31/7/19

Date
Page No.

m & n find G.C.D

1. Factorize m
   $m_1 \times m_2 \times \ldots \times m_j$
2. Factorize n
   $n_1 \times n_2 \times \ldots \times n_j$
3. Identify common factors

$m = 36$
$n = 48$

factor m $= \underline{2 \times 2 \times 3 \times 3}$
factor n $= \underline{2 \times 2 \times 2 \times 2 \times 3}$

Comon $= 2 \times 2 \times 3$
$= 12$

9 divisions

Euclid (m, n)
{
while m donot divide n
  r = n mod m
  n = m
  m = r
end
return m
}

(1)

1/8/19

Lab

1. GCD → (a) factors → (b) Euclids
2. Linear Search

Date
Page No.

```
int gcd (int a, int b)
{
  if (a==0)
     return b;
  if (b==0)
     return a;
  if (a>b)
     return gcd (a-b, b);
  else
     return gcd (a, b-a);
```

5/8/19 Insertion Sort

Insertion Sort (A)

```
1. for j = 2 to A.length
2.   key = A[j]
3.   i = j-1
4. while i>0 and A[i] > key
5.   A[i+1] = A[i]
6.   i = i-1      // end
7.   A[i+1] = key.
```

1	2	3	4	5	6	7	8	9	10
2	7	9	4	1	0	5	6	8	3

j = 4

key = A[4] = 4

i = 4-1 = 3

while 3>0 and 9 > 4.

A[4] = A[3]

A[4] = 9

2	7	9	9	1	0	5	6	8	3

~~while~~ $i = 3 - 1 = 2$

$2 > 0$ and $7 > 4$

$A[3] = 7$

2	7	7	9	1	0	5	6	8	3

$i = 2 - 1 = 1$

$1 > 0$ and $2 > 0$

$A[2] = 4$

2	4	7	9	1	0	5	6	8	3

Worst Case

5	4	3	2	1

$1 \to 1 + 1 = 2$

$2 \to 2 + 2 = 4$

$3 \to 3 + 3 = 6$

$O(1 + 2 + 3 \ldots + n)$

$\cancel{2}\left(\frac{n(n+1)}{\cancel{2}}\right)$

$= n^2 + n$

$= O(n^2)$

average case complexity = worst case complexity.

~~Cast~~ Best Case

$T_n = C_1 n + C_2(n-1) + C_3(n-1) + C_4(n-1) + 0 + 0$
$+ C_7(n-1)$

Date
Page No.

$(C_1 + C_2 + C_3 + C_4 + C_4)n - (C_2 + C_3 + C_4 + C_4)$

$an - b$

$\Omega(n)$

5/8/19

$\boxed{\text{Int}}$

Algorithm

iterative Recursive

① A( )
{
int i;
for (i=1 to n) 1
{ printf ("Hi") n
} n-1
}

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

Date
Page No.

(2) A ( )

{

int i ;　　　　1

for (i = 1 to n)　　　　n

for (j = 1 to n)　　　　n^2

print ("statement")　　　　n^2

$n^2 + n^2 + n + 1$

$2n^2 + n + 1$

$= O(n^2)$

(3) A( )

{

i = 1, S = 1　　　　1

while (S <= n)　　　　n

{

i++　　　　~~n + 1~~

S = S + i　　　　~~n - 1~~

Print ("statement")　　　　~~n - 1~~

}

}

$O(\sqrt{n})$

(4) A( )

{

i = 1

for (i = 1 ; i^2 < n ; i++)　　　　$O(\sqrt{n})$

pf ("　　")

}

Charlie

Date

Page No.

(5) A()

{

int i, j, k, n;

for (i=1; i<=n; i++)

for (j=1; j<=i; j++)

for (k=1, k<=100; k++)

{

printf("Statement")

}

}

$O(n^2)$

i=1	i=2	i=3	i=n
j=1	j=2	j=3	j=n
k=100	k=200	k=300	k=n×100

$100 + 200 + \ldots\ n \times 100$

$100(1+2+3+\ldots n)$

$100\left(\frac{n(n+1)}{2}\right)$

$50(n^2+n)$

$O(n^2)$

(6)

f(i=1; i<=n; i++)

for(j=1, j<=i^2; j++)

for (k=1, k<=n/2, k++)

printf(" ")

}

i=1	i=2	i=n
$j=1^2$	j=4	$j=n^2$
$k=1^2 n/2$	$k=4(\frac{n}{2})$	$k=n^2(\frac{n}{2})$

Date
Page No.

