30/07/2019

Analysis and design of algorithms. (IT-C17)

Books - Algorithms (CLRS) - by cormen PHI publication.
- Algorithms - by Hurwitz Sahni

Algorithm :

- Well defined computational procedure.
- Finite set of instructions that, if followed accomplishes a particular task.

Characteristics of an algorithm :-

1- Input (I/p)
2- Output (O/P)
3 - Definiteness - Each instruction is clear and un-ambiguous.
4 - Finiteness
5 - Effectiveness. - correctness + efficiency

Expressing algorithm :

1- English
2- Pseudo code
3 - Real programming language

Analysis of algorithm :

1- Space complexity
2- Time complexity
   a- Worst case complexity.
   b - Best case complexity.
   c - Average case complexity.

Scanned by CamScanner

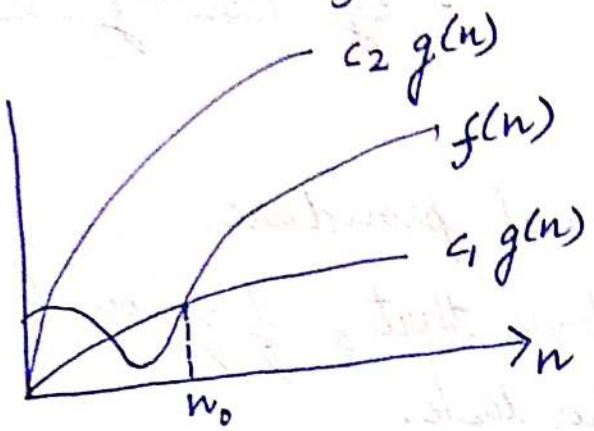
Growth of functions :

Θ-notation :

$f(n) = \Theta(g(n))$: there exist positive constants c_1, c_2 and 'n' such that

$$\boxed{0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)}$$

{for all $n \geq n_0$}



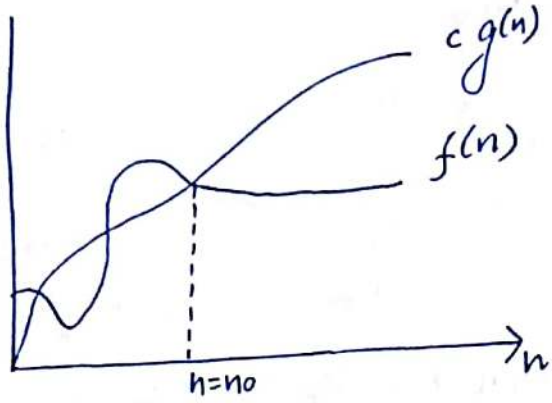
We say that, $g(n)$ is an asymtotically tight bound for $f(n)$.

'Θ' bounds a function from above and below.

Scanned by CamScanner

1/08/2019

Big oh notation (O)



- $f(n) = O(g(n))$, there exists positive constants c and n_0. such that

$$0 \le f(n) \le c\,g(n)$$

for all $n \ge n_0$

- O. notation gives the upper bound on a function to within a constant factor.

If $f(n) = \Theta(g(n))$

$\Downarrow$

$f(n) = O(g(n))$

* used to represent worst case time complexity of an algorithm/ function

Ω - notation - [Big omega notation]

$f(n) = \Omega(g(n))$. There exists +ve constant c and n_0, such that:

$$0 \le c \cdot g(n) \le f(n) \quad \forall\ n > n_0$$

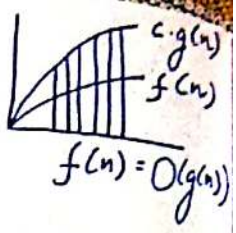
Lower bound $\Rightarrow$ best case

if $f(n) = \Theta(g(n))$ $\Downarrow$

$f(n) = O(g(n))$ $\Downarrow$

$f(n) = \Omega(g(n))$

Scanned by CamScanner



① $\overset{f(n)}{n} \in O(\overset{g(n)}{n^2})$

② $100n + 5 \in O(n^2)$

③ $\frac{1}{2}n(n-1) \in O(n^2)$

④ $n^3 \in O(n^2)$

⑤ $n^4 + n + 1 \in O(n^2)$

⑥ $n^3 \in \Omega(n^2)$

⑦ $\frac{1}{2}n(n-1) \in \Omega(n^2)$

⑧ $100n + 5 \in \Omega(n^2)$

<u>Qo.</u> $f(n) = \frac{1}{2} \cdot n^2 - 3 \cdot n$

Find the values of (c_1, c_2 and n_0), also Θ notation for $f(n)$

Sol^n: Let, $g(n) = n^2$

$$c_1 g(n) \le f(n) \le c_2 \cdot g(n)$$

$$c_1 n^2 \le \frac{1}{2} \cdot n^2 - 3n \le c_2 n^2$$

$\Downarrow$ $c_2 \ge \frac{1}{2} - \frac{3}{n} \rightarrow c_2 \ge \frac{1}{2}$ $\quad c_2 > 0$

$c_1 \le \frac{1}{2} - \frac{3}{n} \leftarrow \boxed{n=7}$

$c_1 \le 1/14$; $c_2 \ge \frac{1}{2}$; $\Theta(n^2)$

Scanned by CamScanner

Q02 If $f(n) = a_0 + a_1 n + a_2 n^2 + \cdots\cdots a_m n^m$ is a polynomial of degree m or less than n

$$\Rightarrow \quad f(n) = \Theta(n^m)$$

Q02 Let $f(n)$ and $g(n)$ are two positive functions.

Prove that :

$$\underbrace{\boxed{\max(\underbrace{f(n)}_{f(n)}, g(n))}}_{h(n)} = \Theta(\underbrace{f(n) + g(n)}_{g(n)})$$

Solution: $c_1(f(n)+g(n)) \le h(n) \le c_2 \cdot (f(n)+g(n))$

Let, $h(n) = \max(f(n), g(n))$ *MAX function

$$\therefore \quad h(n) = \begin{cases} f(n) & : \text{if } f(n) \ge g(n) \\ g(n) & : \text{if } f(n) < g(n) \end{cases}$$

$\because$ $f(n)$ and $g(n)$ are non-negative. So, there exist n_0 for which $f(n) \ge 0$ and $g(n) \ge 0$ for all $n \ge n_0$.

$\therefore$ For $n \ge n_0$

$$f(n) + g(n) \ge f(n) \ge 0 \text{ and}$$

$$f(n) + g(n) \ge g(n) \ge 0$$

But, $h(n)$ can be any of $f(n)$ or $g(n)$, for a particular value of n.

Scanned by CamScanner

$\therefore, \quad f(n) + g(n) \geq h(n) \geq 0$

$h(n) = \max(f(n), g(n)) \leq c_2 \, (f(n) + g(n)),$

where $c_2 = 1$ for all $n \geq n_0$

Similarly, for a particular value of n, $h(n)$ is larger of $f(n)$ or $g(n)$:

$\therefore,$ for all $n \geq n_0$

$$0 \leq f(n) \leq h(n) \quad \text{—— ①}$$

and $$0 \leq g(n) \leq h(n) \quad \text{—— ②}$$

adding ① & ②

$$0 \leq f(n) + g(n) \leq 2 \cdot h(n)$$

$$0 \leq \frac{1}{2}[f(n) + g(n)] \leq h(n)$$

$$\Rightarrow \boxed{c_1 = \frac{1}{2}}$$

Scanned by CamScanner

5/08/2019

Using limits for comparing orders of growth

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & ; f(n) \text{ has a smaller order of growth than } g(n) \\ c>0 & ; f(n) \text{ has same order of growth as } g(n). \\ \infty & ; f(n) \text{ has a larger order of growth than } g(n). \end{cases}$$

① The 1st two cases means that
$$f(n) \in O(g(n))$$

② The last case means that $f(n) \in \Omega(g(n))$

③ The second case means that
$$f(n) \in \Theta(g(n))$$

Qo. Show that $(n+a)^b = \Theta(n^b)$

Solution:
$$f(n) = (n+a)^b$$
$$g(n) = n^b$$

using method II, $\lim_{n\to\infty} \frac{(n+a)^b}{n^b} = \lim_{n\to\infty} \frac{(1+a/n)^b}{1}$

$$= 1 \quad ; \quad c>0$$

$$\Rightarrow f(n) \in \Theta(g(n))$$

Hence proved!!

