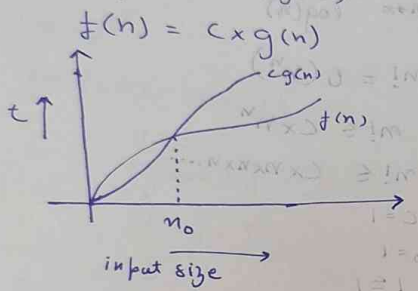
Design and Analysis of Algorithm

Time Complexity

- Time - Space tradeoff
- Asymptotic Notation
  ↳ 3

$\Omega(\ )$ → Best case
$\Theta(\ )$ → Average case
$O(\ )$ → Worst case
↳ tightest/closest bound

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



e.g. $f(n) = 3n+2$ $g(n) = n$

$f(n) \leq c\,g(n)$

$3n+2 \leq c \times n$

$c = 4$ $n_0 = 8$

$O(\ )$ → Big Oh considers tighest bound

$o(\ )$ → Small oh never considers tighest bound

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

→ to check if there is relation b/w two functions using small oh
i.e. $f(n) = O(g(n))$

$f(n) = 2n^2+5$, $g(n) = n^2$

$$\lim_{n\to\infty} \frac{2n^2+5}{n^2} = 2 + \lim_{n\to\infty} \frac{5}{n^2} = 2 \neq 0$$

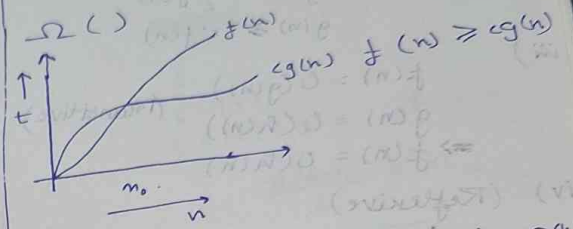
e.g. $f(n) = 7n+8$ $g(n) = n^2$

$$\lim_{n\to\infty} \frac{7n+8}{n^2} = \lim_{n\to\infty} \frac{7}{n} + \frac{8}{n^2}$$

$0 = 0$

$\Rightarrow f(n) = o(g(n))$

$\Omega(\ )$



small omega → ω $f(n) > g(n)$
(don't consider tight bound)

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

→ if this is true then small omega relation exists

e.g. $f(n) = 4n+6$ $g(n) = 1$

$$\lim_{n\to\infty} \frac{4n+6}{1} = \infty$$

$\infty = \infty$

$\therefore \omega()$ exists for this relation

$\Theta(\ )$ $\Omega < \Theta < O$

e.g. $f(n) = 2^{n+1}$ $g(n) = 2^n$

$f(n) < g(n)$

$2^{n+1} \leq 2^n \times c$

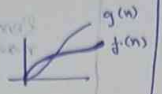
$2 \cdot 2^n \leq c \times 2^n$

$c = 3$, $n_0 = 6$

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

i) $a * f(n) = O(g(n))$

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



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

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

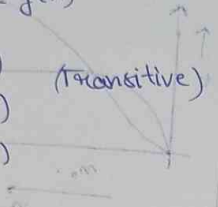
$g(n) \geq \frac{1}{c} f(n)$

$g(n) \geq c' f(n)$

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

$g(n) = O(h(n))$

$\Rightarrow f(n) = O(h(n))$



iv) (Reflexive)

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

v) (Symmetric)

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

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

i) $2^{n+1} = O(n^2)$

$f(n) \leq g(n) \times c$

$2^{n+1} \leq c \times 2^n$

$2^n \cdot 2 \leq 2^n \times c$

$c = 2 \quad n_0 = 1$

ii) $2^{2n} \neq O(2^n)$

Assume $2^{2n} = O(2^n)$

$2^{2n} \leq c \times 2^n$

$2^n \cdot 2^n \leq c \times 2^n$

$2^n \leq c$

variable constant

iii) $n^2 \neq O(n)$

Assume $n^2 = O(n)$

$n^2 \leq c \times n$

$n \leq c$

$\therefore n^2 \neq O(n)$

iv) $\log n + \log(\log(n)) = O(\log(n))$

$\log n + \log(\log(n)) \leq c \times \log(n)$

$\frac{\log(\log(n))}{\log(n)} \leq c$

$\lim_{n \to \infty} \frac{\log(\log(n))}{\log(n)} = 0$

e) $n! = O(n^n)$

$n! \leq c \times n^n$

$n! \leq c \times n \times n \times n \ldots$

$c = 1$

Let $n_0 = 1$

$1 \leq 1$

f) $\sum_{i=1}^{n} \log i = O(n \log(n))$

$\log 1 + \log 2 + \ldots \log(n) = c \times n \times \log(n)$

$\log(n!) \leq \log(n^n) \Rightarrow n \log(n)$

g) $4n + 3 = \Omega(n)$

$4n + 3 \geq cn$

$c = 4$

$4n + 3 \geq 4n$

$n_0 = 1$

h) $4n^3 + 3n + 2 = \Omega(n^3)$

$4n^3 + 3n + 2 \geq c \times n^3$

$c = 4$

$4n^3 + 3n + 2 \geq 4n^3$

$n_0 = 1$

i) $8n^2 + 7n = \Theta(n^2)$

ii) $8n^3 + 7n^2 + 3n = \Theta(n^3)$

Q-2 Find O, Ω, Θ

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

$O(n^2)$

b) $f(n) = n^2 \log n + n$

c) $f(n) = n!$

d) $f(n) = \log(n!)$

$(\log(n))$

$\log(n)$

```
i=1; s=1
while (s<=n)
{
    i++;
    s=s+i;
}
i=i+1
```

3 → n 7 → n 13 → n

i = 1 s = 1

$s = 1+2$

$s = 1+2+3+\ldots+k \geq n$

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

$k = \sqrt{n}$

e.g.

$n = 2^{2^k}$

21 JAN 2024

```
for (i=1 -> n) {
    j = 2
    while (j <= n)
    {
        j = j^2;
    }
}
```

2	$\frac{n(n+1)(2n+1)}{6}$	$n(k+1)$
4		$n(\log(\log(n))+1)$
16		$n\log(\log(n))$
256		
⋮		

(n)

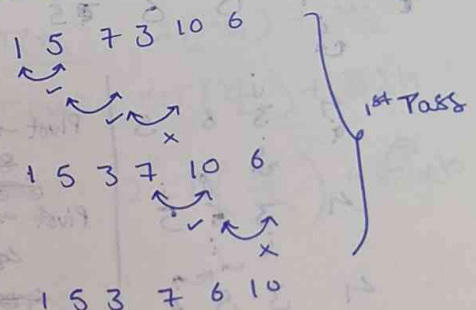
(n)

Q:-

i) $T(n) = 2T(\frac{n}{2}) + c$

ii) $T(n) = 2T(\frac{n}{2}) + n$

Bubble sort



Worst case $O(n^2)$

1 2 3 4 5 Best case

$O(n)$

e.g.