$$\frac{n}{2}\left(1^2+2^2+3^2+\ldots\ldots n^2\right)$$

$$\frac{n}{2}\left(\cancel{n^2}\;\cancel{n^2(n^2+1)}\right)$$

$$\frac{n}{2}\left[\frac{n\cdot(n+1)(2n+1)}{6}\right]$$

$$O(n^4)$$

```
fn()
for (i=1; i<=n ; i= i*2)
{
  print ("statement")
}
```

1 2 3 4 .... k

1 2 2^2 2^3 2^{k-1}

$2^{k-1} = n$

$k-1 = \log_2 n$

$k = \log n$

i = 1 i =

6/8/19

Worst case - Insertion Sort

$$T(n) = C_1 n + C_2(n-1) + C_3(n-1) + C_4 \cdot \sum_{t=2}^{n} t_i + C_5 \sum_{t=2}^{n} t_i - 1 + C_6 + \sum_{t=2}^{n} t_i - 1 + C_7 (n-1)$$

$$= an^2 + bn + c$$

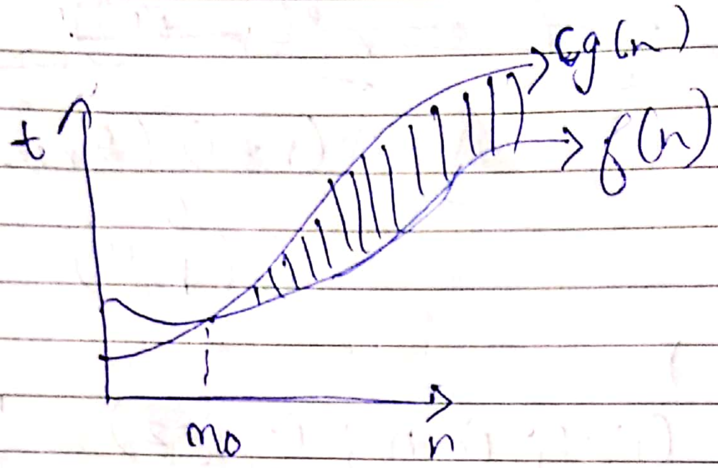
Complexity $= O(n^2)$

Scanned by CamScanner

Charlie
Date
Page No.

Asymptotic Notation

- Big O



$f(n) = an+b$

$f(n) \leq C\,g(n)$ $\qquad 3n+2 \leq 2(n)$

after some $n \geq n_0$
$C > 0, \quad n_0 \geq 1$

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

$f(n) = 3n+2$

$f(n) \leq c\,g(n)$
$3n+2 \leq 4n$

$c = 4, \quad n_0 = 2$

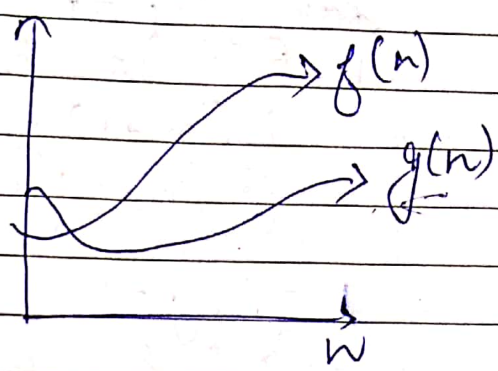
$f(n) = O(gn)$
$= O(n)$

Best Case

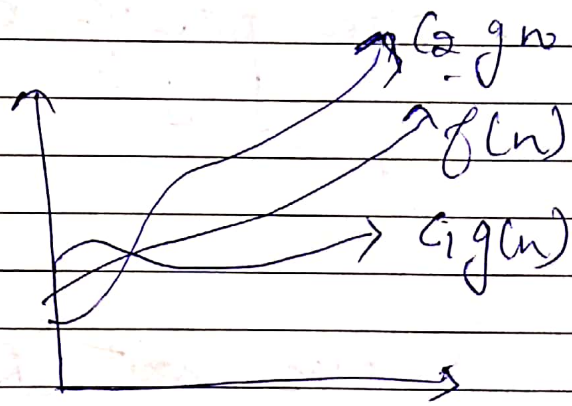
$f(n) = 3n+2$

$f(n) \geq c\,g(n)$
$f(n) = \Omega\,g(n)$
$3n+2 \geq c\,g(n)$
$3n+2 \geq c(n_0)$
$3n+2 \geq 1\,(1)$

