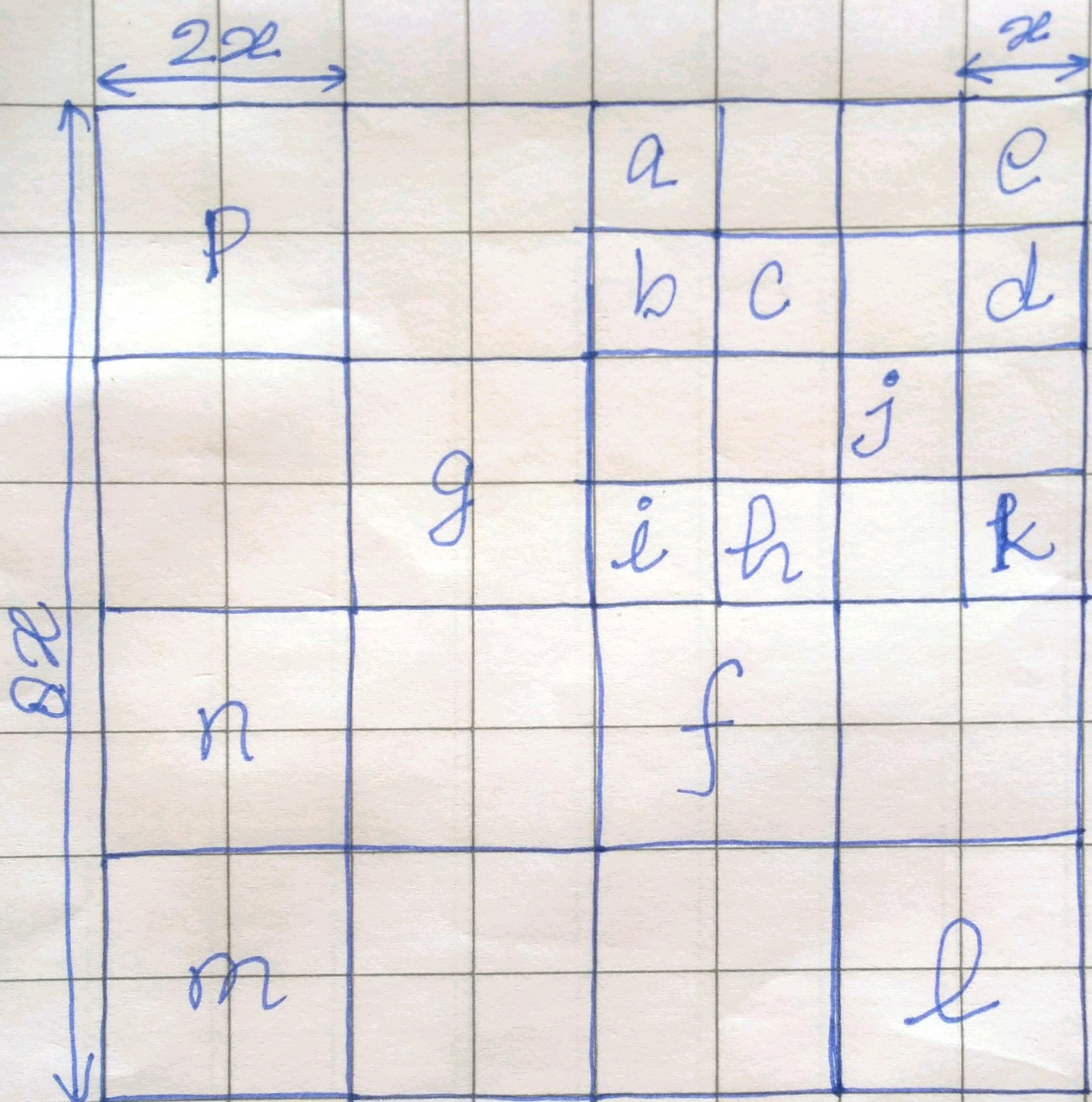
$\boxed{\text{P-1}}$



let x = unit length as depicted

let the definition for "far enough"

distance(s, z_i) $\geq$ side length of s

where s = visited square z_i = evaluation point

Now using above condition if charge is far enough we use ~~direct calcu~~ multipole expansions otherwise direct calculation for potential. (Algo. in lec12 page13)