OR using method I of finding constants

Scanned by CamScanner

method I,

We have to find c_1, c_2 and n_0 such that

$$0 \le c_1 \underbrace{(n^b)}_{g(n)^b} \le \underbrace{(n+a)^b}_{f(n)^b} \le c_2 \underbrace{(n^b)}_{g(n)^b} \text{ for all } n \ge n_0$$

$$n+a \le n+|a|$$

$$n+a \le 2n \quad \text{— ①} \quad \text{for } \underline{\underline{n \ge |a|}}$$

and $\quad n+a \ge n-|a|$

$$n+a \ge \frac{1}{2}n \quad \text{— ②} \quad \text{for } |a| \le \frac{1}{2}n$$

or $\underline{\underline{n \ge 2|a|}}$

$\therefore$, from ① and ②

for $n \ge 2|a|$

$$0 \le \frac{1}{2} \cdot n \le n+a \le 2n$$

Raising to the power b,

$$0 \le \left(\frac{1}{2} \cdot n\right)^b \le (n+a)^b \le (2n)^b$$

$$0 \le \left(\frac{1}{2}\right)^b n^b \le (n+a)^b \le 2^b (n)^b$$

$$c_1 = (1/2)^b$$

$$c_2 = 2^b$$

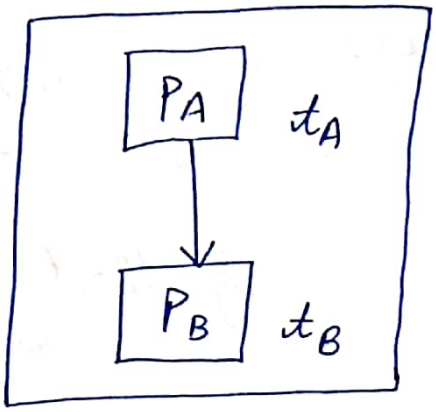
$$n_0 = 2|a|$$

- Important from exam point of view.
: Θ notation

Scanned by CamScanner

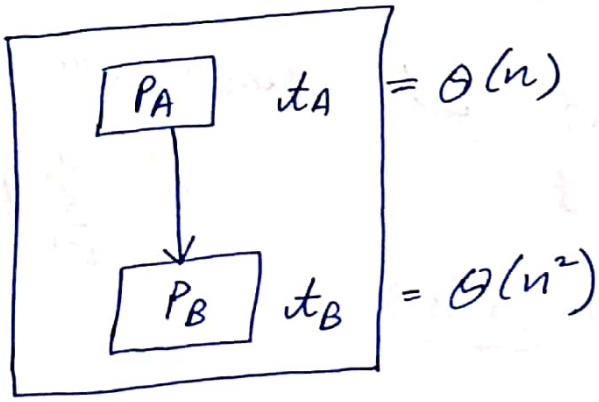
Analysing algorithm control structures

① Sequencing :

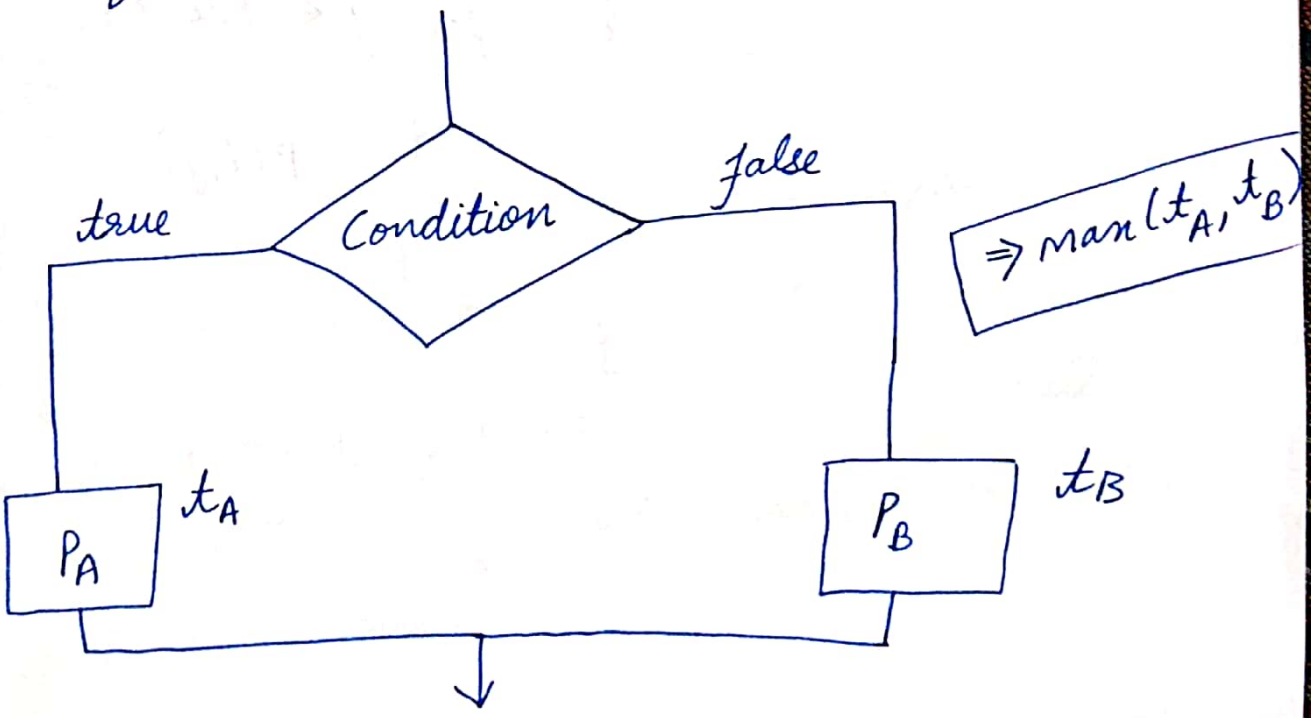


$t_A + t_B = \max(t_A, t_B)$

e.g. $t_A = \Theta(n)$ and $t_B = \Theta(n^2)$



② If-then-else



Scanned by CamScanner

③ <u>For-loops</u>

```
for i ← 1 to m
  {
      P(i)
  }
```

$\longrightarrow t$

$$\sum_{i=1}^{m} t_i \qquad \text{if } t_i = t \quad \forall i \in [1, m)$$

$$\Theta(\underline{mt})$$

<u>Q.</u>

```
for i ← 1 to m
  { set sum ← sum + A[i]
  }
```

$t_i = \Theta(1)$.

$$\sum_{i=1}^{m} t_i$$

<u>Ans:</u> $\underline{\Theta(m)}$ $\Rightarrow m$

<u>Nested-loops:</u>

```
for i ← 1 to m
  {   for j ← 1 to m
        {   P(i,j)
        }
  }
```

Scanned by CamScanner

8/8/2019

MASTER METHOD

For solving recurrences like : (No. of sub-problems to be solved.)

$$T(n) = a\,T(n/b) + f(n)$$, where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is a +ve function.

① If, $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$.