$c = 1,\ n = 1$



Average Case
$c_2 g(n) \geq f(n) \geq c_1 g(n)$
$n \leq 3n+2 \leq 4n$



7/8/19

Space Complexity

O(1) → Insertion sort

Merge Sort (Outplace Algorithm)
Merge (A, p, q, r)
{
n1 = q - p + 1;
n2 = r - q;
let L[1 ... n+1] and R[1 to n2+1] be new arrays
for (i = 1 to n1)
L[i] = A[p + i - 1]

```
for (j=1 to n2)
    R[j] = A[q+j]
    L[n1+1] = ∞
    R[n2+1] = ∞

i = 1, j = 1
for ( k = p to r)
if ( L[i] ≤ R[j])
    A[k] = L[i];
    i = i+1;
else A[k] = R[j]
    j = j+1
}.
```

```
merge_sort (A, p, r)
{
  if p<r
  q = ⌊(p+r)/2⌋ -> floor value
  merge_sort (A, p, q)
  merge_sort (A, q+1, r),
  merge (A, p, q, r)
}
```

Space Complexity nlogn

Charlie
Date
Page No.

8/8/19

Lab - Me-2

(1) Insertion Sort

Execution time => include<time.h>
clock_t begin, end;
begin = clock();
:
:
end = clock();

print (begin - end)

14/8/19

Insertion Sort → n Best
n^2 Average
n^2 worst.

Merge sort → $n \log n$
$n \log n$
$n \log n$

Quick sort → $n \log n$
$n \log n$
n^2

Date
Page No.

Quick Sort :→

9	6	5	0	8	2	4	7

Partition (A, P, r)
{
x = A[r]
i = p-1
for(j = p to r-1)
{ if (A[j] ≤ x)
{
i = i + 1
exchange A[i] with A[j]
}
}
exchange A[i+1] with A[r]
return i+1
}

	9	6	5	0	8	2	4	7
1	6	9	5	0	8	2	4	7 pivot
2	6	5	9	0	8	2	4	7
3	6	5	0	9	8	2	4	7
4	6	5	0	2	8	9	4	7
5	6	5	0	2	4	9	8	7
6	6	5	0	2	4	7	8	9

Quicksort (A, p, r)
{
if (p < r)
{
q = Partition (A, p, r);
Quicksort (A, p, q - 1);
Quicksort (A, q + 1, r);
}

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

Q.S (1, 6)

P(1, 6) QS(1, 3) QS(5, 6)

P(1, 3) Q(1, 1) Q(3, 3) P(5, 6) Q(5, 5)

= 2 = 6

19/8/19

Quick Sort v/s merge sort.