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

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

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

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

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

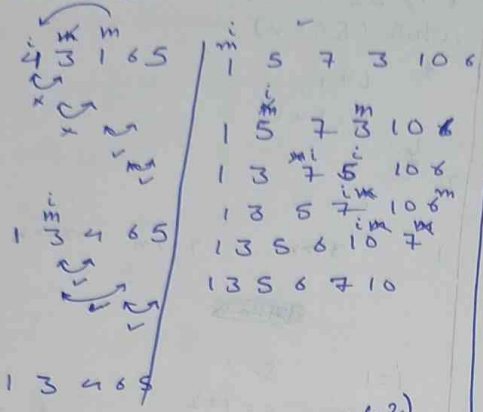
$T(n-3) = T(n-4) + n-4$

- $T(n) = T(n-4) + 4n - 7$

$T(n) = T(n-(n-1)) + n^2 - \ldots$

n^2

Selection Sort



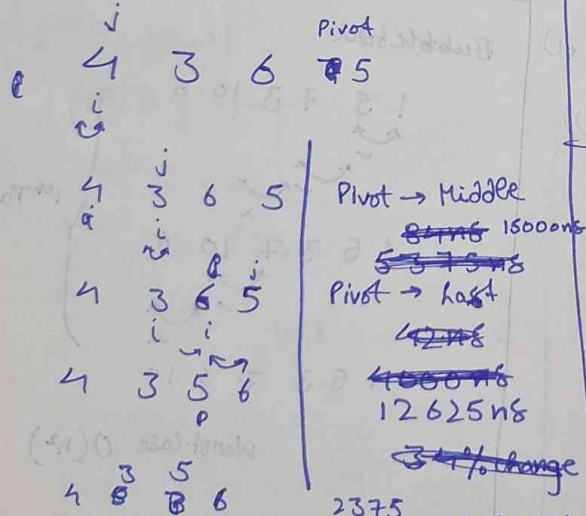
Space comp $O(1)$ $O(n^2)$ $\Omega(n^2)$

Insertion Sort

$O(n^2)$

Space comp $O(1)$ $\Omega(n)$

Quick sort



Pivot → Middle

~~8776~~ 15000ns

~~5375~~

Pivot → last

~~1248~~

~~10000~~

12625ns

~~3% change~~

$\frac{2375}{12625} \rightarrow 18.81\%$

Space Complexity - $O(n)$

~~$O(n\log$~~

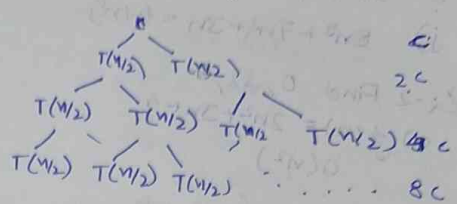
$\Omega(n\log n)$

$O(n^2)$

middle 2100

last 19125

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



$c(1+2+4+\dots)$

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

$\frac{c(2^{\log n}-1)}{2-1}$

$c(n-1)$

$= O(n)$

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

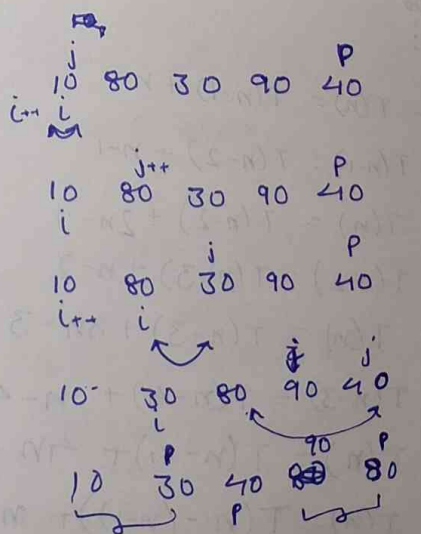
$T(n/2) = 2T(n/4) + \frac{n}{2}$

$T(n) = 4T(\frac{n}{4}) + 2n$

$T(n/4) = 2T(n/8) + n/4$

$T(n) = 8T(n/8) + 3n$

$T(n) = 2^n T(\frac{n}{2^n}) + n \times n$



Merge

$O(ne$

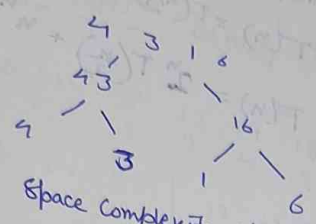
$\Omega(n$

Qui

Merge Sort

$O(n \log n)$

$\Omega(n \log n)$



Space complexity $O(n)$

$n(1+2+4+8\ldots)$

$n\left(\frac{2^{\log_2 n}-1}{2-1}\right)$

$n($

Quick Sort

Pivot = ~~Starting~~ 14

Ascending → ~~2~~625 ns

Descending → 27833 ns

Ascending 17 291

Descending 24 791

43.37%

Pivot Middle

Descending - 15 458

57417

23K

17917

23K

22458

i) $T(n) = 2T(n/2) + c$

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

$T(n) = 4T\left(\frac{n}{4}\right) + 3c$

$= 2^k T\left(\frac{n}{2^k}\right) + 2^{\cdots} c$

$2^k = n$

$k = \log n$

$T(1) = 1 = \frac{n}{2^k}$

$\frac{1(2^{k+1}-1)}{2-1}$

$= 2^{k+1} = 2^k = 2^{\log_2 n}$

$= n$

$O(n)$

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

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

$T(n) = 4T\left(\frac{n}{4}\right) + 2n$

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

$\frac{n}{2^k} = 1$

$k = \log n$

$= 2^{\log_2 n} T(1) + \log n \times n$

$= (\log n) \times n$

$1 < \log n < \sqrt{n} < n < n^2 < n^{\log 3}$

22nd JAN 2024

if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ if $0 \le c < \infty$ then big Oh exists (O)

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ $\quad 0 < c \le \infty$ then Ω exist

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ $\quad 0 < c < \infty$ then Θ exists