$\phi(p)$: Multipole expansions of the squares containing each set of the following charges: {a,b,c}, {e,d}, {i,h}, {j,k}, {f}, {l}, {n}, {m}

and Direct calculation of the potential due to ~~each of the~~ following charges: g

$\phi(g)$: Multipole expansion of the squares containing each set of the following charges: {e,d}, {j,k} {l}, {m}, ~~and~~ {h}, {a}, {c}

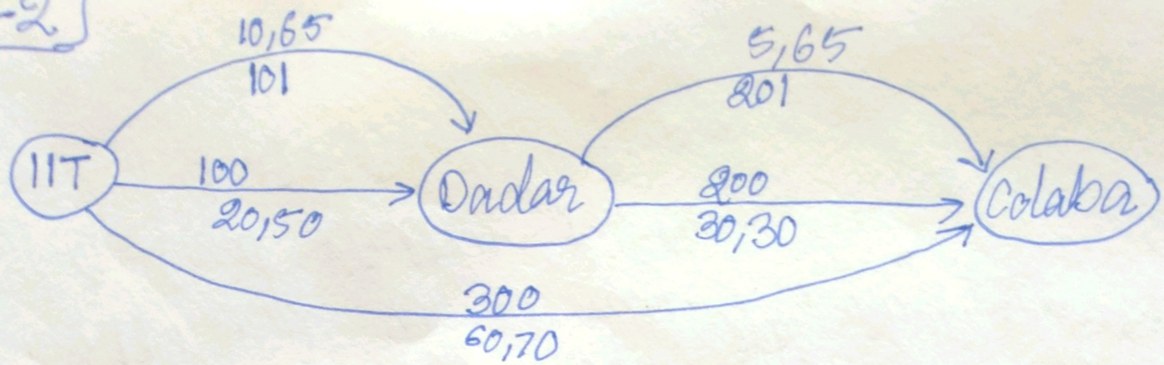
Direct calculation of the potential due to each of the following charges: p, f, n, i, b

$\phi(b)$: Multipole expansion of the square containing each set of the following charges: {p}, {n}, {m}, {f} {l}, {d}, {e}, {j}, {k}, {i}, {h}

direct calculation of the potential due to each of the following charges: a, b, c, g

$\phi(\vec{r})$: Multipole expansion of the square containing each set of the following charges: $\{p\}, \{g\}$ $\{f\}, \{l\}, \{a, b, c\}, \{i, h\}, \{j, k\}, \{e, d\}$ and direct calculation of the potential due to: n charge n.

P-2



lets above schematic diagram of routes and time (min)

lets first solve for dadar to colaba Route 200 VS Route 201

since 65 (running time R201) > 30 + 30 (R200 : running time + waiting time)

Hence there is no point choosing Route 201 if we are at dadar.

For Route $_{IIT \to dadar}$ (Use ~~Exptd~~ Expected optional algorithm)

$$\text{Expt. waiting time} = \frac{1}{A} = \frac{1}{\frac{1}{20}+\frac{1}{10}} = \frac{20}{3} \text{ min}$$

$$\text{Expt. Running time} = \frac{\frac{1}{20}\times 50 + \frac{1}{10}\times 65}{\frac{1}{20}+\frac{1}{10}} = \frac{180}{3} = 60 \text{ min}$$

Now Expt. waiting time $_{IIT \to colaba}$ (via dadar)

$$= \frac{20}{3} + 30 = 110/3 \text{ min}$$

Expt. Running time $_{IIT \to colaba}$ (via dadar)

$$= 60 + 30 = 90 \text{ min}$$

$70 \text{ min} < 90 \text{ min} < (70+60) \text{ min}$

Hence

$$\begin{tikzcd} IIT \arrow[r, bend left, "\frac{110}{3}, 90"] \arrow[r, bend right, "60, 70"'] & Colaba \end{tikzcd}$$

Total Expted Waiting Time IIT → Colaba

$$\frac{1}{\frac{1}{\frac{110}{3}} + \frac{1}{60}} = \frac{(110)\cdot 60}{3\times 60+110} = \frac{660\cancel{0}}{29\cancel{0}} = \frac{660}{29} \text{ min}$$