20/8/19

Heap Sort

Height	1	2	3	4
Nodes	3	7	15	31

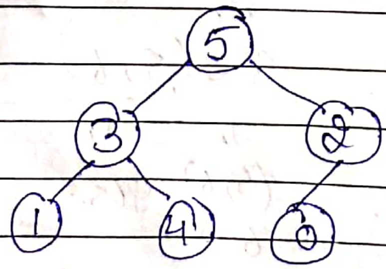
nodes $n = (2^{h+1} - 1)$

$h = \lfloor \log n \rfloor$

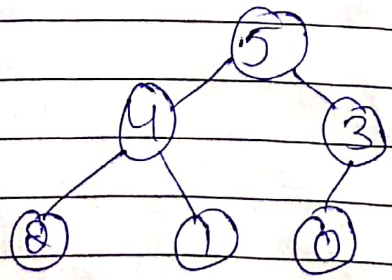
5	3	2	1	4	0

~~left child = 2i~~

right child = $2i+1$

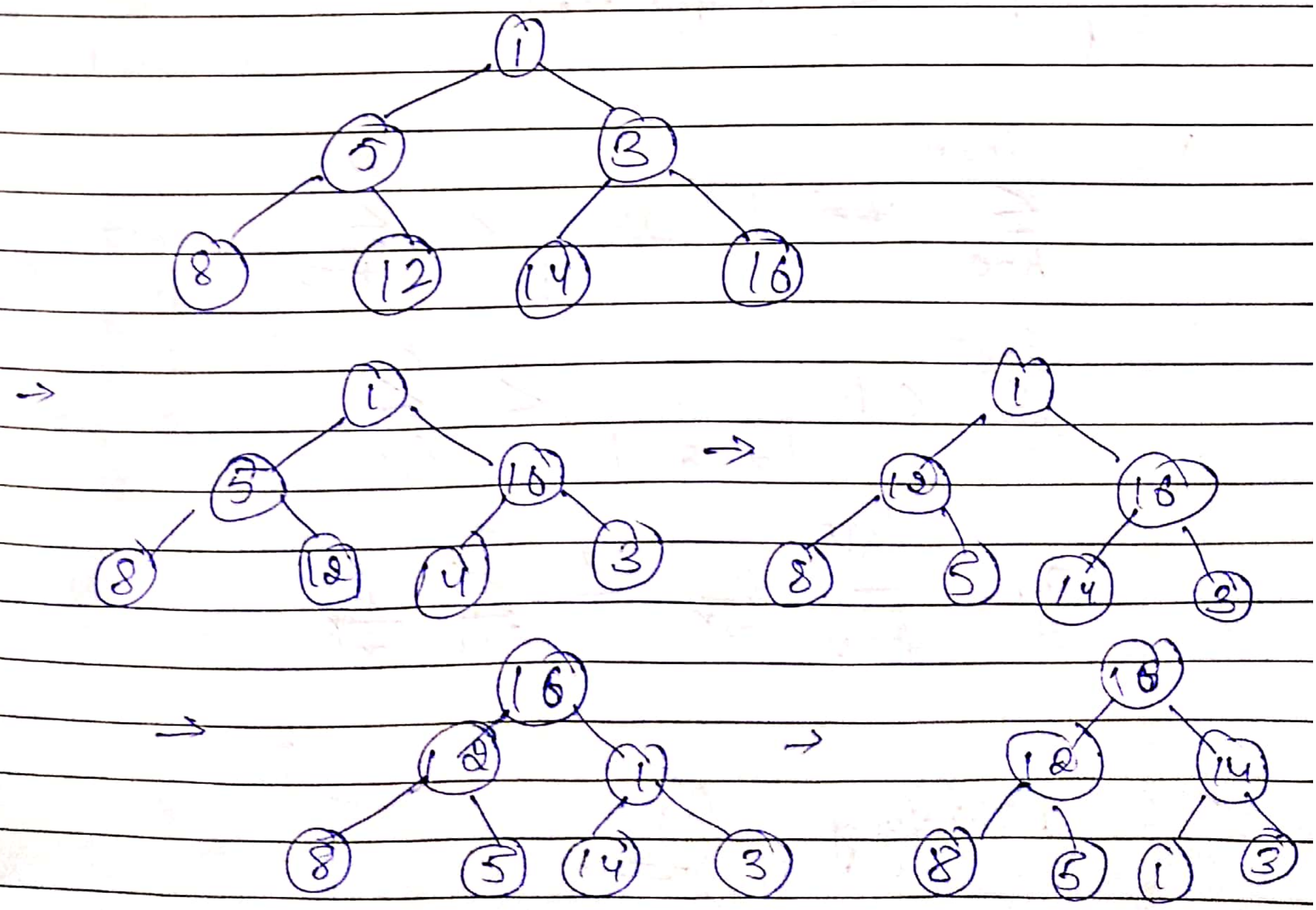


5	4	3	2	1	0



```
max_heapify (A, i)
{
l = 2i;
r = 2i+1;
if (l ≤ A.heapsize and A[l] > A[i])
    largest = l;
else largest = l;
if (r ≤ A.heapsize and A[r] > A[largest])
    largest = r;
if (largest ≠ i)
exchange A[i] with A[largest]
max_heapify (A, largest);
}
```

no. of leaf nodes = $\left(\frac{n}{2}\right) + 1$



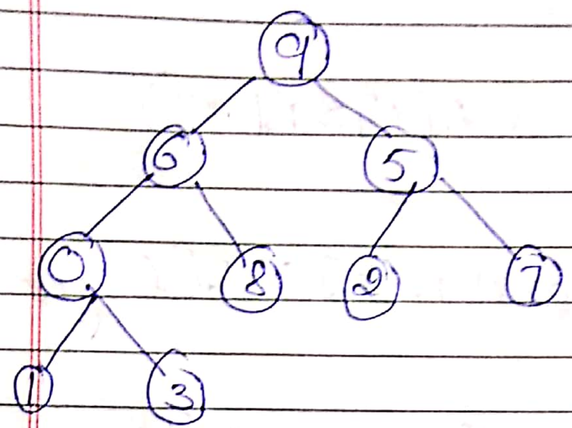
Charlie

Date

Page No.

01/8/19

9	6	5	0	8	2	7	1	3



Build_Maxheap(A)

{

A heap_size = A.length

for(i = ⌊A.length/2⌋ down to 1)

max_heapify(A, i)

}

$$\sum_{h=0}^{\log n} \cancel{O(h)}\, ch \left\lceil \frac{n}{2^{h+1}} \right\rceil < \sum_{h=0}^{\infty}$$

$2^h, 2^1$

$$\sum_{h=0}^{\log n} ch \left\lceil \frac{n}{2^h \cdot 2} \right\rceil < \sum_{h=0}^{\infty}$$

$$\frac{cn}{2} \sum_{h=0}^{\log n} \frac{h}{2^h} < \sum_{h=0}^{\infty} \frac{h}{2^h}$$

heapifying $O(n)$

Date
Page No.