Insertion Sort $O(n^2)$, $\Omega(n)$

j i n

4 3 1 6

```
for (i=1 ⟶ n) {
    key = arr[i];
    j = i-1;
    while ( key < arr[j] && j >= 0)
    {
        swap (arr[j], key);
        j--;
    }
    arr[j+1] = key;
}
```

Q-1

```
while (m != n)
{
    if (m > n)
    {
        m = m - n;
    }
    else
    {
        n = n - m;
    }
}
```

$O(\log(\min(m,n)))$

$O(\log(n))$

Q-2 ~~$T(n) = T(n/3) + T(n/3)$~~

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

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

Sol":

3. $n\left(\frac{1}{3}\right)^i = 1$

$i = \log_3(n)$

$O(n \log(n))$

4. $T(n) = 8T(n/2) + n^2$

$T(n) = O(n^n)$ $\quad n^{\log_2 8} = n^3$

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

$T(n) = aT(n/b) + \Theta(n^k \log^p n)$

if $a > b^k$ $\quad T(n) = \Theta(n^{\log_b a})$

if $a = b^k$

a) if $p > -1$ $\quad T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$

b) if $p = -1$ $\quad T(n) = \Theta(n^{\log_b a} \log(\log n))$

c) if $p < -1$ $\quad T(n) = \Theta(n^{\log_b a})$

if $a < b^k$

a) if $p \ge 0$ $\quad T(n) = \Theta(n^k \log^p n)$

b) if $p < 0$ $\quad T(n) = \dots$

Selection Sort

Find min^m and swap n times

Merge Sort $O(n \log n)$

16 14 2 12 19 18
0 1 2 3 4 5

$\frac{0+5}{2} = 2$

16 14 2 | 12 19 18

$\frac{0+2}{2} = 1$

16 14 | 2   12 19 | 18

16 | 14 | 2 | 12 | 19 | 18

```
MD(P, r)
{ if (P < r) {
    a = (P+r)/2
    MD (P, a);
    MD( a+1, r);
    MS(P, a, r);
}

MS (P, a, r)
{
   create L1 of size(x1) P to a;
     "    L2 "   "(x2) a+1 to r;
   copy   A[P→a] to L1
     "    A[a+1→r] to L2;
       L1[x1+1] = ∞;
       L2[x2+1] = ∞;
}
i = j = 0     K = P
while (K < P)
{  if (L1[i] < L2[j])
     { A[K++] = L1[i++]; }
   else {
       A[K++] = L2[j++];
} } }
```

16 14 2 12 19 18
14 16 2 12 19 18
2 14 16 12 18 19

2	12	14	16	18	19

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

$O(n \log n)$

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

$T(n) \rightarrow n$

$T(n/3) \rightarrow n/3$

$T(2n/3) \rightarrow \frac{2n}{3}$

$\frac{n}{3^k} = 1$

$n = 3^k$

$k = \log_3 n$

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

$T(n/2) = 2^{n/2} T(n/4) + \left(\frac{n}{2}\right)^{n/2}$

$T(n) = 2^n [T(n/2)] + n^n$

$= 2^n \left[2^{n/2} T(n/4) + \left(\frac{n}{2}\right)^{n/2}\right] + n^n$

$= 2^{3n/2} T(n/4) + \frac{2^n \times n^{n/2}}{2^{n/2}} + n^n$

$= 2^{3n/2} T\left(\frac{n}{4}\right) + 2^{n/2} \times n^{n/2} + n^n$

Heap sort (nlogn)

Quick sort (nlogn)

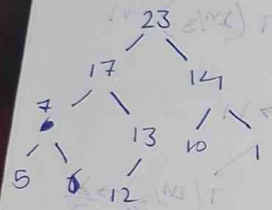
Heap Sort

4 7 2 3 9 6

Min heap → Reverse Sorted (Descending order)

