1 — Time & Space analysis

1.1 Introduction to asymptotic notations

Asymptotic notation

1) Big (Oh) O

$f(n) = 3n+2 \quad g(n) = n$

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

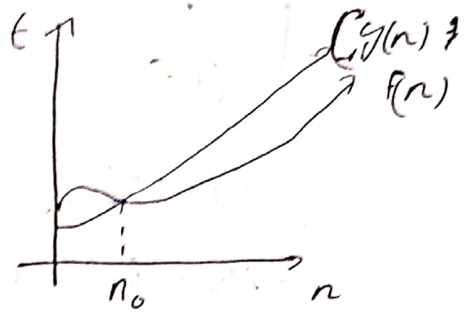
$f(n) \leq cg(n)$

$3n+2 \leq cn$

$\boxed{c = 4}$

$3n+2 \leq 4n$

$\boxed{n \geq 2}$



$f(n) \leq cg(n)$

$n \geq n_0$

$c > 0, \; n_0 \geq 1$

$\boxed{f(n) = O(g(n))}$

$g(n) : n \rightarrow$ we prefer — least upper bound

n^2

n^3 — highest

$\boxed{f(n) = O\,C\,g(n)}$

2) Big omega $-\Omega$

$f(n) = 3n+2$

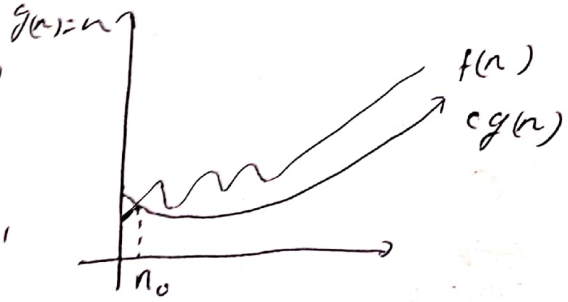
$f(n) = \Omega g(n)$

$f(n) \geq cg(n)$

$3n+2 \geq cn$

$c = 1$

$3n+2 = \Omega(n)$



$f(n) \geq cg(n), \; n \geq n_0$

$c > 0, \; n_0 \geq 1$

$f(n) = \Omega(n)$ — we prefer tightest lower bound

$\Omega(n)$

↓

$\log n$

↓

$\log \log(n)$

3) Big theta θ

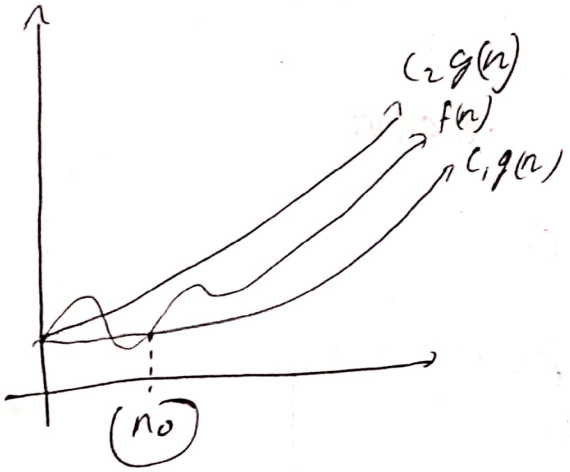
$f(n) = 3n+2 \quad g(n) = n$

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

$3n+2 \leq g(n) \quad n_0 \geq 1$

$f(n) \geq c_1 g(n)$

$3n+2 \geq n \quad , n_0 \geq 1$



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

$c_1 g(n) \leq f(n) \leq c_2 g(n)$

$c_1, c_2 > 0$

$n \geq n_0$

$n_0 \geq 1$

$3n^2 + n + 1 = \theta(n^2)$ highest power

$3n^3 + n^2 = \theta(n^3)$

O — Least

Ω — best

θ — ~~avegi~~ average

1.2 Time complexity Analysis of iterative programs

(a)
```
A()
{
int i,j;
for(i=1 to n) -> n
    for(j=1 to n) -> n×n
        Print(a.vi);
}
```
$O(n)^2$ - complexity