Space = ~~O(n)~~ log n

23/8/19

Tute

Complexity for Recursive Program :-

Back substitution

Recursion tree

Master's Theorem

BACK SUBSTITUTION

$T(n) = 1 + T(n-1)$ —① $n > 1$

$T(n-1) = 1 + T(n-2)$ —②

$T(n-2) = 1 + T(n-3)$ —③

$T(n-3) = 1 + T(n-4)$ —④

$$
\begin{aligned}
T(n) &= 1 + 1 + T(n-2) \\
&= 2 + T(n-2) \\
&= 2 + 1 + T(n-3) \\
&= 3 + T(n-3)
\end{aligned}
$$

for k recursion

$T(n) = k + T(n-k)$

$n - k = 1$ $\quad k = n-1$

$T(n) = n \cancel{-1} + \cancel{n-n+1}$

$= n$

Complexity = $O(n)$

Page No.

$\log_{10} 10 = 1$
$\log_{10} 2 = 2$
$\log_2$

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

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

$T(n-2) = n-2 + T(n-3)$

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

$T_n = n + n-1 + n-2 + T(n-3)$

$T_n = n + n-1 + n-2 + n-3 + T(n-4)$

$T_n = n + n-1 + n-2 + n-3 \ldots n-k + T(n-(k+1))$

$n-k-1 = 1$

$n-k = 2$

$k = n-2$

$\cancel{n} - \cancel{n} + 2 + \cancel{n} - \cancel{n} + 2 - 1$

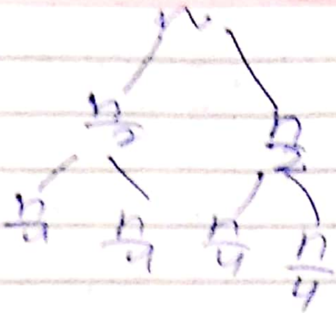
$2 + 1$

$\frac{n(n+1)}{2}$

$O(n^2)$

Date
Page No.

Recursion Tree



$T(n) = 2T\left(\frac{n}{2}\right) + C$

$C\left(2^0 + 2^1 + 2^2 + 2^3 \ldots 2^n\right)$

$C \times \log n = O(n \log n)$

~~29/8/19~~

$\boxed{\text{Counting Sort}}$

30/8/19 Tute

Master Theorem

$T(n) = aT\left(\frac{n}{b}\right) + O\left(n^k \log^p n\right)$

$a \geq 1,\ b > 1,\ k \geq 0$ and p is real no.

1) if $a > b^k$, then $T(n) = O\left(n^{\log_b a}\right)$

2) if $a = b^k$

(a) if $p > -1$, then $T(n) = O\left(n^{\log_b a} \log^{p+1} n\right)$
(b) if $p = -1$, then $T(n) = O\left(n^{\log_b a} \log \log n\right)$
(c) if $p < -1$, then $T(n) = O\left(n^{\log_b a}\right)$

3) if $a < b^k$

(a) if $p \geq 0$, then $T(n) = O\left(n^k \log^p n\right)$
(b) if $p < 0$, then $T(n) = O\left(n^k\right)$

Scanned by CamScanner

Charlie
Date
Page No.

~~Merge Sort.~~

1. $T(n) = 2T\left(\frac{n}{2}\right) + n\log n$
2. $T(n) = \cancel{8}T\left(\frac{n}{2}\right) + n^2$
3. $T(n) = 6T\left(\frac{n}{3}\right) + n^2 \log n$
4. $T(n) = \sqrt{2}\,T\left(\frac{n}{2}\right) + \log n$
5. $T(n) = 4T\left(\frac{n}{3}\right) + n^2$
6. $T(n) = 16T\left(\frac{n}{4}\right) + n$
7. $T(n) = 2T\left(\frac{n}{4}\right) + n^{0.5}$

$\cancel{T(n) =}$

(1) $T(n) = 2T\left(\frac{n}{2}\right) + n\log n$

$a = 2,\ b = 2,\ k = 1,\ p = 1$

$a = b^k = 2 = 2^1$

$T(n) = \Theta\left(n^{\log_2 2} \log^2 n\right)$

$= \Theta\left(n \log^2 n\right)$

Randomized quick sort.

Charlie

Date

Page No.