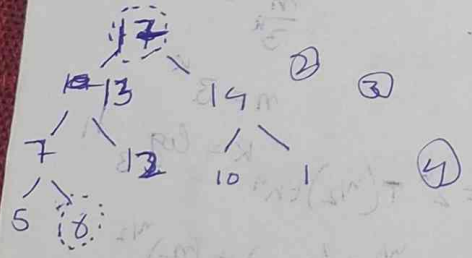
Max heap → Sorted

23, 17, 14, 6, 13, 10, 1, 5, 7, 12



23	17	14	7	13	10	1	5	6	12

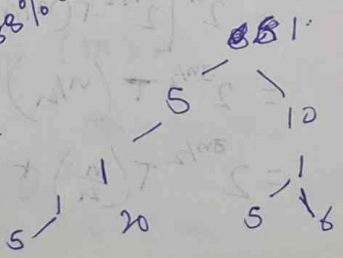
12	17	14	7	13	10	1	5	6	23

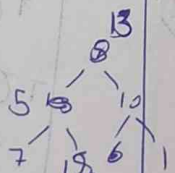


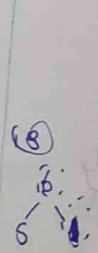
88% slower

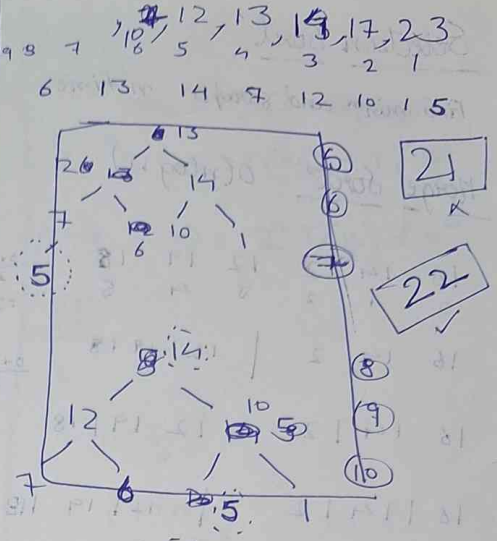
37916 ns

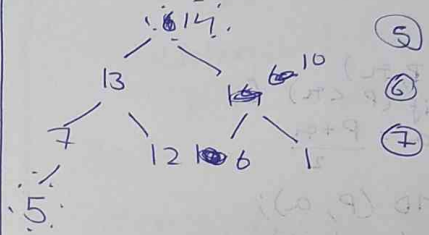
→ 329542

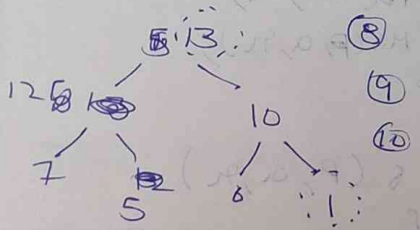


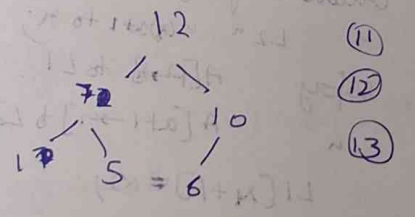


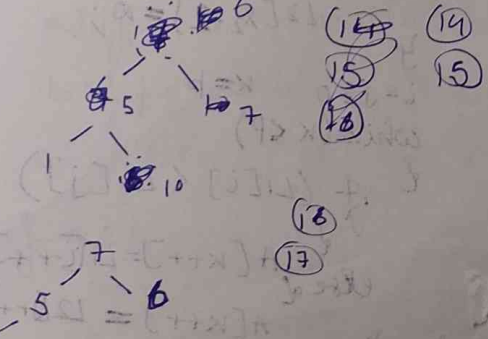












Radix Sort

$O(d \times O(\text{sorting algo}))$

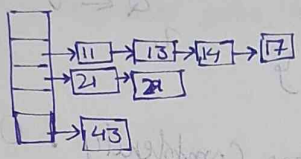
- Stable Sort

129
226
317
416
111
120
630

Bucket Sort

$\sum_{i=1}^{n} O(n^2)$

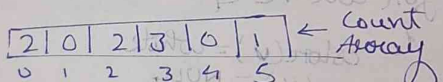
11
12
21
14
29
13
43

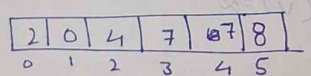


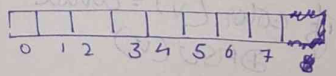
Counting Sort

2, 5, 3, 0, 2, 3, 0 3

Range - 0 to 5







Quick Sort

Lomuto Partition

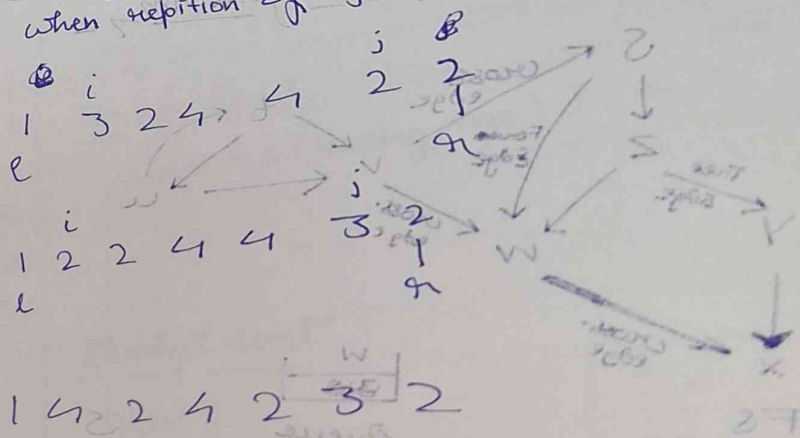
```
PA(A, P, q)
{
  x = A[q]
  i = -1, j = 0,
  while(j < n-2)
  { if(A[i] < x){
      i++;
      swap(arr[i], arr[j]);
    }
    swap(A[i+1], A[n]);
    return i+1;
}
```

Hoare Partition

one index from left, one from right

3-Way Quick Sort

when repition of digits is high



```
while (true) {
    i = -1;
    p = -1;
    j = r-1;
    while (A[i] < A[r]) {
        while (A[i] > A[r]) {
            j--;
            if (j>i) break;
        }
        swap (A[i], A[j])
        swap (A[++p], A[i])
        swap (A[--j], A[j])
    }
    swap (A[++i], A[r]);
    swap (A[q→p])
        A[p+1→r]
    swap (A[q to r-1],
        A[j→q-1])
```

- FLAT
- DAA (Quick sort)

π → Parent

→ source node

```
2. color(S) = Grey
3. π(S) = NULL
4. d(S) = 0
5. Q ← S
6. while (Q != NULL)
7.     u = Dequeue(Q)
8.     for all v ∈ adj u {
           if (color(v) == white)
           {
               π(v) = u
               d(v) = d(u)+1;
               color(v) = Grey
               Q ← v
           }
       }
```

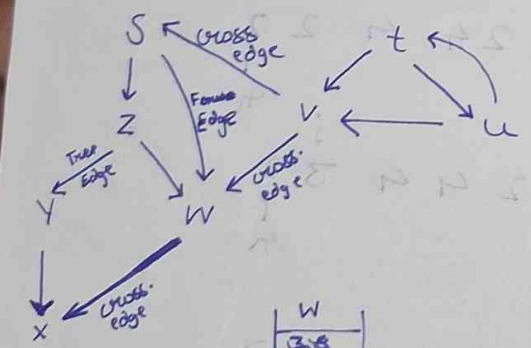
O(E)

Time Complexity : $O(E+V)$

if graph is very dense
the T.C : $\Theta(E)$

3rd Feb 2025

Graph Algorithm



BFS

W

Queue S

```
1. for all v ∈ G
   {
       color(v) = white
       π(v) = NULL
       d(v) = ∞
   }
```

O(v)

```
DFS
time = 0
for all v ∈ G
{
    color(v) = white
    π(v) = NULL
    d(v) = ∞
}
for all v ∈ G
{
    if color(v) = white
        DFSU(v)
}
```

O(v)

```
DFSU(u)
{
    color(u) = Grey
    time++;
    d(u) = time,
    for all v ∈ adj(v)
```

```
{
    if (color(v) = white) {
        π(v) = u
        DFS(v)
    }
    time++;
    color(u) = black;
    f(u) = time;
}
```

O(E)

T.C. : O(V+E)

Forward Edge → connects with parent, ~~child~~ and grand child

Cross Edge → connects one tree to other (unrelated trees)

Tree Edge → connects parent with child

Back Edge → connects from child to parent

Topological Sorting using DFS

t, u, v, s, z, w, y, x

← Right to left

~~Q. Call DFS for the~~

1. Call DFS for the entire graph
2. Once a vertex is finished insert it in the front of linked list
3. Return the linked list