Then $\rightarrow T(n) = \Theta(n^{\log_b a})$

② If, $f(n) = \Theta(n^{\log_b a})$. Then, $T(n) = \Theta(n^{\log_b a} \log n)$

③ If, $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $a f(n/b) \leq c f(n)$ for some constant '$c < 1$' and all sufficiently large n, then

$$T(n) = \Theta(f(n)).$$

(i) a is the No. of problems solved recursively.

(ii) b is the size of each sub-problem.

(iii) n/b is the size of input to the recursive call.

(iv) $f(n)$ is the cost of dividing and re-combining the sub-problems.

Scanned by CamScanner

Qo. 1

$$T(n) = 9\,T(n/3) + n$$

Solⁿ

$\rightarrow$ $a = 9$
$b = 3$
$f(n) = n$

$$n^{\log_b a} = n^{\log_3 9} = n^2$$

$$f(n) = O\left(n^{\log_b a - \epsilon}\right) \quad \epsilon > 0$$

$$n = O\left(n^{2 - \epsilon}\right) \quad \epsilon = 0.2$$

type-1

condition holds true.

$$\boxed{T(n) = \Theta(n^2)}$$

Big Oh notation,
$f(n) = O(g(n))$
$\Rightarrow f(n) \le c\,g(n)$

Qo 2

$$T(n) = T(2n/3) + 1$$

$\rightarrow$ $a = 1$
$b = 3/2$
$f(n) = 1$

Solⁿ

$$n^{\log_b a} = n^{\log_{3/2} 1} = 1$$

$$f(n) = 1$$

$$\rightarrow f(n) = \Theta(1)$$

condition ② or type ②.

$$\boxed{T(n) = \Theta(\log n)}$$

Scanned by CamScanner

Q.3 $T(n) = 3T(n/4) + n\log n$

Sol^n

$a = 3$
$b = 4$
$f(n) = n\log n$

$n^{\log_a b} = n^{\log_4 3} = n^{0.793 \text{ (Approx)}}$

$f(n) = n\log n$ type-3.

$f(n) = \Omega(n^{\log_4 3 + \epsilon})$

$a f(n/b) = 3 \times \frac{n}{4} \log(n/4)$

$= \frac{3n}{4} \log(n/4) \leq \frac{3}{4} n\log n$

For large value of n

$c = 3/4$.

$$\boxed{T(n) = \Theta(n\log n)}$$

? D&C NP hard.

Q.4 $T(n) = 4T(n/2) + n^3$

$a = 4$
$b = 2$
$f(n) = n^3$

$n^{\log_b a} = n^{\log_2 4}$ type-3.

$4f(n/2) = \frac{\cancel{4} n^3}{\cancel{8}\, 2} \leq c\, n^3$

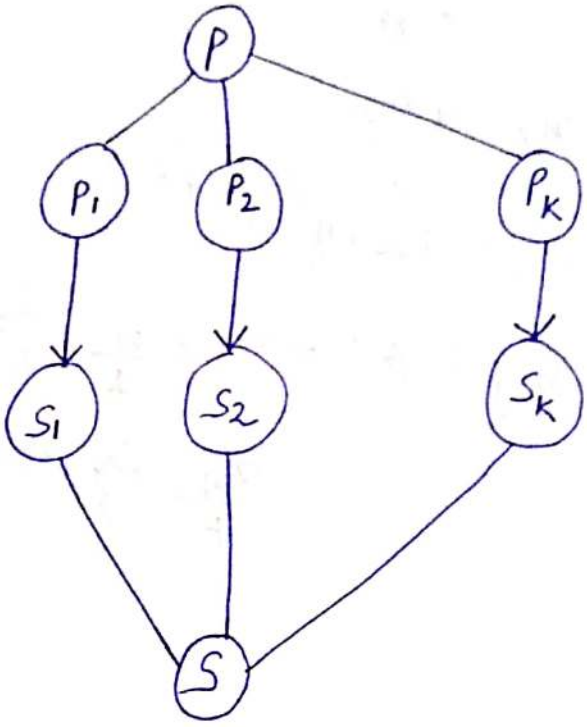
$c \to 1/2$.

$$\boxed{T(n) = \Theta(n^3)}$$

Scanned by CamScanner

22/08/2019

Divide and conquer approach (DAC)



Following are classic problems on DAC :-

1. Strassen's matrix multiplication.
2. Binary search.
3. Max and min.
4. Merge sort.
5. Quick sort.

General recurrence relation

$T(n) = aT(n/b) + O(n)$

(Master's theorem).

```
DAC (P)
{   if small(P)
        {  S(P);
        }
    else {
        divide P into P1, P2 ..... PK
        apply DAC(P1), DAC(P2) .... DAC(PK)
        combine DAC(P1), DAC(P2) ... DAC(PK)
        }
}
```

Scanned by CamScanner

$A_{n \times n}$ $B_{n \times n}$

STRASSEN MULTIPLICATION.

$O(n^3) \rightarrow$ Multiplication of two matrices.

$O(n^2) \rightarrow$ Addition of two matrices.

4X4

$$\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

$$\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

For $n \leq 2$, small problem.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$\begin{cases} c_{11} = a_{11} \cdot b_{11} + a_{12} \cdot b_{21} \\ c_{12} = a_{11} \cdot b_{12} + a_{12} \cdot b_{22} \\ c_{21} = a_{21} \cdot b_{11} + a_{22} \cdot b_{21} \\ c_{22} = a_{21} \cdot b_{12} + a_{22} \cdot b_{22} \end{cases}$$

$O(1)$ operations.

$$T(n) = \begin{cases} 8\,T(n/2) + n^2 & n > 2 \\ O(1) & n \leq 2 \end{cases}$$

using master's theorem, $\hookrightarrow$ $O(n^3)$.

$\Downarrow$ Strassen reduced to $\boxed{O(n^{2.81})}$ $\rightarrow$ need to learn formulae given by Strassen

Mid-sem syllabus: Till now + greedy + DP (option)

Scanned by CamScanner

Strassen's formulae

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

7 Products

$\boxed{\text{* wonders of Strassen}}$

$P_1 = a(f-h)$

$P_2 = (a+b)\cdot h$

$P_3 = (c+d)\cdot e$

$P_4 = d\cdot(g-e)$

$P_5 = (a+d)\cdot(e+h)$

$P_6 = (b-d)(g+h)$

$P_7 = (a-c)\cdot(e+f)$

$r = P_5 + P_4 - P_2 + P_6$

$s = P_1 + P_2$

$t = P_3 + P_4$

$u = P_5 + P_1 - P_3 - P_7$

order of matrix : power of 2

→ If not, add 0's to the matrix.

Fact: It took Strassen, 10 years of research to come up with the products.

$$T(n) = \begin{cases} 7T(n/2) + n^2 & ; n > 2 \\ \Theta(1) & ; n \le 2 \end{cases}$$

Using, Master's theorem,

$$\underline{\Theta(n^{\log_2 7}) = \Theta(n^{2.81})}.$$

Scanned by CamScanner

e.g.

$$\begin{pmatrix} 1 & 3 \\ 5 & 7 \end{pmatrix}_{n \times n} \times \begin{pmatrix} 8 & 4 \\ 6 & 2 \end{pmatrix}$$

For simplicity, $n = 2$

(i) Normal multiplication,

$$\begin{pmatrix} 8+18 & 4+6 \\ 40+42 & 20+14 \end{pmatrix} = \begin{pmatrix} 26 & 10 \\ 82 & 34 \end{pmatrix}_{n \times n}$$

(ii)

$P_1 = 1(4-2) = 2$

$P_2 = (1+3) \cdot 2 = 8$

$P_3 = (5+7) \cdot 8 = 96$

$P_4 = 7 \cdot (6-8) = -14$

$P_5 = (1+7) \cdot (10) = 80$

$P_6 = (3-7) \cdot (6+2) = -32$

$P_7 = (1-5)(8+4) = -48$

Now,

$r = 80 + (-14) - 8 + (-32)$

$r = 26$ ✓

$s = 2 + 8 = 10$ ✓

$t = 96 - 14 = 82$ ✓

$u = 80 + 2 - 96 - (-48)$

$= 34$ ✓

* Strassen matrix multiplication verified !!