2. $T(n) = 2T\left(\frac{n}{2}\right) + n^2$

$a=2,\ b=2,\ k=2,\ p=0$

$2 < 2^2$

$T(n) = O(n^2 \log^0 n)$

$= O(n^2)$

3. $T(n) = 6T\left(\frac{n}{3}\right) + n^2 \log n$

$a=6,\ b=3,\ k=2,\ \beta=1$

$6 < 3^2$

$T(n) = O(n^2 \log n)$

$98 \% 10$

2/9/19

Radix Sort :->

98, 40, 1, 500, 67
0, 1, 2, 3, 4

$(98/1) = 98 \% 10 = 8$

exp = 1 $(98/1) \% 10$

| 0 | 1 | | | | 7 | 8 |

| 9 | 1 | 7 | 8 |

Here the recursive dynamic.

Divide & conquer
Randomization

Dynamic Programming

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

0 1 1 2 3 5 ...

```
int fib (int n)
if (n <= 1)
  return n
return fib(n-1) + fib(n-2)
```

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

Mamoization
Tabulization

Dynamic Programming :- It is an algorithmic paradigm that solves a given complex problem by breaking it into subproblems and storing the result of subproblems to avoid computing the same results again and again.

Conditions :-

1. Overlapping subproblem.
2. Optimal sub structure

Given problem is said to have optimal substructure property if an optimal solution of given solution can be obtained by using solution of its subproblems.

Chain matrix multiplication

$$\begin{bmatrix} 1 & 5 & 9 & 7 & 3 & 4 \\ 2 & 1 & 9 & 7 & 2 & 6 \\ 9 & 5 & 2 & 2 & 3 & 5 \\ 6 & 6 & 1 & 3 & 1 & 7 \end{bmatrix}_{4\times6} \begin{bmatrix} 5 & 1 & 3 \\ 9 & 5 & 1 \\ 8 & 7 & 6 \\ 9 & 6 & 8 \\ 8 & 1 & 3 \\ 2 & 2 & 9 \end{bmatrix}_{6\times3}$$

$$= \begin{bmatrix} \; \end{bmatrix}_{4\times3}$$

no. of multiplication = 72

A_1	A_2	A_3	A_4	A_5
4×10	10×3	3×12	12×20	20×7
P_1 P_2	P_2 P_3	P_3 P_4	P_4 P_5	$P_5 \times P_6$
P_0 P_1	P_1 P_2	P_2 P_3	P_3 P_4	P_4

4×3 $\quad$ 3×12

4×12 $\quad$ 12×20

$4\times20 \quad \times \quad 20\times7$

12
× 4
48

120
144
960

10
3
12

A1	A2	A3	A4	A5
4×10	10×3	3×12	12×20	20×7

120
144
960
560
1784

4×3 $\quad$ 3×12

4×12 $\quad$ 12×20

4×20 $\quad$ 20×7

1784

Goal :- Find optimal way to multiply these matrices to perform fewest multiplication

Dynamic Programming

Step 1 :- Check if the problem have optimal substructure and overlapping subproblem.

Step 2 :- Define a recursive ~~prob~~ formula.

$$M[i,j] = \begin{cases} 0 & \text{if } i = j \\ \min\limits_{i \le k < j} M[i,k] + M[k+1,j] + P_{i-1} P_k P_j \end{cases}$$

Step 3 :- Note the optimal solution

$A_1 A_2$ $A_1 A_2 A_3$ $A_1 A_2 A_3 A_4$
$A_2 A_3$ $A_2 A_3 A_4$ $A_2 A_3 A_4 A_5$
$A_3 A_4$ $A_3 A_4 A_5$ $A_1 A_2 A_3 A_4 A_5$
$A_4 A_5$

i/j →	1	2	3	4	5
1	0	120	264	1080	1344
2	x	0	360	1320	1350
3	x	x	0	720	1140
4	x	x	x	0	1680
5	x	x	x	x	0

Date 13
Page No. 20

$i=1,\ j=2$

$M[1,2] = M[1,1] + M[2,2] + P_0 P_1 P_2$
$= 0 + 0 + 120$
$= 120$

$i=2,\ j=3$

$M[2,3] = M[2,2] + M[3,3] + P_1 P_2 P_3$
$k=2$
$= 0 + 0 + 360$
$= 360$

$M[1,3] = M[1,2] + M[2,3] + P_0 P_2 P_3$
$k=2$
$= 120 + \cancel{360} + 144$
$= \cancel{624}\ 264$

$k=1 \quad = M[1,1] + M[2,3] + P_0 P_1 P_3$
$= 0 + 360 + 480 = 840$

$360 + 144 = 504$
$504 + 120 = 624$

~~$M[1,4] = M[1,3] + M[3,$~~
~~$k=2$~~

$M[3,4] = M[3,3] + M[4,4] + P_2 P_3 P_4$
$k=3$
$= 0 + 0 + 720$
$= 720$

$M[2,4] = M[2,3] + M[4,4] + P_1 P_3 P_4$
$k=3$
$= 360 + 0$

Date
Page No.