(a)
```
A()
{
i=1;
for(i=1; i<=√n; i++)
Print("a.vi");
}
```
$O(\sqrt{n})$

(a)
```
A()
{
int i,j,k,n;
for(i=1; i<=n; i++)
{
  for(j=1; i<=i; j++)
  {
    for(k=1; k<=100; k++)
    {
      Pf("a.vi");
    }
  }
}
```

(a) 1991
```
A()
{ i=1; s=1;
while(s<=n)
{
  i++
  s=s+i
  Pf("a.vi");
}
}
```

s	1	3	6	10	---	$\frac{k(k+1)}{2}$
i	1	2	3	4	--	k

According to while statement

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

$\frac{k^2+k}{2} > n$

$k = O(\sqrt{n})$

i = 1	i = 2	i = 3	--- i = n
j = 1	j = 2	j = 3	--- j = n
k = 100	k = 2×100	j = 3×100	--- k = n×100

total time = 100 + 200 + 300 + --- n×100

100 (1 + 2 + 3 + --- n)

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

$= O(n^2)$

Q 6

binary search

A ← sorted array
n ← size of array
x ← value to be searched

Set start = 0
Set end = n

While x not found
    if end < start
    Exit : x doesnt exist

    Set midpoint = (start + end) / 2

    if A[mid point] < x
        Set ~~start~~ start = midpoint + 1

    if A[mid point] > x

        Set end = mid point - 1

    if A[mid point] = x
        Exit : x is at location mid point

end while

end

2.2 Merge sort algorithm, analysis and problems

```
Merge(A, p, q, r)
{
  n1 = q-p+1
  n2 = r-q
  Let L[1 ... n1+1] & R[1 to n2+1] be new array
  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
}
```

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

$= \theta(n \log n)$ → best & worst

```
Merge-sort (A p r)   T(n)
{ if p<r
  q = L(p+r)/2]
  Merge-sort (A, p, q)   T(n/2)
  Merge-sort (A, q+1, r)   T(n/2)
  Merge (A, p, q, r)   O(n)
}
```

Quick sort

```
Partition(A, l, h)
{
    Pivot = A[l]
    i = l, j = h;
    while (i < j)
    {
        do
        { i++;
        } while (A[i] <= Pivot);
        do {
            j--;
        } while (A[j] > Pivot);
        if (i < j)
            swap (A[i], A[j]);
    }
    swap(A[l], A[j]);
    return j;
}

Quicksort (A, l, h)
    if (l < h)
    {
        j = partition(A, l, h);
        Quicksort (A, l, j);
        Quicksort (A, j+1, h);
```

Strassen's Multiplication

1 $P = (A_{11} + A_{22}) \times (B_{11} + B_{22})$

2 $Q = (A_{21} + A_{22}) \times B_{11}$

3 $R = A_{11} (B_{12} - B_{22})$

4 $S = A_{22} \times (B_{21} - B_{11})$

5 $T = (A_{11} + A_{12}) \times B_{22}$

6 $U = (A_{21} - A_{11}) \times (B_{11} + B_{12})$

7 $V = (A_{12} - A_{22}) \times (B_{21} + B_{22})$

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

$C_{11} = P + S - T + V$

$C_{12} = R + T$

$C_{21} = Q + S$

$C_{22} = P + R - Q + U$

1 + 4 − 5 + 7

3 + 5

2 + 7

1 + 3 − 2 + 6

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

1 3 2 5 7 6 4

3 2 4 7 6 5

1 2 3 4 5 6 7

3 Greedy Algorithm

3.1 Introduction to greedy Algorithm

Optimization → Greedy → $O(n^k)$ ✓, $O(2^n)$ ✗
→ Dynamic program

min cost
max profit
max reliability
min risk

O(n!) or O(n^n) O(2^n)

3.2 Greedy knap sack algorithm

(1)

