\,(y}\times 10^m + z)$$

$$(\underbrace{wy}_{n/2} \times 10^{2m} + \underbrace{wz}_{n/2} \times 10^m + \cancel{w}\,\underbrace{xy}_{n/2} \times 10^m + \underbrace{xz}_{n/2})$$



$= -2$ Negative