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

$T(n) = aT(n/b) + \Theta(n^k \log^p n)$

1. if $a > b^k$
   $T(n) = \Theta(n^{\log_b a})$
2. if $a = b^k$
   a) if $p > -1$ $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$
   b) if $p = -1$ $T(n) = \Theta(n^{\log_b a} \log\log(n))$
   c) if $p < -1$ $T(n) = \Theta(n^{\log_b a})$
3. if $a < b^k$
   a) if $p >= 0$ $T(n) = \Theta(n^k \log^p n)$
   b) if $p < 0$ $T(n) = \Theta(n^k)$

Radix Sort

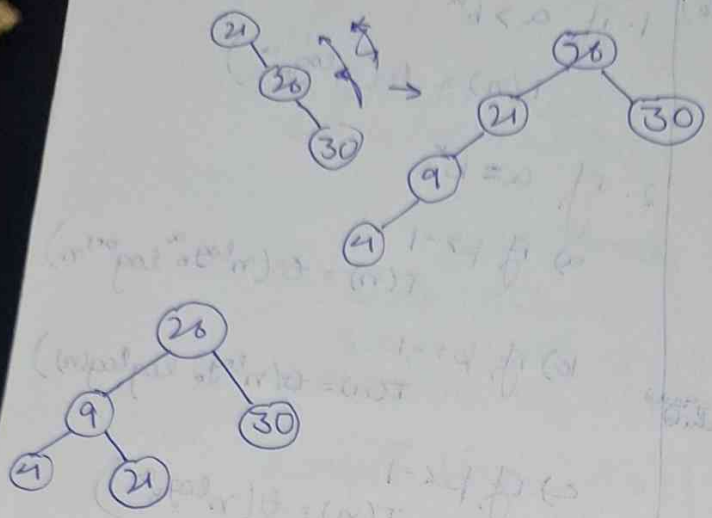
O(d * O(Sorting Algo))

Bucket Sort

$O(n^2)$

Space Comp. : O(n+k)

21, 26, 30, 9, 4, 14, 28, 18, 15, 10, 2, 3, 7



11th Feb 2025

Strongly connected Graph

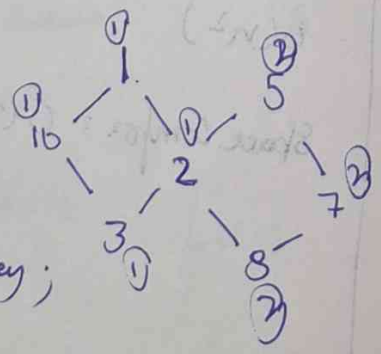
↳ there is a path from one vertex to every vertex.

Strongly_Connected component

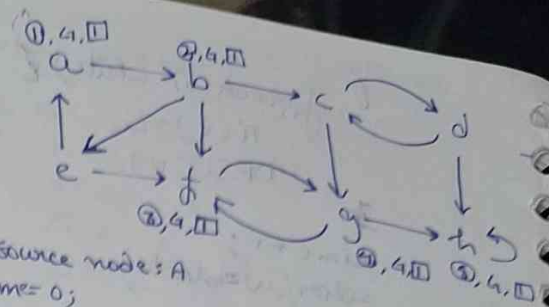
DFS

```
for all V∈G
    color(V) = white;
int id = 0;
for all V∈G
{
    if (color(V) == white)
    {
        id++;
        DFSV(V)
    }
}
```

DFSV(V)



Color(V) = grey;
comp(V) = id;

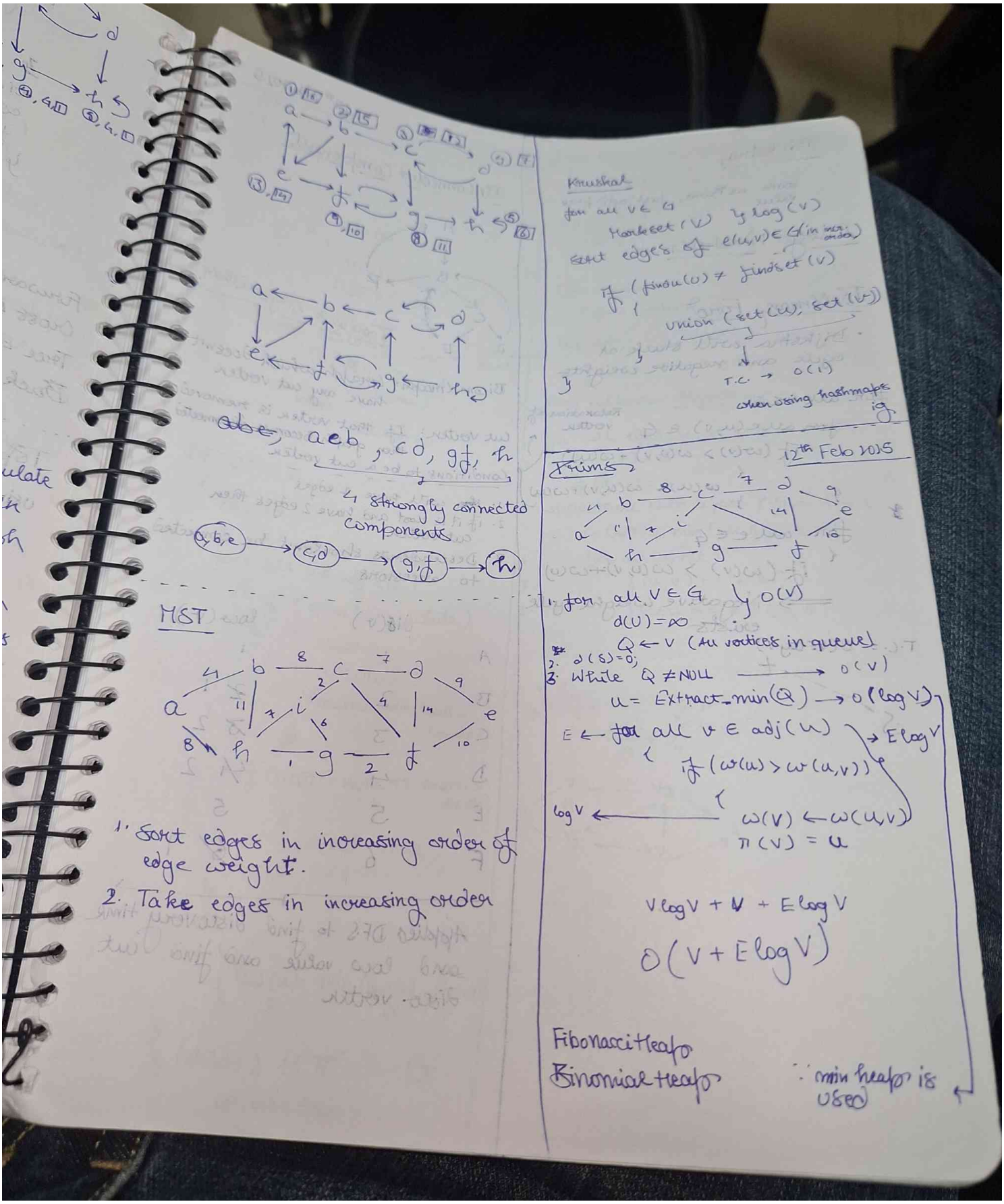


Source node: A

```
time = 0;
for all V∈G
{
    color(V) = white;
}
for all V∈G
{
    if (color(V) == white)
    {
```