→ Lab work: Strassen matrix multiplication

☆ { Class test on Monday }

Scanned by CamScanner

29/08/2019

Maximum and minimum in an array. D&C problem

```
maxmin ( a, n, max, min)
{       max = min = a[i]
        for (i=2 to n) do
        {   if (a[i] > max ) then max = a[i]
            if (a[i] < min) then min = a[i]
        }
}
```

```
maxmin (i, j, max, min)
{   if (i==j) then max = min = a[i]   // For one element
    else if (i==j-1) then
    {   if (a[i] < a[j]) then
            { max = a[j]
              min = a[i]
            }
        else
            { max = a[i]
              min = a[j]
            }
    }
    else
    {   mid = (i+j)/2
        maxmin (i, mid, max, min)
        maxmin (mid+1, j, max1, min1)
        if (max < max1) then
                max = max1
        if (min > min1) then
                min = min1
    }
}
```

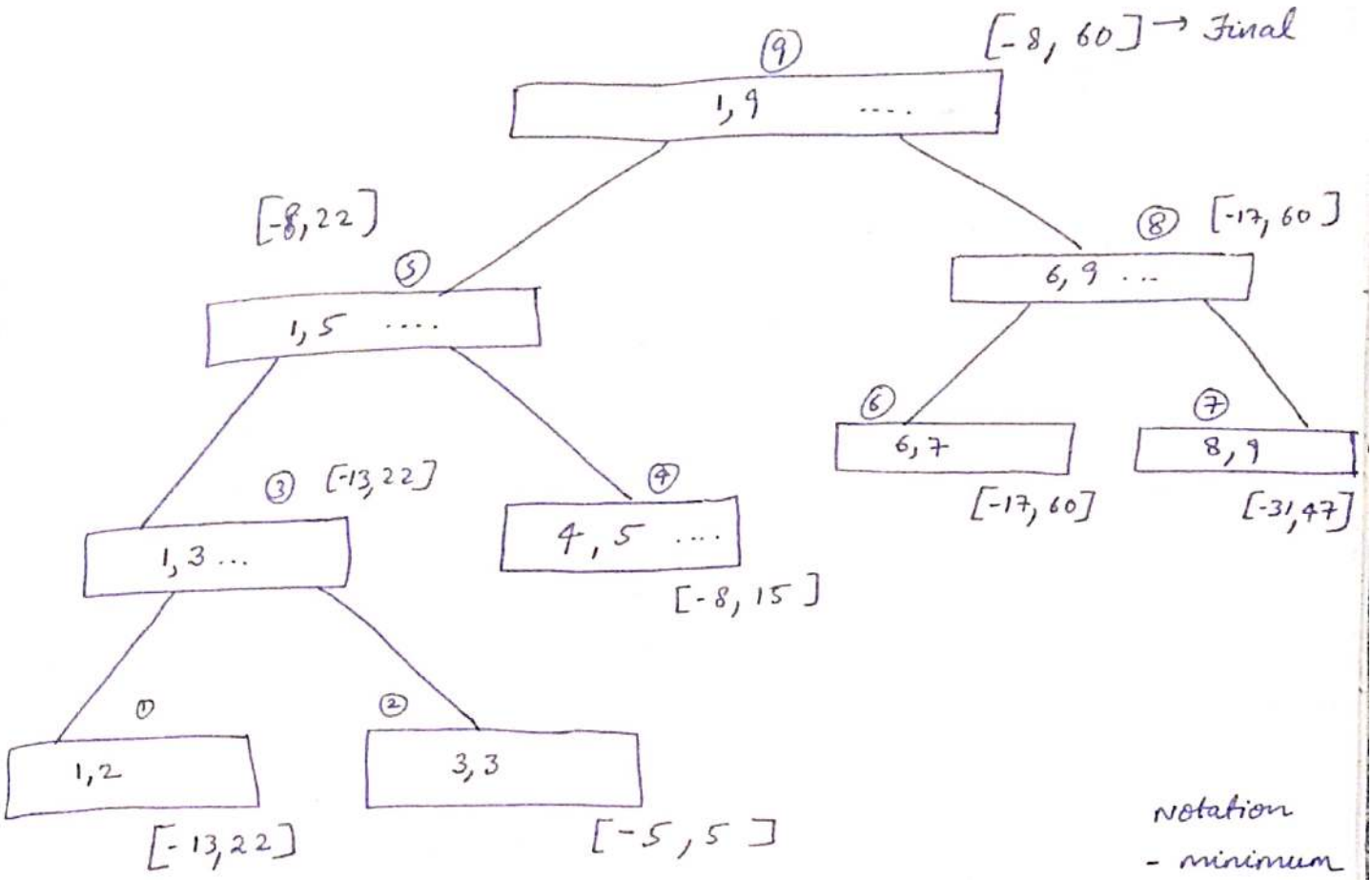
// For two elements

not required to compute

Scanned by CamScanner

e.g.

1	2	3	4	5	6	7	8	9
22	13	5	8	15	60	17	31	47



Notation
- minimum

$$T(n) = \begin{cases} 2T(n/2) + O(1) & n > 2 \\ 1 & n = 2 \\ 0 & n = 1 \end{cases}$$

⇒ Time complexity is reduced.

However, space complexity increases.

→ Dawn of mid-semester
Sept. 16th to Sept 20th

Next class: Greedy approach.

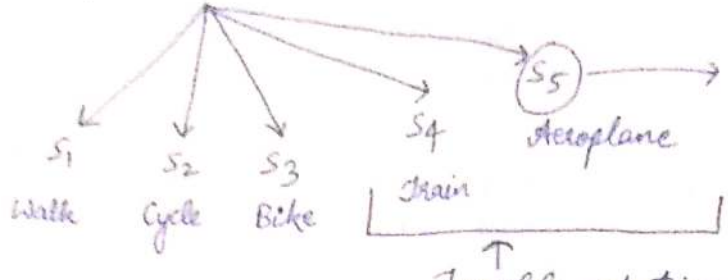
Scanned by CamScanner

03/09/2019

Greedy approach / method.

250 Km. → distance

time constraint ← P : A $\xrightarrow{(12hr-2hr)}$ B



Feasible solutions.

Strategies :
(i) Greedy.
(ii) Dynamic programming
(iii) Branch and bound.

N-Queen problem ← 1

1. Dijkstra
2. Knapsack problem (Greedy)
3. Activity selection.
4. Task / Job scheduling
5. Huffman coding
6. Prim/Kruskal
7. Min Spanning tree

1. → Matrix-Chain-Multiplication
2. → LCS
3. → 0/1 Knapsack

Knapsack. (Assume it as a bag) with capacity = 15 kg

objects	1	2	3	4	5	6	7
profits	10	5	15	7	6	18	3
weights	2	3	5	7	1	4	1

objective : maximize profit.
optimization problem

Fractional knapsack : Fraction of the object can be taken.
Profit and Weight will also be fractional in this case.

maximize profit with minimum weight

$\frac{P_i}{W_i}$:→

5	1.6	3	1	6	4.5	3
1	$\frac{2}{3}$	1		1	1	1

Scanned by CamScanner

$= 2\times1 + 5\times\frac{2}{5} + 5\times1 + (\cancel{6}\,1)\times1 + 1\times4 + 1\times1$

$= 2 + 2 + 5 + 1 + 4 + 1$

$= \underline{15\ kg}$.