$m = 20$

$n = 3$

	Obj 1	Obj 2	Obj 3
Profit	25	24	15
Weight	18	15	10
P/W	$\frac{25}{18} = 1.4$	$\frac{24}{15}$ 1.6	$\frac{15}{10}$ 1.5

Greedy about profit

	W	P
Obj 1	18	25
Obj 2	2	$24\left(\frac{2}{15}\right)$
	20	28.2

Greedy about Weight

	W	P
Obj 3	10	15
Obj 2	10	$\frac{24}{3}\left(\frac{10}{15}\right)$
		31

Greedy about P/W

	W	P
Obj 2	15	24
Obj 3	5	$15\left(\frac{5}{10}\right)$
		31.5

Greedy knapsack

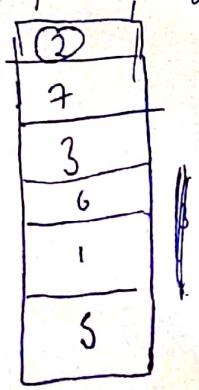
```
{
  for i=1 to n;
     compute Pi/wi;          — O(n)
  Sort object in non increasing order O(n log n)
  for i=1 to n
     if (m >= wi)            } O(n)
     m = m - wi;
     p = p + Pi;
     else break;                                  O(n log n)
  if (m > 0)
     p = p + Pi (m/wi);      O(1)
}
```

Q m = 15 n = 7

object	1	2	3	4	5	6	7
Profit	10	5	15	7	6	18	3
Weight	2	3	5	7	1	4	1
	5	1.6	3	1	6	4.5	3

6 5 4.5 3 3 1.6 1

Obj no (5 1 1 1 6, 3, 7, 2, 4)



m = ~~15 14 12 4~~ 0 2

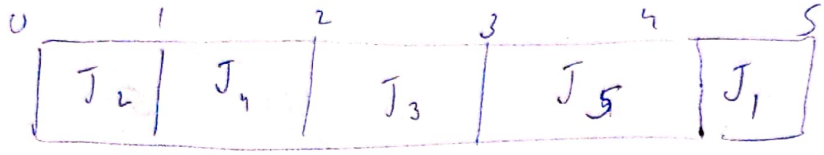
P = 6 + 10 + 18 + 15 + 3 + $\frac{2}{3}$ × 5

= 55.3

Q

	J_1	J_2	J_3	J_4	J_5	J_6
D	5	3	3	2	4	2
P	200	180	190	300	120	100

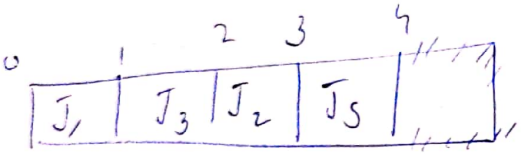
avg case - $O(n \log n)$



worst case $O(n^2)$

Q

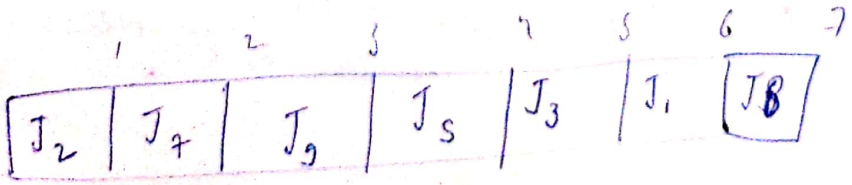
Jobs	J_1	J_2	J_3	J_4	J_5
Profits	2	4	3	1	10
deading	3	3	3	4	4



max profit - 2 + 3 + 4 + 10 = 19

Q

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9
P	15	20	30	18	18	10	23	16	25
D	7	2	5	3	4	5	2	7	3