Algo

1. Apply DFS on graph and calculate the finish time of every vertex
2. Calculate transpose of graph GT.
3. Call DFS on GT, but in the main loop of DFS, consider the vertices in the decreasing order of finish time.
4. Output the vertices of each tree in the depth first forest computed at step no. 3.


Kruskal
for all v ∈ G
Makeset(v) ↳ log(v)
sort edges of e(u,v) ∈ G (in incr. order)
if (finds(u) ≠ finds(v))
union(set(u), set(v))
T.C. → O(1)
when using hashmaps
12th Feb 2015
Prims
1. for all V ∈ G ↳ O(V)
d(V) = ∞
Q ← V (all vertices in queue)
2. d(s) = 0;
3. While Q ≠ NULL → O(V)
u = Extract_min(Q) → O(log V)
for all v ∈ adj(u) → E log V
if (w(u) > w(u,v))
w(V) ← w(u,v)
π(V) = u
log V
V log V + V + E log V
O(V + E log V)
Fibonacci heaps
Binomial heaps
∵ min heaps is used
aeb, cd, gf, h
strongly connected components
MST
1. Sort edges in increasing order of edge weight.
2. Take edges in increasing order

Dijkstra

same as Prims, just add prev value.

Floyd Warshall

Bellman Ford

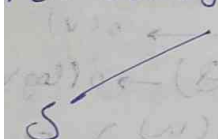
- Dijkstra will stuck at cycle and negative weights

```
for all v ∈ G
    for all e(u,v) ∈ G
        if (w(v) > w(u,v) + w(u))
        {
            w(u) = w(u,v) + w(u)
        }
}
for all e ∈ G
{
    if (w(v) > w(u,v) + w(u))
```

Relaxation of vertex

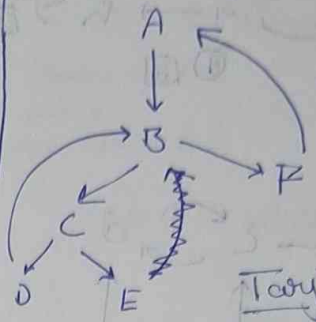
⟹ Negative weight cycle exists

T.C. = O(VE log V)



17th Feb 2025

BSI. Connected Component



Tarjan's Algorithm

Biconn' Graph: Graph which doesn't have any cut vertex

Cut Vertex: If that vertex is removed then graph becomes disconnected

Conditions to be a cut vertex

1. ~~sho~~ will have 2 edges
2. if it is root and have 2 edges then cut vertex
3. Descendents shouldn't be connected to ancestors

	dis(v)	low(v)
A	1	1
B	2	~~2~~ 1
C	3	~~3~~ 2
D	4	~~4~~ 2
E	5	5
F	6	~~6~~ 1

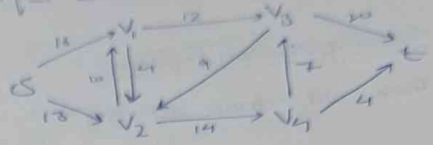
Applied DFS to find discovery time and low value and find cut ~~disco~~ vertex

```
DFS(G)
{
    time = 0
    for all V ∈ G
    {
        if color(v) = white
        {
            π(v) = NULL;
            DSF V(v)
        }
    }
    for all V ∈ G
        if(AP(v) = true)
            retrival;
        else
            return 0;
}
```

Articulation Point

```
DFSU(u)
{
    child = 0
    d(u) = low(u) = ++time
    for all V ∈ adj(u)
    {
        if (color(v) == white)
        {
            child++;
            DFSU(v)
            low(u) = min(low(u), low(v));
        }
        if(π(v) != NULL && low(v) > dis(u))
        {
            AP[u] = true;
        }
    }
    else
    {
        low(u) = min(low(u), dis(v))
    }
}
if(child >= 2 && π[u] = -1){
    AP(u) = true;
}
```

Max Flow



Flow conservation

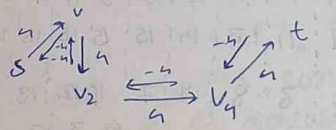
$\sum$ flow (vertex) = 0

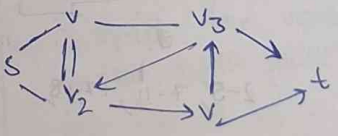
Capacity constraint

$f(u,v) \leq c(u,v)$

Ford Fulkerson Algorithm

1. for all edges $(u,v) \in G$
2. $f(u,v) = 0$
3. $f(v,u) = 0$
4. while there exist a path P from S to t
5. $c_p(m) = \min[\text{all } (u,v) \in P]$
6. for all $(u,v) \in P$
7. $f(u,v) += c_p(m)$
8. $f(v,u) = -f(u,v)$





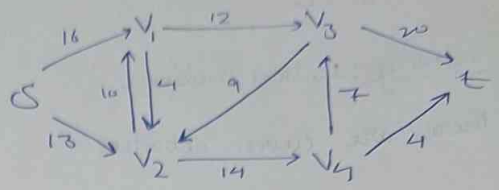
```
DFS(G)
{
    time = 0
    for all V ∈ G
    {
        if color(v) = white
        {
            π(v) = NULL;
            DSFV(v)
        }
    }
    for all V ∈ G
        if(AP(v) = True)
            return;
        else
            return 0;
}
```

Articulation Point

```
DFSU(u)
{
    child = 0
    d(u) = low(u) = ++time
    for all v ∈ adj(u)
    {
        if (color(v) == white)
        {
            child++;
            DFSU(v)
            low(u) = min(low(u), low(v));
            if(π(u) != NULL && low(v) > dis(u))
            {
                AP[u] = true;
            }
        }
        else
        {
            low(u) = min(low(u), dis(v))
        }
    }
    if(child >= 2 && π[u] = -1){
        AP(u) = true;
    }
```