Profit

$= 10 + \frac{2}{3}\cdot5 + 15 + 6 + 18 + 3$

$= \underline{55.3}$.

Job/Task sequencing/scheduling with deadlines.

$n = 5$

		✓				
Jobs	J_1	J_2	J_3	J_4	J_5	
			X			
Profits	20	15	10	5	1	decreasing order of profits
Deadlines	2	2	1	3	3	

- All jobs are unit time jobs.

$0 - \overset{J_2}{1} - \overset{J_1}{2} - \overset{J_4}{3}$

$J_2\ J_1\ J_4$ or

$J_1\ J_2\ J_4$

→ profit $= 20 + 15 + 5$
$= 40$

→ penalty $= 10 + 1$
$= 11$

Next class : → Activity selection
+
→ Huffman coding

⇒ Mid-semester confirmed date : $\underline{20^{th}\ Sept.\ 2019}$

Scanned by CamScanner

05/09/2019

Huffman codes

Message - B C C A B B D D A E C C B B A E D D C C

FIXED LENGTH

000	A	→	3
001	B	→	5
010	C	→	6
011	D	→	4
100	E	→	2
			20 characters.

20 Bytes memory requirement i.e. 160 bits.

Memory : ↓ ENCODED.

$(8\times5 + 3\times5) + (3\times20)$

$55 + 60 = 115$ bits.

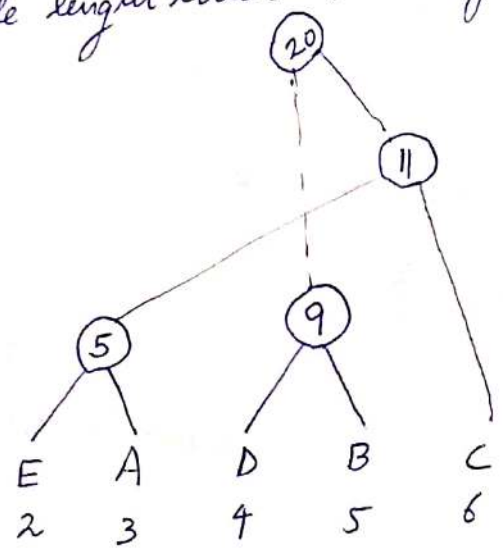
Lack of rule

→ Fixed length code : (n bits) → 2^n codes

→ Variable length code : (code length of character) $\propto \frac{1}{\text{(frequency of occurrence)}}$

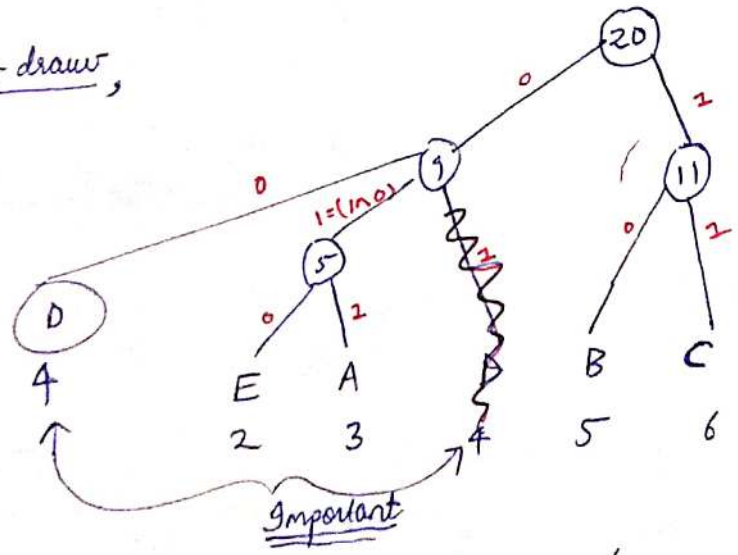
CONSTRUCTION of HUFFMAN TREE

e.g.