Expt. Total Running Time IIT → colaba

$$= \frac{70\times\frac{1}{60} + 90\times\frac{3}{110}}{\frac{1}{60} + \frac{3}{110}} = \frac{\frac{70\times 110 + 90\times 60\times 3}{\cancel{(60\times 110)}}}{\frac{(290)}{\cancel{(60\times 110)}}}$$

$$= \frac{770+1620}{29} = \frac{2390}{29} \text{ min}$$

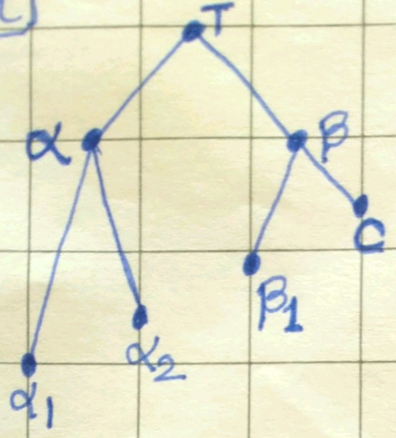
Hence IIT → colaba

Expted Waiting Time = 22.76 min | Total Expt time = 105.2 min

Expt. Running Time = 82.41 min

where we choose one which come first at IIT out of Route (100, 101, 300). and if we reached dadar then simply choose Route 200.

P-3a



using optional Binary Trie Lemma:
if codewords have length more than 1 the left and right tree of root tree have their child tree i.e. atleast 4 leaves in tree.

also we know that code word with least frequency are merged first and codeword with highest frequency are merged last, and eventual biggest frequencies are directly connected to root.
Hence to avoid merging of codeword "c" directly to tree T assume c is as least as possible. Hence we ~~could~~ conclude c with frequency $f(c) > 2/5$ is ~~might~~ have highest frequency.
according to merge in tree (all are assumption)

$f(\alpha_1), f(\alpha_2) \leq f(\beta_1), f(c)$ — ① ; $f(c) > 2/5$ — ⓪

$\Rightarrow f(\alpha) + f(\beta_1) < 3/5$ — ②

also $f(\alpha) > f(\beta_1), f(c) > 2/5$ — ③

using ② and ③ $f(\beta_1) < 1/5$

$f(\alpha) = f(\alpha_1) + f(\alpha_2)$ if $f(\alpha_1)$ increase $f(\alpha_2)$ decreases

hence to keep minimum $(\max(f(\alpha_1), f(\alpha_2)))$

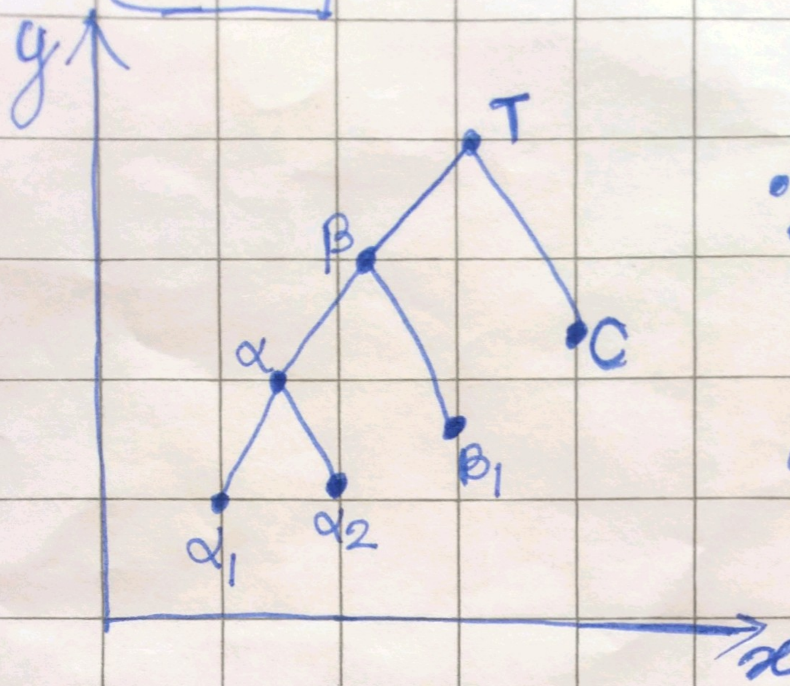
$f(\alpha_1) = f(\alpha_2) = \frac{f(\alpha)}{2} > \frac{(2/5)}{5} > 1/5 > f(\beta_1)$ — ④

contradiction ①

according to assumption $f(\beta_1) > f(\alpha_1)$ which contradict conclusion equation ④ hence our assumption is wrong and

Tree will have atleast one codeword with length = 1.