Max Flow

18th Feb 2025



Flow conservation

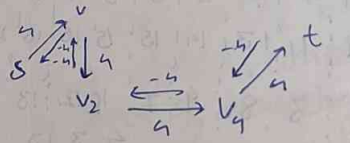
Σ flow (vertex) = 0

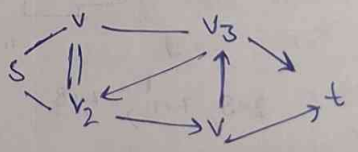
Capacity constraint

$f(u,v) \leq c(u,v)$

Ford Fulkerson Algorithm

1. for all edges $(u,v) \in G$
2. $f(u,v) = 0$
3. $f(v,u) = 0$
4. while there exist a path P from S to t
5. $c_p(m) = \min[\text{all } (u,v) \in P]$
6. for all $(u,v) \in P$
7. $f(u,v) += c_p(m)$
8. $f(v,u) = -f(u,v)$





GREEDY ALGORITHM

19th Feb 2025

Activity Selection Problem

There are eleven activities

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	0	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

$L = a_1$ $f_L = a$

for all activities $a_i \in A$

{

if ($s_{a_i} > f_L$)

{

$f_L \cdot U = a_i \Rightarrow (f_L = f_L \overset{\text{union}}{\cup} a_i)$

$L = a_i$

}

}

$f_L = 1, 4, 8, 11$

There are 13 activities

i	1	2	3	4	5	6	7	8	9	10	11	12	13
s_i	1	4	7	2	3	4	5	7	7	3	16	17	14
f_i	8	6	11	5	8	9	15	13	14	15	20	18	18

s_i	2	4	1	3	4	7	7	7	5	3	17	14	16
f_i	5	6	8	8	9	11	13	14	15	15	18	18	20
i	4	2	1	5	6	3	8	9	7	10	12	13	11

4, 3, 13

2-5, 7-11, 17-18

4, 3, 12

→ sorting algo.

T.C.

$O(n \log n)$

Knap Sack Problem

BFS (G, s)

```
1.  for each vertex u ∈ G.V - {s}
2.      u.color = white
3.      u.d = ∞
4.      u.π = NULL
5.  s.color = grey
6.  s.d = 0
7.  s.π = NULL
8.  Q = ∅
9.  Enqueue (Q, s)
10. while Q ≠ ∅
11.     u = dequeue (Q)
12.     for each v ∈ G.Adj[u]
13.         if color == white
14.             v.color = grey
15.             v.d = u.d + 1
16.             v.π = u
17.             Enqueue (Q, v)
18.     u.color = black
```

O(V+E)

TOPOLOGICAL-SORT (G)

1. Call DFS(G) to compute finishing time v.f for each vertex v
2. as each vertex is finished, insert it onto the front of a linked list
3. return the linked list of vertices.

DFS (G)

```
1.  for each vertex u ∈ G.V
2.      u.color = white
3.      u.π = NULL
4.  time = 0
5.  for each vertex u ∈ G.V
6.      if u.color == white
7.          DFS-VISIT (G, u)
```

DFS-VISIT (G, u)

```
1.  time = time + 1
2.  u.d = time
3.  u.color = grey
4.  for each v ∈ G.Adj[u]
5.      if v.color == white
6.          v.π = u
7.          DFS-VISIT (G, u)
8.  u.color = Black
9.  time = time + 1
10. u.f = time
```

O(V+E)

STRONGLY-CONNECTED-COMPONENTS(G)

1. call DFS(G) to compute finishing time u.f for each vertex u
2. compute G^T
3. call DFS(G^T), but in the main loop of DFS, consider the vertices in order of decreasing u.f
4. Output the vertices of each tree in the depth first forest formed in line 3 as a seperate strongly connected component

24th Feb 2025

Huffman Code

(Greedy, Heaps)

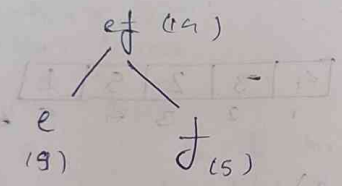
- fixed length string a, b, x, y, t

Q- a-f alphabets

a	b	c	d	e	f
45	13	12	16	9	5

1. Arrange / Sort acc. to freq
2. Extract smallest one's and make them left and right child

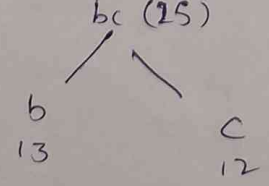
a	d	b	c	e	f
45	16	13	12	9	5



3. now replace fe (14) in the sequence

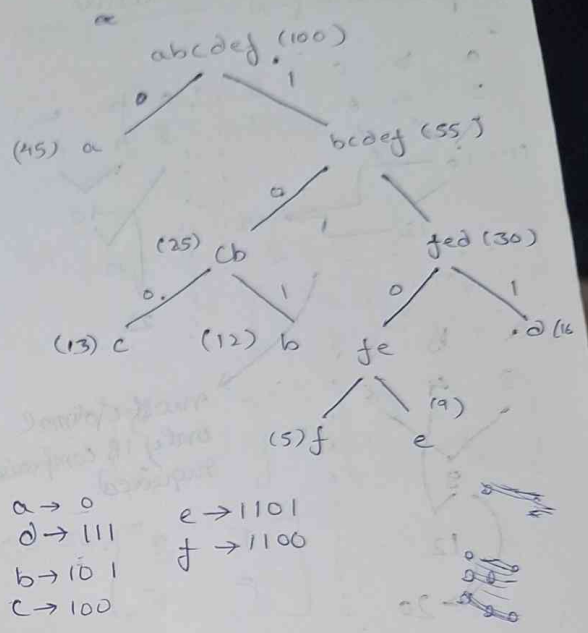
a	d	ef	b	c
45	16	14	13	12

4. Now again extract last two



5 so on

(first minimum left child
second " right child)



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

Q- 9 | 101 | 100
a b c
f

45 + 3(16+13+12) + 4(14)