$P_0 = 4$ $\quad P_3 = 12$
$P_1 = 10$ $\quad P_4 = 20$
$P_2 = 3$ $\quad P_5 = 7$

$i = 2, \; j = 4$

$M[2,4] = M[2,2] + M[3,4] + P_1 P_2 P_4$

$k = 2 \quad = 0 + 720 + \cancel{600}\; 10 \times 3 \times 20$

$= 1320$

$k = 3 \quad = M[2,3] + M[4,4] + P_1 P_3 P_4$

$= 360 + 0 + 10 \times 12 \times 20$

$= 360 + 2400$

$= 2760$

$M[3,5] = M[3,3] + M[4,5] + P_2 P_3 P_5$

$k = 3 \quad = 0 + 1680 + 3 \times 12 \times 7$

$= 1680 + 252$

$= 1932$

$k = 4 \Rightarrow M[3,4] + M[5,5] + P_2 P_4 P_5$

$= 720 + 0 + 3 \times 20 \times 7$

$= 720 + 420$

$= 1140$

$M[1,4] = M[1,1] + M[2,4] + P_0 P_1 P_4$

$k = 1 \quad = 0 + 1320 + 4 \times 10 \times 20$

$= 1320 + 800$

$= 2120$

$M[1,4] = M[1,2] + M[3,4] + P_0 P_2 P_4$

$k = 2 \quad = 120 + 720 + 4 \times 3 \times 20$

$= 840 + 240$

$= 1080$

$$M[1,4] = M[1,3] + M[4,4] + P_0 P_3 P_4$$
$k=3$
$$= 264 + 0 + 4 \times 12 \times 20$$
$$= 264 + 960$$
$$= 1224$$

$$M[2,5] = M[2,2] + M[3,5] + P_1 P_2 P_5$$
$\boxed{k=2}$
$$= 0 + 1140 + 10 \times 3 \times 7$$
$$= 1140 + 210 = 1350$$

$$k=3 \quad = M[2,3] + M[4,5] + P_1 P_3 P_5$$
$$= 360 + 1680 + 10 \times 12 \times 7$$
$$= 2040 + 840$$
$$= 2880$$

$$k=4 \quad = M[2,4] + M[5,5] + P_1 P_4 P_5$$
$$= 1320 + 0 + 10 \times 20 \times 7$$
$$= 1320 + 1400$$
$$= 2720$$

$$M[1,5] = M[1,1] + M[2,5] + P_0 P_1 P_5$$
$k=1$
$$= 0 + 1350 + 4 \times 10 \times 7$$
$$= 1350 + 280$$
$$= 1630$$

$$M[1,5] = M[1,2] + M[3,5] + P_0 P_2 P_5$$
$\boxed{k=2}$
$$= 120 + 1140 + 4 \times 3 \times 7$$
$$= 1260 + 84$$
$$= 1344$$

1140
120
1260

36
32
72

14
×4
56

84
×4

Charlie

Date

Page No. 6

$M[1,5] = M[1,3] + M[4,5] + P_0 P_3 P_5$

$k = 3$ $\quad = 264 + 1680 + 4 \times 12 \times 7$

$= 1944 + 336$

$= 2280$

1680
264
1944

$M[1,5] = M[1,4] + M[5,5] + P_0 P_4 P_5$

$k = 4$ $\quad = 1080 + 0 + 4 \times 20 \times 7$

$= 1080 + 560$

$= 1640$

$((A_1 \; A_2) \; ((A_3 \; A_4) \; A_5))$

$4\times10 \quad 10\times3 \quad 3\times12 \quad 12\times20 \quad 20\times7$

120
(4,3)

720
(3,20)

420
(3,7)

84

120
720
420
84
1344

4/9/19

LCS → Longest common subsequence

S1 abcdefg

S2 abxdg

a, b, d, g, ab, dg, ~~abd~~ ~~abd~~,

a, b, d, g, ag, bg, bd, ab, dg, ~~abd~~, abd, abdg

LCS ("AXYT", "AYZX")

$LCS[i][j] = LCS[i-1][j-1] + 1$

$LCS[i][j] = (LCS[i-1][j], LCS[i][j-1])$

LCS("AXYT", "AYZX")

- LCS("AXY", "AYZX")
  - LCS("AX", "AYZX")
  - LCS("AXY", "AYZ")
- LCS("AXYT", "AYZ")
  - LCS("AXY", "AYZ")
  - LCS("AXYT", "AY")

Date
Page No.