Profit = 30 + 25 + 23 + 20 + 18 + 16 + 15

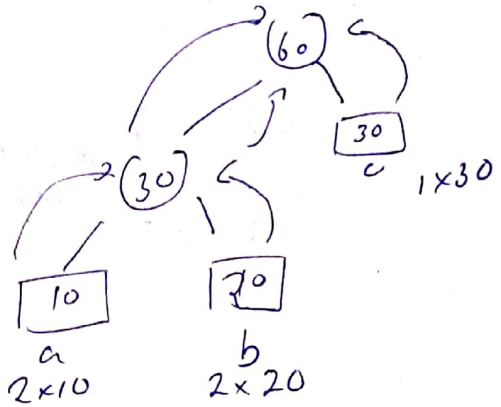
3.7 optimal merge patterns:

Files a b c

10 20 30

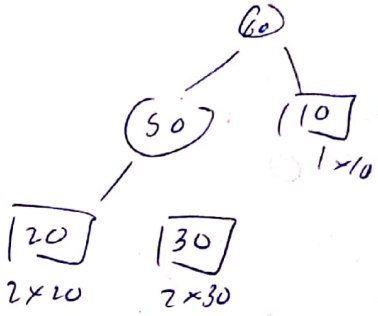
records

We need least movement



Total record movement

20 + 40 + 30 = 90



= 110 ×

min heap (n)

2 min $o(\log n)$ } $(n-1)$ times

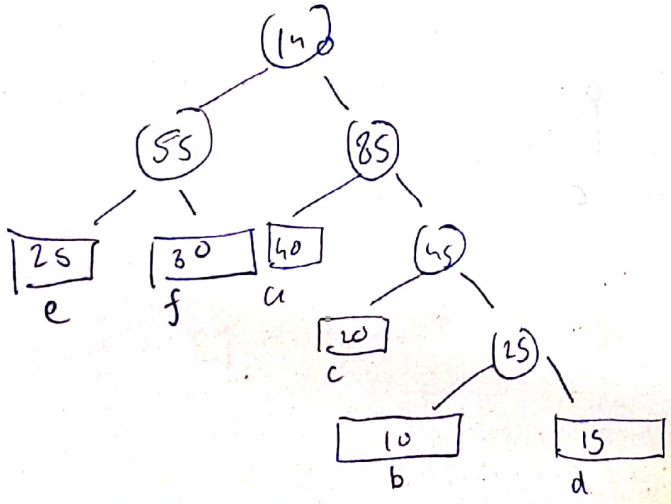
insert $o(\log n)$

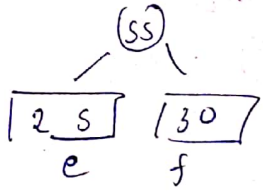
$o(n \log n)$

a a b c d e f

40 10 20 15 25 30

25 45 85 55

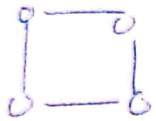




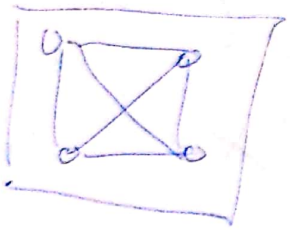
3.3 Introduction to spanning tree and kirchhoff theorem