P-3b



∵ f(each node) < $\frac{1}{3}$ there would be atleast 4 ~~a~~ leaves in tree

using optimal Binary tree lemmas ~~lets~~ above figure ~~depicted~~ represent tree have some codeword (C) with length = 1

Similar to part(a)
we know that its better to have higher frequency as possible to merge ~~it~~ some codeword at last directly to root ~~tree~~.
Hence $f(C) = \frac{1}{3}^{-}$ just lessthan $1/3$ —— ⓪
Also we have according to merging preference.

$$f(\alpha_1), f(\alpha_2) \leq f(\beta_1), f(\alpha) \leq f(C), f(\beta) \text{ —— } ①$$

But $f(\beta) = \frac{2}{3}^{+}$ $\{\because f(C) < \frac{1}{3} \Rightarrow f(\beta) > 2/3\}$

$\Rightarrow f(\beta) > 2/3$ —— ②

to minimize ; $\max(f(\alpha), f(\beta_1))$

$\Rightarrow f(\alpha) = f(\beta_1) = \frac{1^{+}}{3} = \frac{f(\beta)}{2} > \frac{1}{3} > f(C) \longrightarrow ③$

contradicted ①

according to figure as per merging preference
$f(\beta_1), f(\alpha) < f(C)$ But we concluded ③

$f(\alpha) = \frac{1}{3} > f(C)$ which contradicts our assumptions. Hence there is no codewords having length 1

[P4] let graph G is given as adjacency ~~matrix~~ list.

$G = \{ V_1(P_{11}, P_{12}), V_2(P_{21}, P_{22}) --- V_i(P_{i1}, P_{i2}) --- V_n(P_{n1}, P_{n2}) \}$

where $V_i \to$ vertex, $P_{ij} \to$ node. also P_{ij} may be equal to P_{xy} for some points i.e. they are not vertices as per notation.

$H \to$ stores visited nodes $\to$ a kind of hash table.

$A \to$ stores list of edges on vertices having maximum matching.

Nodel $\to$ stores all node of graph G given
ie $\{ P_{11}, P_{12}, P_{21}, P_{22} --- P_{i1} P_{i2} — P_{n1} P_{n2} \}$

Now find all points in Nodel having least frequency minimum occurrence (they occurs at ~~the~~ maximum depth of tree) and store them in list leastF $\{P_0 -- P_x\}$

Repeat (till leastF $\{\}$ is not empty) {
take first point from leastF ~~P_0~~ find corresponding Vertex $V_i(P_0 — P_y)$ ~~store V_i to~~ from graph G
store V_i to "A", remove V_i from G
remove P_0 from leastF
remove P_0, P_y from Nodel() }

repeat until Nodel is not empty.

---

worst case above algo. runs in $n \times n = O(n^2)$

P-5 given D[1...n] P[1...n]

~~slot~~A (max D[1....n]) → array of slots allotments of jobs which is initiased with zero. ie. slotA[i] = 0 for each i, also ~~cat~~ size of slotA could be infinite But lets keep it max(D(1--n)) which is maximum number of slots required. lets say max(D[1---n]) = s

MPG (D[1...n], P[1,---n], slotA[1....s])
// Return slotA after updating, which is required
// for maximum profit using greedy algo.

⇒ Now lets P[j] = max P[1...n] for some
P[j] ∈ P[1...n]

⇒ Now find maximum K s.t. K ≤ D[j] and
slot A[K] ~~=~~ is zero

⇒ if such K doesn't exist leave the job with P[j] profit and D[j] ~~the~~ deadline for last, left schedule.

⇒ And if K exist then update slot A[K] = D[j]
also update $D' = D - D[j]$, $P' = P - P[j]$ } remove scheduled job.

Now call

MPG (D'[1....n-1], P'(1...n-1), slotA(1.....S)]

return,

After that if we want to perform job left for last schedule just append them in any order to slotA[] → array with not filled index i.e if slotA[k]=0.

Since new instance is ~~same~~ similar as original for D' P' except max profit already scheduled at max possible deadline to make profit; eventually it derives optimal solution.

Time will be roughly

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

obviously which is polynomial in n