smaller : LHS
larger : RHS

Re-draw,



CODES

D : 00
E : 010
A : 011
B : 10
C : 11

$(9+10+12+8+6) + (8\times5 + (9+8+9+10+12))$

↓ 45 bits

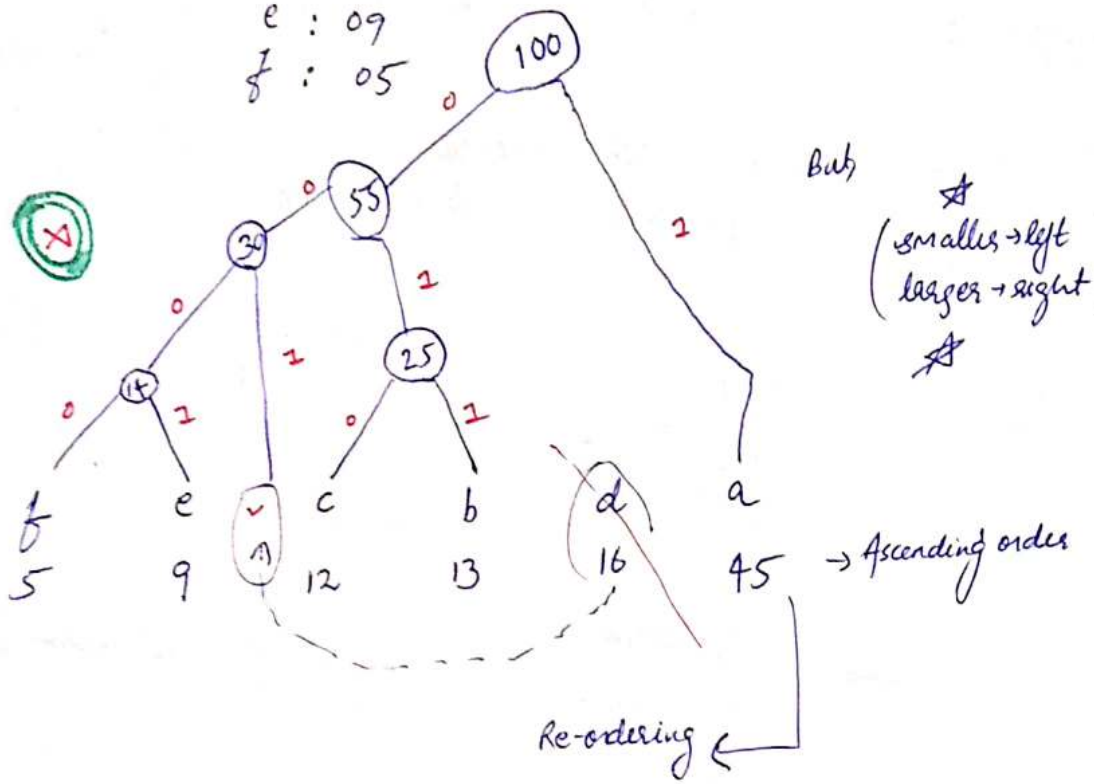
for information mapping

Scanned by CamScanner

Q1) Construct Huffman tree for the following character set

a : 45
b : 13
c : 12
d : 16
e : 09
f : 05

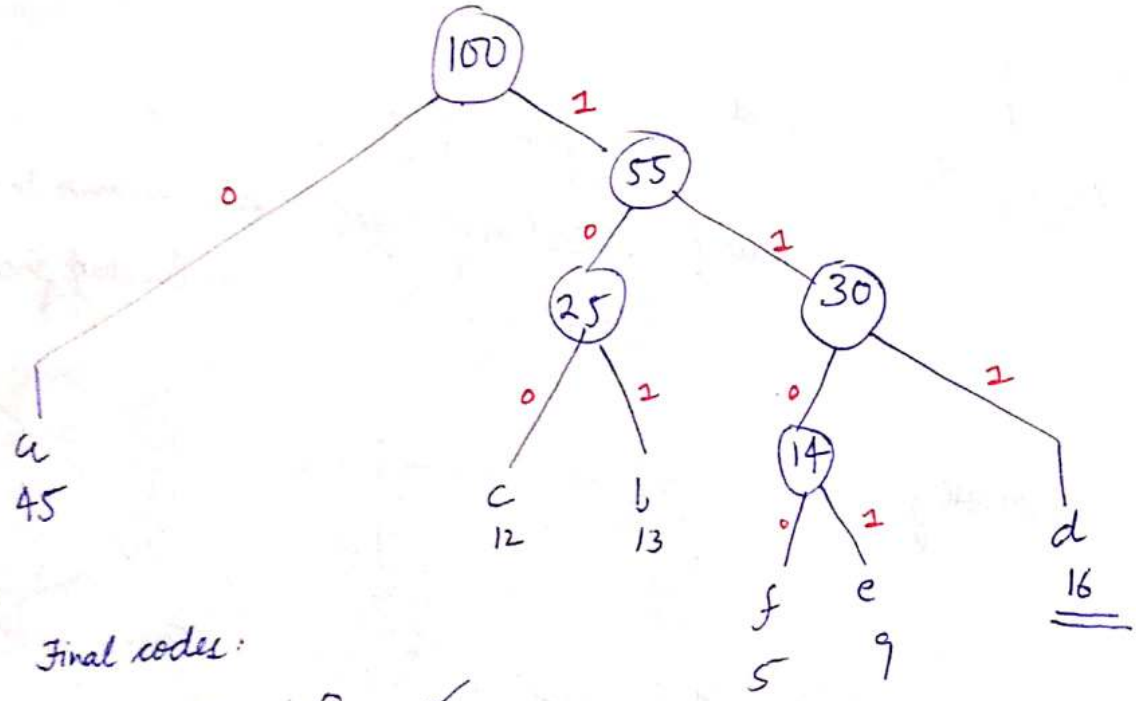
Sol



But

☆ (smaller → left, larger → right) ☆

→ Ascending order



Final codes:

a : 0 ✓
b : 101 ✓
c : 100 ✓
d : 111 ✓
e : 1101 ✓
f : 1100 ✓

* easier way :
→ after merging the two minimum, place it at its correct position.

Scanned by CamScanner

03/10/2019

→ Dynamic Programming.

→ Maximization / Minimization problems
→ Counting problems.

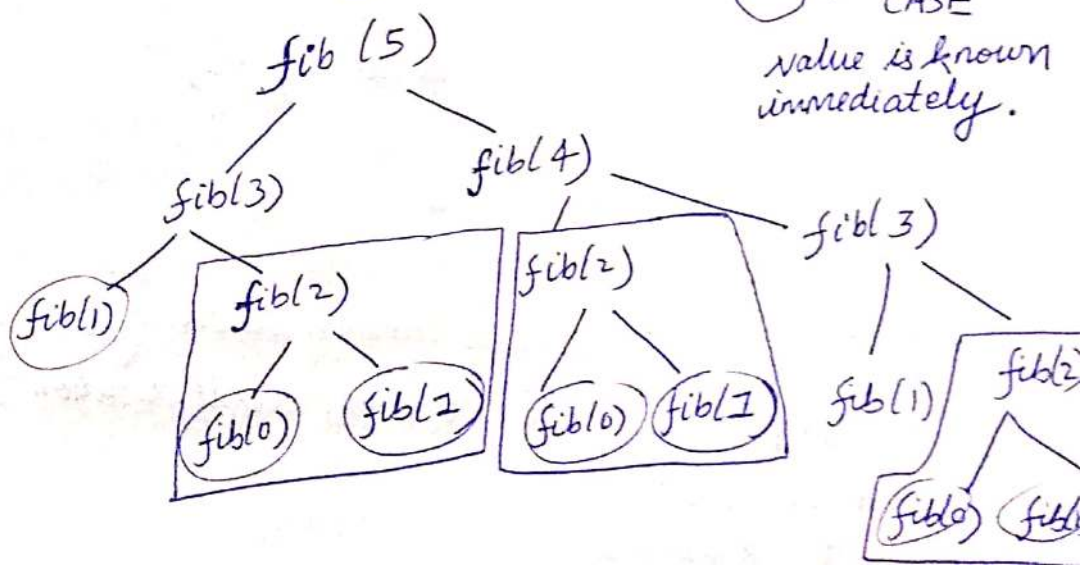
Methods:
A) Recursive (using memoization)
B) Iterative (using tabulation).

Example: Fibonacci series:

$$fib(n) = \begin{cases} 0 & ; \text{if } n=0 \\ 1 & ; \text{if } n=1 \\ fib(n-2)+fib(n-1) & ; \text{if } n>1 \end{cases}$$

```
int fib(int n)
{   if (n≤1)
        return n;
    return (fib(n-2) + fib(n-1));
}
```

Recursion tree:



○ → BASE CASE
value is known immediately.

Repetitive work:

Scanned by CamScanner

DP solution :

$O(n)$

0	1	1	2	3	5
0	1	2	3	4	5

$T(n) = n+1$

Time complexity analysis of recursive Solution,

$$T(n) = 2T(n-1) + 1$$

$$O(2^n).$$

→ Memoization is Top-Down approach.
→ Tabulation is bottom-up approach.

Tabulation solution : Array of size $(n+1)$.

```
int fib(int n)
{       f[0]=0;
        f[1]=1

        for (int i=2; i<=n; i++)
            { f[i]=f[i-2]+f[i-1];
            }
        return f[n];
}
```

(i) L C S — Longest common subsequence
(ii) M.C.M — Matrix chain multiplication
(iii) Optimal B.S.T
(iv) 0-1 Knapsack

Scanned by CamScanner

M.C.M.

(Matrix Chain Multiplication)

$$\boxed{\underset{m\times n}{[A_1]}\ \underset{n\times p}{[A_2]}}\ \underset{p\times l}{[A_3]}$$

→ $m\times n\times p$ operations required.

$$A_1\times A_2\times A_3$$

$(A_1A_2)A_3$ OR $A_1(A_2A_3)$

For e.g.

A_1: $\underset{d_0}{2}\times\underset{d_1}{3}$ $\quad A_2$: $\underset{d_1}{3}\times\underset{d_2}{4}$ $\quad A_3$: $\underset{d_2}{4}\times\underset{d_3}{2}$

$(A_1\times A_2)\times A_3$

→ $(2\times3\times4)=24$

$2\times4\times2=16$

$\boxed{16+24=40}$

$A_1\times(A_2\times A_3)$

$3\times4\times2=24$

$2\times3\times2=12$

$\boxed{24+12=36}$ ✓ better

Total No. of operations :

Formula: (Generalization)

$$C[\overset{i}{1},\overset{k}{2}]+C[\overset{k+1}{3},\overset{j}{3}]+(\underset{i-1}{d_0}\times\underset{k}{d_2}\times\underset{j}{d_3})$$

$$C[\overset{i}{1},\overset{k}{2}]+C[\overset{k+1}{2},\overset{j}{3}]+(\underset{i-1}{d_0}\times\underset{k}{d_1}\times\underset{j}{d_3})$$

$$\boxed{C[i,j]=\overset{\text{(minimum)}}{M}\left[C[i,k]+C[k+1,j]+(d_{i-1}\times d_k\times d_j)\right]}$$

where, $i\le k<j$ LOOP.

