ECES 215 | M proposes W, in turn W accepts or trades

Proof of Correctness — By contradiction

① Terminates
Each time m proposes to new 'w', only n^2 proposal

② Perfect matching
Zeus not matched upon termination, then some women not matched.
Amy was never proposed to, But Zeus proposed to every one.

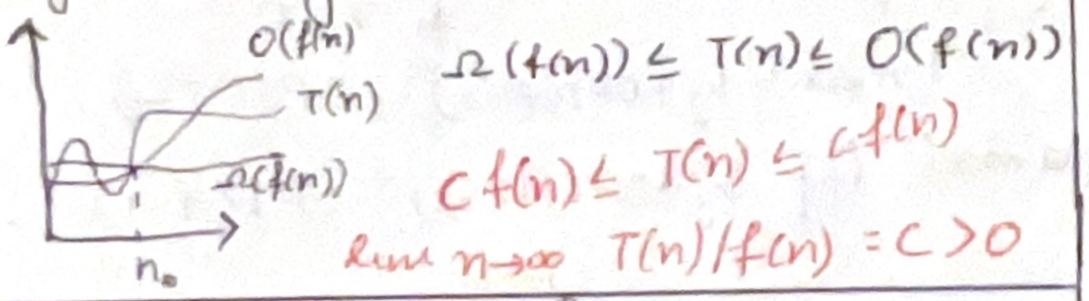
③ Stability
case1: Z never proposed to A } stable
Z prefers QS over A
case2: Z proposes, A rejects } stable
A prefers QS over Z

- Perfect stable matching always exists for stable marriage, but not stable roommate
- GS - finds 'A' stable match (atmost n^2)
  - Unique and man optimal.

- Algo is efficient if runtime cN^d
- No algorithm is strategy proof.

Asymptotic growth.

$\log_2 n < n < n\log n < n^2 < n^3 < 1.5^n < 2^n < n!$



$\Omega(f(n)) \le T(n) \le O(f(n))$

$cf(n) \le T(n) \le cf(n)$

$\lim_{n\to\infty} T(n)/f(n) = c > 0$

Matrix — list
+ or − → (O(1)) and ← space → O(|V|+|E|)
finding adjacent → O(1) → O(deg(u))

BFS ← NOT unique → DFS

BFS:
- Queue
- layer by layer
- FIFO
- searching for vertices close to the source
- O(V+E) - list, $O(n^2)$ - Matrix.
- no backtracking

DFS:
- Stack
- subtree by subtree
- LIFO
- sol away from source
- O(V+E), $O(n^2)$

if (x,y) edge exist in graph and tree distance b/w x & y = 1

Bipartite (2 colour) edge → 1 blue & 1 Red.
- Can not have odd length cycle.
- Cant have 2 nodes connected in the same layer.

N & E = A ∪ B | Sum of degree in vertices in U = Sum of degree in Vertices V

Unconnected G

Strongly connected - if s is reachable from every node & vice versa.

DAG - Directed Acyclic graph.

Topological order - Directed graph, such that all nodes point forward.
- find node with no incoming edges.

(G → DAG → TO) Proof by Induction
① Prove its true for 1 → Base case
② Prove for n ③ Prove for n+1
not unique | O(m+n)

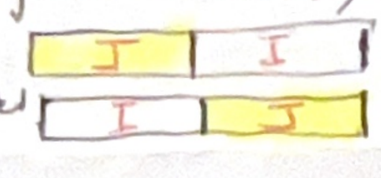
If G → TO → DAG

Greedy algorithm → optimal
- locally optimises at every step for a global solution

How to prove it is optimal:
- greedy stays ahead • exchange argu
- structural bound

Interval scheduling - finish time nlogn
Interval partitioning - Start time (Inf)
Inversion - greedy has no inversion

Such a pair, $i<j$ & $d_i \le d_j$ (due time), but j scheduled earlier.
- Swapping reduces inver



$l'_k = l_k$ for All $k \neq i,j$

$l'_i \le l_i$

$l'_j = f'_j - d_j = f_i - d_j$ (j finishes at f_i)

$l'_j < f_i - d_i$

$\boxed{l'_j \le l_i}$

Shortest path.
Relaxation if $(d[u]+c(u,v) < d[v])$
then $d[v] = d[u] + c(u,v)$
Running time = $O(n^2)$ → PQ → O(mlogn)
- Neg length don't work.

MST → ST → Subgraph of a graph.
No of ST → n^{n-2}

Prims — $O(n)^2$ → BH → O(mlogn)
- select min cost edge
- same step with connected vertices.

Kruskals — $O(n^2)$ → Min heap → O(mlogn)
- select min cost edge.
- should not form cycle

Reverse delete
- delete largest edge
- check G is still connected
- Repeat until edge = vertices − 1

Connected G — Can not be planar — NSC

Cutset property

S → subset of nodes, & e be min cost edge with exactly 1 endpoint in S. The MST must contain e.

cycle & cutset intersect at even no of edges.

cycle property

C-cycle & f be the max cost edge belonging to C. Then the MST does not contain f.

