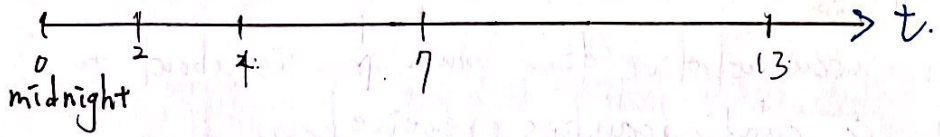
Problem 5.

B05202068 劉耿宇.

(1) c_i: 8 5 1 9



$C_{tot} = c_1 + c_2 + c_3 + c_4 = 23$.

(2) The latest visit time for P_i (i-th professor) is $r_i - t_i$

We can start from $\min_{1 \le i \le N} t_i = t_1$ (assumption ⇒ ∵ assumption 1).

Since we have to visit as much professor as possible w/o breaking the rule

let T_i be the accumulative time when one is about to visit P_i

$$T_i = \begin{cases} T_{i-1}, & T_{i-1} > r_i - t_i \ (\text{NOT visit}) \\ t_i + T_{i-1}, & T_{i-1} \le r_i - t_i \ (\text{the previous end time} \le \text{the latest visit time of } P_i) \\ t_1, & i = 1 \end{cases}$$

So whenever we visit P_i, the accumulative candy received + 1

$$\therefore C_i = \begin{cases} C_{i-1}, & T_{i-1} > r_i - t_i \\ 1 + C_{i-1}, & T_{i-1} \le r_i - t_i \\ c_1 = 1, & i = 1 \end{cases}, \quad i = 1 \sim N. \Rightarrow \text{runs in } O(N).$$

(3) Still need to visit as much professors as possible.

We can sort t_i first → use merge sort in $O(N \log N)$

and start from $\min_{1 \le i \le N} t_i$ as in (2), then do the same thing in (2).

$\{t_i\} \xrightarrow{N \log N} \{t_j\}$ sorted.

$$T_j = \begin{cases} T_{j-1}, & T_{j-1} > r_j - t_j \\ t_j + T_{j-1}, & T_{j-1} \le r_j - t_j \\ t_1, & j = 1 \end{cases}$$

* In (2) and (3), we also need to check if $r_i \ge t_i$ before each visit

(4) With the constraint $\sum_{i=1}^{N} c_i < M$.

let $T_{i,c}$ be the accumulative time when one run to P_i with total candies received less than c.

$$T_{i,c} = \begin{cases} T_{i-1,c} & \text{if } T_{i-1,c-c_i} > r_i - t_i \text{ or } c \le c_i \\ t_i + T_{i-1,c-c_i}, & \text{if } T_{i-1,c-c_i} \le r_i - t_i \text{ and } c > c_i \\ \begin{cases} t_1, & i=1,\ c>c_i \\ 0, & i=1,\ c \le c_i \end{cases} \end{cases}$$

$i = 1 \sim N$, $c = 0 \sim M$. → find if $T_{N,M}$ has value and backtracing until $T_{i,c} = 0$. (during backtracing, pick all c_i received)

the table can be done in $N \times M$ steps ⇒ $O(NM)$ ✻✻

eg. $t_i = [2,2,3,6,10]$, $r_i = [3,4,15,16,10]$, $N=5$, $M=18$.

c_i	i \ M	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
8	1	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2
5	2	0	0	0	0	0	0	2	2	2	2	2	2	2	2	4	4	4	4	4
1	3	0	0	3	3	3	3	3	5	5	5	5	5	5	5	5	7	7	7	7
9	4	0	0	0	0	0	0	0	0	0	0	6	9	9	9	9	9	11	11	11
8	5	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	10	11

$T_{5,10} = t_5 + T_{4,10-8}$ $\quad T_{5,18} = T_{4,18}$

So when $M=18$, the maximum candies one can get = $T_{4,18-8} < r_5 - t_5$ = 11 = 6.

$T_{5,18} \rightarrow T_{4,18} = 12 \rightarrow T_{3,18-9} = 5 \rightarrow T_{2,9-1} = 2$