Scanned by CamScanner

For e.g.

$$[A_1] \times [A_2] \times [A_3] \times [A_4]$$

$$\underset{d_0}{3} \times \underset{d_1}{2} \quad \underset{d_2}{2 \times 4} \quad \underset{d_3}{4 \times 2} \quad \underset{d_4}{2 \times 5}$$

$$C[1,4] = \underset{k=1 \text{ to } 3}{\text{MINIMUM}} \begin{cases} k=1; & C[1,1] + C[2,4] + (d_0 \times d_1 \times d_4) \\ k=2; & C[1,2] + C[3,4] + (d_0 \times d_2 \times d_4) \\ k=3; & C[1,3] + C[4,4] + (d_0 \times d_3 \times d_4) \end{cases}$$

$k \rightarrow$ split in matrix

Table:

COST TABLE

	1	2	3	4
1	0	0	0	0
2	24	16	40	
3	28	36		
4				

$i \rightarrow$

$\rightarrow$ diagonal traversal of the matrix

	1	2	3
1	1	2	3
2	1	2	3
3	3	3	
4			

$j \nearrow$ $\quad i \searrow$

$\rightarrow$ value of k for which minimum was achieved.

$$\rightarrow \quad C[1,2] = \underset{k=1}{\text{Min}} \left[\overset{0}{C[1,1]} + C[2,2] + (d_0 \times d_1 \times d_2) \right]$$

$3 \times 2 \times 4$

$= 24$

Scanned by CamScanner

Paranthesization: $((A_1)(A_2\ A_3))\ A_4$ → order of multiplication.

objective : Order of multiplication of matrices.

10/10/2019

.2

Q1 Find order of paranthesization for the given matrix:

common

k

Inc. order of matrices

		(MATRIX number)
A_1	30×35	← 0
A_2	35×15	← 1
A_3	15×5	← 2
A_4	5×10	← 3
A_5	10×20	← 4
A_6	20×25	← 5

growth

No. of MATRICES

	0	1	2	3	4	5	
0	0	15750	7875 (0), k=0 ~~18000~~ ~~78750~~	~~24,625~~ 21,000 (1) 9375 (2)	11875 (k=2)		6
1		0	2,625	~~41,375~~ ~~26,750~~ (2)	7125 ~~K=2~~	10,500 (k=2)	5
2			0	750	~~3750~~ ~~5000~~ 2500 (2)	5375 (k=2)	3+1
3				0	1,000	3,500 ~~20,000~~ (K=4)	3
4					0	5,000	2
5						0	1

(0,2)

$i = 0$;
$j = 2$;
$k = 1$;

$(i \to k) + (k+1, j)$

$i \times k \times j$
first, low, last

extra information to be added

ONLY ONE WAY

$(0,1) \to (30 \times 35 \times 15)$ + $30 \times 15 \times 5$ $k=1$

+ $30 \times 35 \times 5$ $k=0$ $\boxed{0,2}$ REDO

$(1,2) \to (35 \times 15 \times 5)$

$(2,3) \to 15 \times 5 \times 10$

$(3,4) \to 5 \times 10 \times 20$

$(4,5) \to 10 \times 20 \times 25$

→ $(A\ B)\ C$
$A\ (BC)$
$(AC)\ B$ ✗

Scanned by CamScanner

OPTIMAL 2-SIZED MATRIX

1,3

$k=1$ $\quad C[1,1] + C[2,3] + (35 \times 15 \times 10) = 6000$

$k=2$ $\quad C[1,2] + C[3,3] + (35 \times 5 \times 10) = 4,375$

2,4

$k=2$ $\quad C[2,2] + C[3,4] + (15 \times 5 \times 20) =$

$k=3$ $\quad C[2,3] + C[4,4] + (15 \times 10 \times 20) =$

$k=4$? PROBLEM itself

3,5

$k=3$ $\quad C[3,3] + C[4,5] + (5 \times 10 \times 25)$

$k=4$ $\quad C[3,4] + C[5,5] + (5 \times 20 \times 25)$

(0,3)

$k=0 \Rightarrow C[0,0] + C[1,3] + (30 \times 35 \times 10) = 14,875$

$k=1 \Rightarrow C[0,1] + C[2,3] + (30 \times 15 \times 10) = 21,000$

$k=2 \Rightarrow C[0,2] + C[3,3] + (30 \times 5 \times 10) = 9,375$

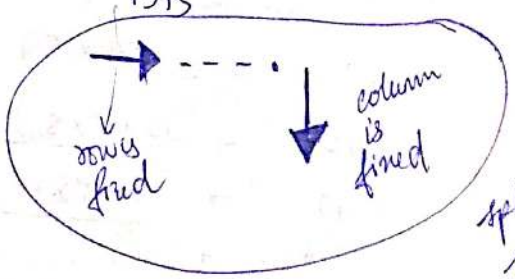
1, 4

combining

$k=1;\ C[1,1] + C[2,4] + (35 \times 15 \times 20) = 13,000$

$k=2;\ C[1,2] + C[3,4] + (35 \times 5 \times 20) = 7,125$

$k=3;\ C[1,3] + C[4,4] + (35 \times 10 \times 20) = 11,375$



splitting pattern

Scanned by CamScanner

$\boxed{2,5}$

$k=2$; $3500 +$ $(15 \times 5 \times 25) = 5,375$

$k=3$; $750 + 5000 +$ $(15 \times 10 \times 25) = 9,500$

$k=4$; $2500 +$ $(15 \times 20 \times 25) = 10,000$

end of 4 matrix result

$\boxed{0,4}$

5 matrix multiplication

$k=0$; $7125 + 0$ $+ (30 \times 35 \times 20) = 26,125$

$k=1$; $15750 + 2500$ $+ (30 \times 15 \times 20) = 27,250$

$k=2$; $7875 + 1,000$ $+ (30 \times 5 \times 20) = 11,875$

$k=3$; 9375 $+ (30 \times 10 \times 20) = 15,375$

COMBINATION effort or effort to combine the split halves

$\boxed{1,5}$

$k=1$; ~~7125~~ $5375 + (35 \times 15 \times 25) = 18,500$

$k=2$; ~~15750 + 2500~~ $2625 + 3500$ $+ (35 \times 5 \times 25) = 10,500$

$k=3$; $4375 + 5000$ $+ (35 \times 10 \times 25) = 18,125$

$k=4$; $0 + 7125$ $+ (35 \times 20 \times 25) = 24,625$

RESULT 9

$\boxed{0,5}$

		1	5	
(1,5)	$k=0$;	$10,500 + 0$	$+ (30 \times 35 \times 25)$	$= 36,750$
(2,4)	$k=1$;	$15,750 + 5375$	$+ (30 \times 15 \times 25)$	$= 32,375$
(3,3)	$k=2$;	$7875 + 3500$	$+ (30 \times 5 \times 25)$	$= 15,125$
(4,2)	$k=3$;	$9375 + 5000$	$+ (30 \times 10 \times 25)$	$= 21,875$
(5,1)	$k=4$;	$11875 + 0$	$+ (30 \times 20 \times 25)$	$= 26,875$

MIN

Matrix split

→ Paranthesization :

$$((A_1)(A_2 \quad A_3)) \mid ((A_4 \quad A_5)(A_6))$$

0 1 2 3 4 5

Time Complexity : $O(N^3)$

Scanned by CamScanner

21/10/2019

.2

SYLLABUS

Refer to IT-2 for class work

DYNAMIC PROGRAMMING :

(i) Matrix Chain Multiplication
(ii) 0-1 knapsack
(iii) LCS
(iv) Optimal Binary Search Tree.
(v) Floyd Warshall.

BRANCH & BOUND

(i) Subset sum
(ii) N-Queen
(iii) -

STRING MATCHING ALGORITHMS

- Rabin
-
-

INTRO

- NP complete

Scanned by CamScanner

L. C. S.

(Longest · Common · Subsequence)

What?
length of longest common subsequence

string 1 = "a b c d e f g h i j"

string 2 = "c d g i"

Let, str1 = "stone"
str2 = "longest"

How?