Graph

simple



min = 0



max edge = $n C_2$

$= \frac{n(n-1)}{2}$

$= \frac{n^2 - n}{2}$

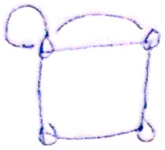
$E = O(n^2)$

$= O(v^2)$

$v : O(\log E)$

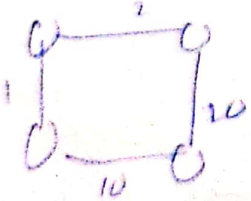
$E : O(v^2)$

multi



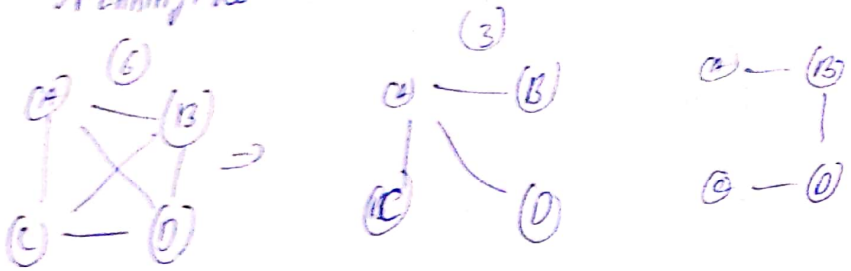
more than one edge b/w 2 nodes

weighted



unweighted

Spanning tree



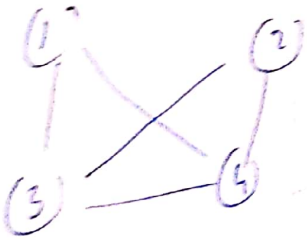
Possible spanning tree - (3)

no of spanning tree $K_n - n^{n-2}$ → for complete graphs only

$K_3 - 3^{3-2} = 3$

$K_4 = 4^{4-2} = 16$

Kirchoff theorem



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

(i) Digond 0's → degree of node

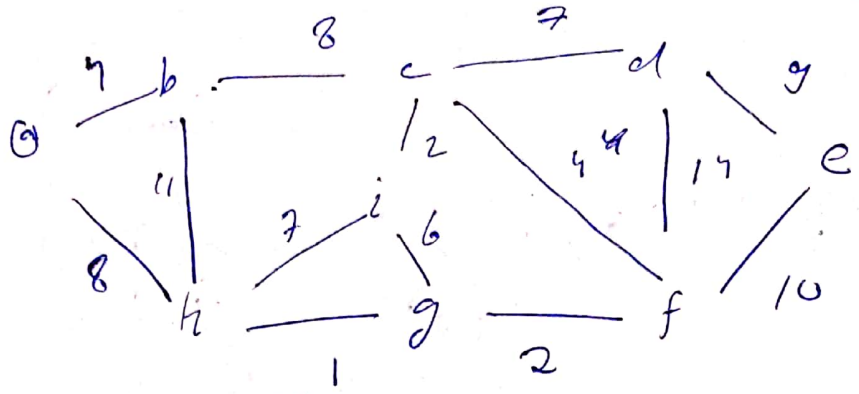
(ii) non digond 1's → "-1"

(iii) non digond 0's → '0'

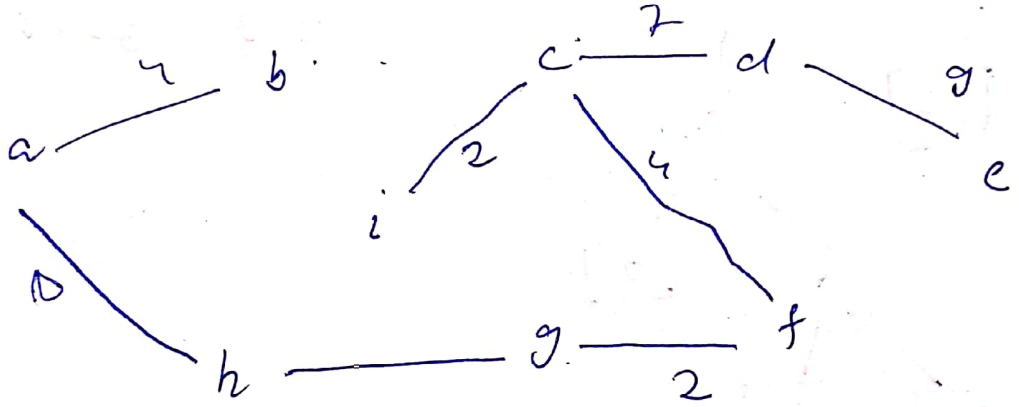
	1	2	3	4
1	2	0	-1	-1
2	0	2	-1	-1
3	-1	-1	3	-1
4	-1	-1	-1	3

no of ST - co factor of any element $= 1(2(9-1)-(-1)(-3-1)-1(+3)$

$= 8$



Prim's



Kruskal