LCS

	ϕ	A	G	G	T	A	B
ϕ	0	0	0	0	0	0	0
G	0	0	1	1	1	1	1
X	0	0	1	1	1	1	1
T	0	0	1	1	2	2	2
X	0	0	1	1	2	2	2
A	0	1	1	1	2	3	3
Y	0	1	1	1	2	3	3
B	0	1	1	1	2	3	4

GTAB

6/9/19 Tute.

Function Chain Matrix (P)

Step 1: Initialization loop for $(i \rightarrow 1$ to $n)$
Set mmul $[i, i] \leftarrow 0$ // $i = j$

Step 2: loop, length of subsequence
for length $\leftarrow 2$ to n
{
for $i \leftarrow 1$ to $n -$ length $+ 1$
set $j \leftarrow i +$ length $- 1$
set mmul$[i, j] \leftarrow \infty$
}

for $k \leftarrow i$ to $j-1$
{ set $q \leftarrow$ mmul$[i, k]$ + mmul$[k+1, j]$
$+ P[i-1] \times P[k] \times P[j]$

if (q < mmul[i, j] then
{
Set mmul[i,j] ← q
Set s[i,j] ← k

Step 3: return mmul[i,j] { s[i,j]
}
}
}

9/9/19

chain matrix
heap sort (algo + example + analysis)
Diff b/w Quick Sort and Randomize quick sort

11/9/19

A1	A2	A3	A4
5×4	4×6	6×2	2×7

	1	2	3	4
1	0	120	88	
2		0	48	104
3			0	84
4				0

	1	2	3	4
1	0	1	1	
2		0	2	~~2~~ 3
3			0	3
4				0

0/1 KNap-Sack Problem

W → maximum capacity.
Total wt. 7

wt	Val
1	1
3	4
4	5
5	7

	wt	0	1	2	3	4	5	6	7
(1)	1	0	1	1	1	1	1	1	1
(3)	2	0	1	1	4	5	5	5	5
(4)	3	0	1	1	4	5	6	6	6
(5)	4	0	1	1	4	5	7	8	9

11/9/19

wt	value
1	1
3	4
4	5
5	7

		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
(1)	1	0	1	1	1	1	1	1	1
(3)	2	0	1	1	4	5	5	5	5
(4)	3	0	1	1	4	5	6	6	~~6~~ 9 ↑
(5)	4	0	1	1	4	5	7	8	9

Date
Page No.

Algorithm :-

Step 1 :- loop, initialize

for $w \leftarrow 0$ to W
  set $KP[0, w] \leftarrow 0$

Step 2 :- loop
  for $(i \leftarrow 1$ to $n)$
    set $Kp[i, 0] \leftarrow 0$

Step 3 :- loop, checking & whether part of solution or not.

for $(i \leftarrow 1$ to $n)$
  for $w \leftarrow 0$ to W
    if $(w_i \le w)$ then item i can be part of solution,
      if $(v_i + Kp[i-1, w-w_i] > Kp[i-1, w]$ then
        set $Kp[i, w] \leftarrow v_i + Kp[i-1, w-w_i]$
      else
        set $Kp[i, w] \leftarrow Kp[i-1, n]$
    else
      set $Kp[i, w] \leftarrow Kp(i-1, w] \rightarrow w_i > w$

Step 4 :- Finished.

5	3	2	1	4	0

heap size - 6

for (6/2 → 1)

for (3 → 1)

i = 3

l = 2i = ~~6~~

r = ~~7~~

3

Date
Page No.

5	4	1	2	6	8
0	1	2	3	4	5

$\frac{6}{2} - 1 = 2$

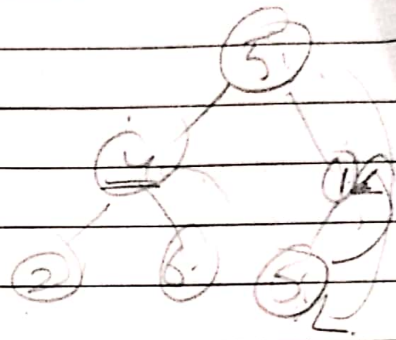
largest : 2

$l = 2 * i + 1$

$= 5$

$r = 2 * i + 2$

$= 7$



6 , [ ]

2   $5 < 6$ $arr[5] > 1$

$5 > 1$ $i = 5$

largest = $l = 5$

$7 < 6$

arr, 6, 5

largest = 5

$l = 2 * 6 = 12$