6

$\frac{3 \times 100}{224}$

74.6 %

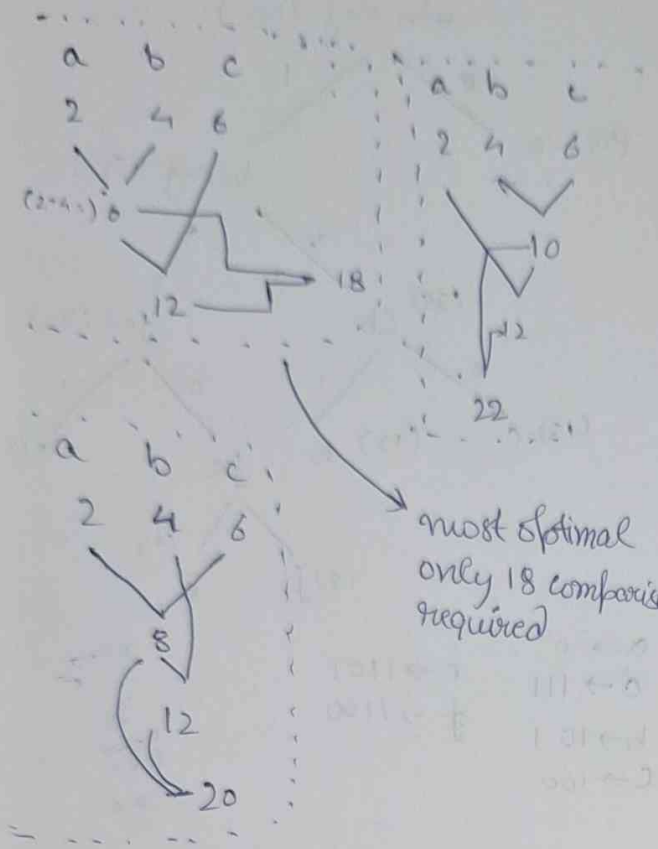
$\frac{224}{300}$

$\frac{76}{300}$ → 25.3 %

$\frac{27}{}$

O(n log n)
↳ sorting

Optimised Merge Pattern



most optimal
only 18 comparisons required

a, b, c are different file systems have files in ~~order~~ sorted order

25th Feb 2025

Huffman (c)

1. Q ← c
2. for (i=1 → n-1)
3. new node x
4. t = x → leftchild = Extractmin (Q)
5. p = x → rightchild = Extractmin(Q)
6. f(x) = f(t) + f(p) // frequency
7. enqueue(Q, x)
8. return Extractmin (a)

a b c d e f

OMP (A)

1. Q ← A , sum = 0
2. for(i=1 → n-1)
3. t = Extractmin (Q)
4. p = Extractmin (Q)
5. f(x) = f(t) + f(p)
6. sum+ = x
7. Enqueue (Q, x)
8. return Extractmin (Q)

Job Sequencing with deadline

Jobs	1	2	3	4	5	6
Deadline	5	3	3	2	4	2

Task: Schedule max^m no of jobs

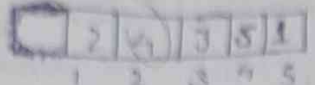
JS	4	3	2	5	1
	1	2	3	4	5 ← Deadline

JS(n, A)

1. Sort A in desc order
   Make an array JS of max^m deadline
2. for (i=1 → n)
3. ~~increase~~
3. insert job at JS[d_i]
   If free otherwise find a free slot in rev. order
4. exit

Jobs	1	2	3	4	5	6
Deadline	5	3	3	2	4	2
Profit	200	180	190	300	120	100

This time will we to profit



MATRIX MULTIPLICATION 3rd MARCH 2025

$A \times B = C$

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

$c_{11} = a_{11} \times b_{11} + a_{12} \times b_{21}$
$c_{12} = a_{11} \times b_{12} + a_{12} \times b_{22}$
$c_{21} = a_{21} \times b_{11} + a_{22} \times b_{21}$
$c_{22} = a_{21} \times b_{12} + a_{22} \times b_{22}$

$$\begin{bmatrix} 1 & 1 & 2 & 1 \\ 2 & 1 & 2 & 2 \\ 1 & 2 & 1 & 2 \\ 2 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 & 2 \\ 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

A: a_{11} a_{12} a_{21} a_{22} — 4×4 — B: b_{11} b_{12} b_{21} b_{22} — 4×4

$$\begin{bmatrix} 3 & 4 & 4 & 5 \\ 4 & 6 & 6 & 6 \\ 3 & 3 & 3 & 4 \\ 3 & 3 & 3 & 4 \end{bmatrix} \times$$

C

$$c_{11} = \begin{bmatrix} 3 & 4 \\ 4 & 6 \end{bmatrix} + \begin{bmatrix} 3 & 3 \\ 4 & 4 \end{bmatrix}$$

$$c_{11} = \begin{bmatrix} 6 & 7 \\ 8 & 10 \end{bmatrix}$$

$$c_{12} = \begin{bmatrix} 3 & 5 \\ 4 & 5 \end{bmatrix} + \begin{bmatrix} 3 & 3 \\ 4 & 6 \end{bmatrix}$$

$$c_{12} = \begin{bmatrix} 6 & 8 \\ 8 & 11 \end{bmatrix}$$

$$c_{21} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 6 \\ 4 & 6 \end{bmatrix} + \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}$$

$$c_{21} = \begin{bmatrix} 8 & 9 \\ 7 & 9 \end{bmatrix}$$

$$c_{22} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix} + \begin{bmatrix} 3 & 4 \\ 3 & 4 \end{bmatrix}$$

$$c_{22} = \begin{bmatrix} 8 & 8 \\ 7 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 6 & 7 & 6 & 8 \\ 8 & 10 & 8 & 11 \\ 8 & 9 & 8 & 8 \\ 7 & 9 & 7 & 9 \end{bmatrix}$$

```
MM(A, B, n)
{
  if (n==2)
  {
    c11 = a11*b11 + a12*b21
    c12 = ...
    ...
  }
  else
  {
    c11 = MM(A11, B11, n/2) +
          MM(A12, B21, n/2)
    ...
    c22 = ...
  }
}
```

$T(n) = 8T(n/2) + n^2$

By master's Theorem

$T.C = O(n^3) = O(n^{\log_2 8})$

Strassen ~~did~~ reduced multiplications from 8 to 7

$\therefore T.C = O(n^{\log_2 7})$

$P = (A_{11} + A_{22}) * (B_{11} + B_{22})$

$Q = (A_{21} + A_{22}) * B_{11}$

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

$S = A_{22} * (B_{21} - B_{11})$

$T = (A_{11} + A_{12}) * B_{22}$

$U = (A_{21} - A_{11}) * (B_{11} + B_{12})$

$V = (A_{12} - A_{22}) * (B_{21} + B_{22})$

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

$C_{12} = R + T$

$C_{21} = Q + S$

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

7 Multiplications

18 Addition / Subtraction

Closest Pair Problem

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