---

divide & conquer

- Recursive
- prb & subprb → same type

- Merge sort.

→ Divide into 2, recursively solve & Merge

T(n) = no of comparisions to merge.

$$T(n) = \begin{cases} 0 & \text{if } n=1 \\ 2T(n/2)+n & \text{otherwise} \end{cases}$$

- Running time $O(n \log n) \Rightarrow O(n)$ - merge, $O(\log n)$ - no of level

Proof n power of 2.

for $n>1$ $\frac{T(n)}{n} = \frac{2T(n/2)}{n} + 1$

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

$= \frac{2T(n/4)}{n/4} + 1 + 1$

$= \frac{T(n/n)}{n/n} + 1 \dots + 1$

$= \log_2 n$

$T(n) = n \log n$

MT: $aT(n/b) + O(n^d)$

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

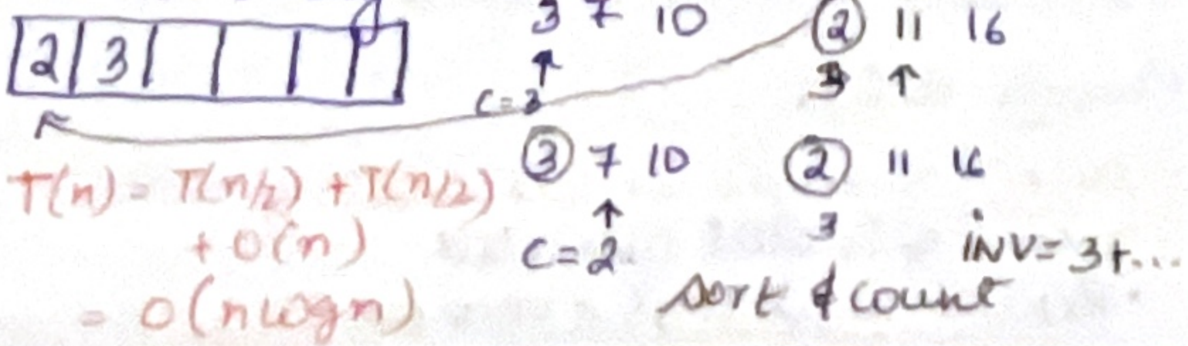
a - no of subprb
b - size of subprb
c - cost to do the merge.

$a>0, b>1, d \geq 0$

INVERSION i & j are inverted if $i<j$, but $a[i] > a[j]$

BF = $O(n^2)$

- divide & conquer
- count inversion in 2 divide array
- Merge count.
- sort array.



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

$= O(n \log n)$

Integer Multiplication

- 2 n-digit integer
- Multiply 4 1/2 n digit integer
- add 2 1/2 n & shift to get result.

$X = 2^{n/2} \cdot x_1 + x_0$

$Y = 2^{n/2} \cdot y_1 + y_2$

$XY = 2^n x_1 y_1 + 2^{n/2}(x_1 y_0 + x_0 y_1) + x_0 y_0$

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

MT $d < \log_b a \quad T(n) = O(n^2)$

Karatsuba → $T(n) = 3T(n/2) + cn$

MT $d < \log_b a \Rightarrow O(n^{1.585})$

Matrix → A & B into 1/2 n 1/2 n

$T(n) \leq 8T(n/2) + \Theta(n^2)$

Coppersmith → $T(n) = 7T(n/2) + \Theta(n^2)$.

$= O(n^{\log_2 7}) = O(n^{2.81})$ COPPER → $O(n^{2.372})$

Dynamic Programming

- Series of overlapping subprb
- subprb reused several times.

Weighted Interval

Job → $J = \{s_j, f_j, v_j\}$ 2 jobs compatible they dont overlap

P(J) = largest index $i<j$ such that previous Job i is compatible with J.

Bellmans equation

$$OPT(J) = \begin{cases} 0 & \text{if } J=0 \\ \max\{v_j + OPT(P(J)), OPT(J-1)\} \end{cases}$$

J optimal iff $v_j + OPT(P(J)) > OPT(J-1)$

- Has redundant subprb.
- 2 Ways to solve (a) Memoization top down (global array) → O(n)
  (b) unwind recursion bottom-up.

IS - Greedy $g_1 \dots g_k \quad R_1 \dots R_k$

BC: g_1 choosen by greedy $f(g_1) \leq f(R_1)$

IH $f(g+1) \leq f(R+1)$, NOW suppose $R_j = (S_j, F_j)$

f_j is smallest $f(g_i) < f(R_i) < S(R_{i+1}) \leq f(R_{i+1})$

IH, compatible

RT

1. BFS, DFS → $O(m+n)$ ← DAG
2. IS - $O(n \log n)$ → Dijkstra, prims, kruskals, merge sort WIS