		" "	l	o	n	g	e	s	t
		0	1	2	3	4	5	6	7
" "	0	0	0	0	0	0	0	0	0
s	1	0	0	0	0	0	0	1	1
t	2	0	0	0	0	0	0	1	2
o	3	0	0	1	1	1	1	1	2
n	4	0	0	~~0~~ 1	1+1 = 2	2	2	2	2
e	5	0	0	1	2	2	3	3	3

s(1)
s(2)
s(1) == (2

longest subsequence : one

How to backtrack ? (Inverse logic of coming down match characters and backtrack)

Time complexity: O(m×n)

m×n

Scanned by CamScanner

example:

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

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

Y \ X		0	1 A	2 B	3 C	4 B	5 D	6 A	7 B
	0	0	0	0	0	0	0	0	0
B	2	0	0	1	1	1	1	1	1
D	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
B	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	2	3	4	4

→ length of the longest common subsequence

single call as 'A' = 'A'

two calls as 'A' != 'B'

what → Backtracking strategy.

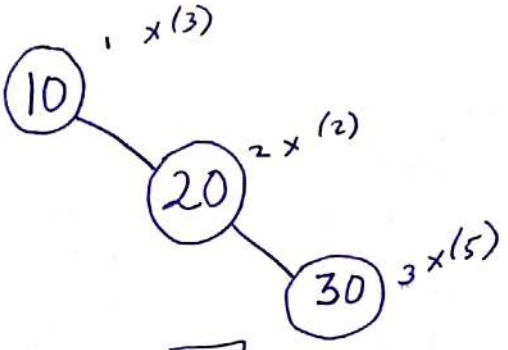
31/10/2019.

·2

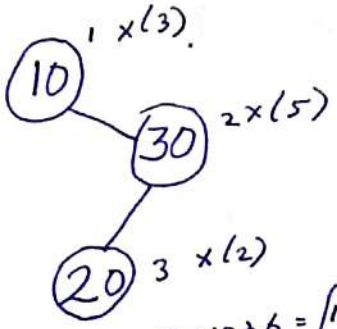
BINARY · SEARCH · TREE

key : [10, 20, 30]

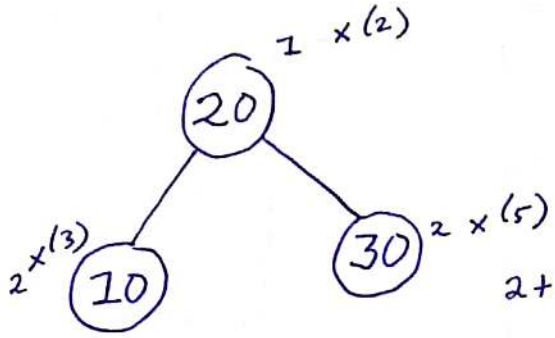
What ? Number of BST's that can be formed using key (s).



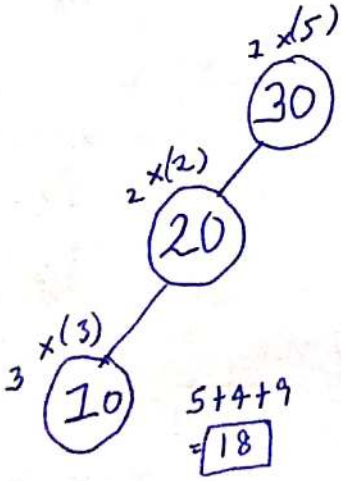
3+4+15 = 22



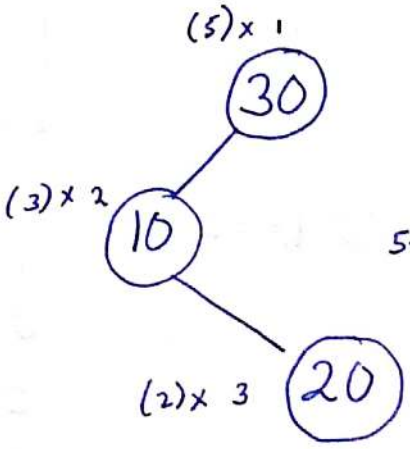
3+10+6 = 19



2+6+10 = 18



5+4+9 = 18



MINIMUM

5+6+6 = 17

No. of BST's = $\frac{2nC_n}{(n+1)!}$ ← CATALAN SERIES

Scanned by CamScanner

Q. key : [10 20 30]

frequency : [3 2 5]

Problem

	1	2	3	4
key	10	20	30	40
frequency	4	2	6	3

i \ j	0	1	2	3	4
0	0	4	min(8,10) 8	23	26
1		0	2	10	16
2			0	6	12
3				0	3
4					0

$(j-i) == 1$

C [0 1]

C [1, 2]

C [2, 3]

C [3, 4]

$(j-i) == 2$

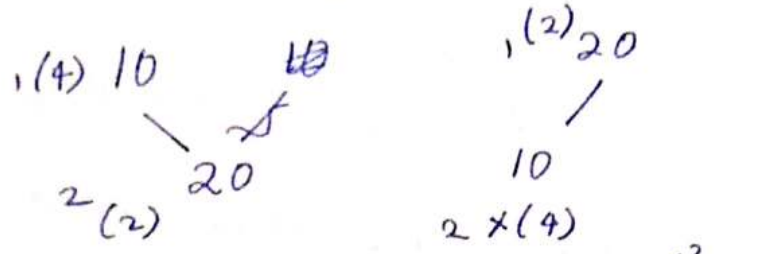
C [0,2] =

C [1,3] =

C [3,4] =

Scanned by CamScanner

→ $c[0,2]$:- Key 10 and 20



$4+4=8$ $\qquad$ $2+4^{\times 2}=10$

CASE $(j-i) == 3$ (5 combinations)

$c[0,3] =$

$c[1,4] =$

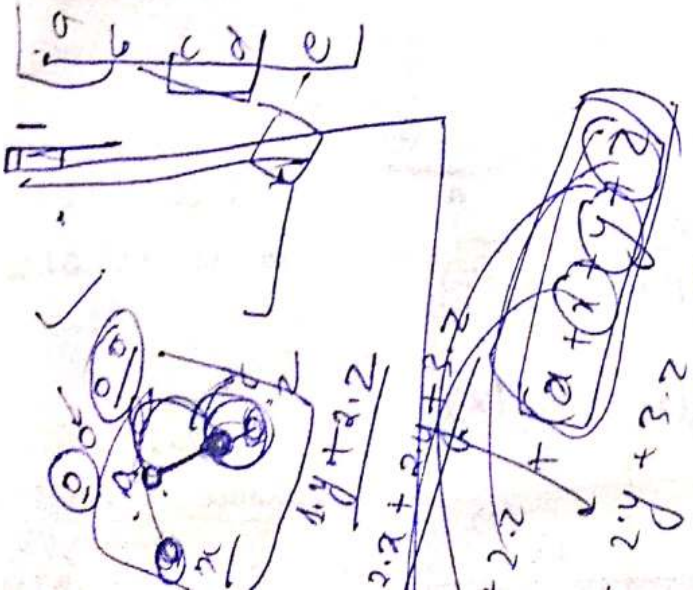
$(j-i) == 4$

$c[0,4] =$

Formula :

$$c[i,j] = \begin{cases} f_i & ; \text{ if } \boxed{j=i-1} \\ \min\limits_{i \le r \le j} \left[c[i,r-1] + c[r+1,j] + w[i,j] \right] & \text{if } i \le j \end{cases}$$

Calculate



Scanned by CamScanner