$c_t = 9$

$c_t = 1$

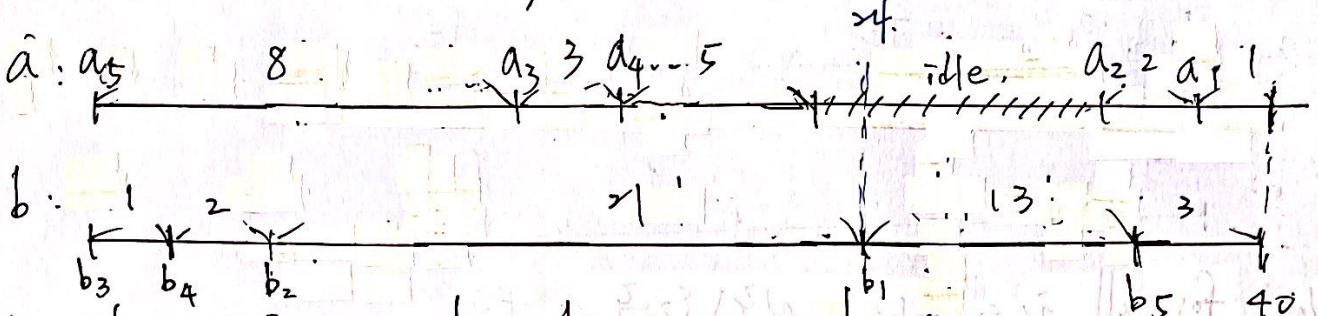
$T_{1,8-5} = 0$

$c_t = 5$

$\therefore C = 9+1+5 = 15 < M$. ✻✻

0 t_2 t_3+t_2 $t_4+t_3+t_2 = 11$

(5) when $x=5$, $\min(f_5, g_5) = \max_{1\le i\le N=5}(\min(f_i, g_i)) = \max_{1\le i\le 5}(1,2,1,2,3)$



∴ The minimum ending time is at 40. ※

$a = \{9, 7, 8, 11, 0\}$, $b = \{4, 3, 0, 1, 7\}$

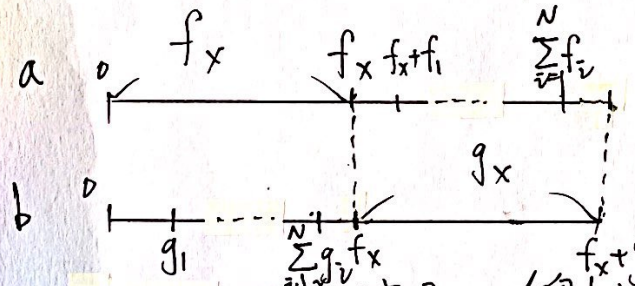
(6) if $\exists x$ s.t. $(f_x + g_x) > \max(\sum_{i=1}^{N} f_i, \sum_{i=1}^{N} g_i)$,

pf: then the minimal ending time is $f_x + g_x$.

With loss of generality, suppose $\sum_{i=1}^{N} f_i \ge \sum_{i=1}^{N} g_i$,

we have $f_x + g_x > \sum_{i=1}^{N} f_i \ge \sum_{i=1}^{N} g_i \Rightarrow g_x > \sum_{i\ne x}^{N} f_i$ and $f_x > \sum_{i\ne x}^{N} g_i$.

(Note that if we change to $\sum_{i=1}^{N} g_i \ge \sum_{i=1}^{N} f_i$, the result won't change.)



∴ the ending time takes at least $f_x + g_x$

(7) Discuss with 廖晨皓/劉安浚.

In order to mismatch each visit of professor i for a_i, b_i, → Consider a specific prof. x s.t.

$\min(f_x, g_x) = \max_{1\le i\le N}(\min(f_i, g_i))$. w.l.o.g. → suppose $f_x \ge g_x$

$f_x \ge g_x = \min(f_x, g_x)$.

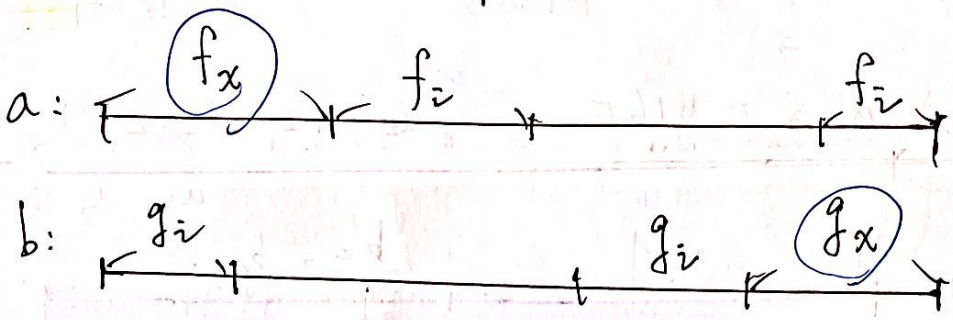
$$\begin{cases} \ge g_i, & i \in \{1,\sim,N\}\setminus x, \text{ if } f_i \ge g_i \\ \ge f_i, & i \in \{1,\sim,N\}\setminus x, \text{ if } f_i < g_i. \end{cases}$$

we have two important inequalities:

$$\begin{cases} f_x \ge g_i & \text{if } f_i \ge g_i \\ g_x \ge f_i & \text{if } f_i < g_i \end{cases}$$

$\Rightarrow$ g_x, f_x can make sure f_i, g_i won't overlap

∴ a will do x first and b will do x at the end.



Now, traverse all $i \in \{1, \sim, N\} \setminus \{x\}$

① if $f_i \geq g_i$, then $f_x \geq g_i$ → place i at the left most available spot.

② if $f_i < g_i$, then $f_i \leq g_x$ → place i at the right most available spot

thus takes $2(N-1)$ comparisons ⇒ $O(N)$ time complexity ##

Problem 6 :

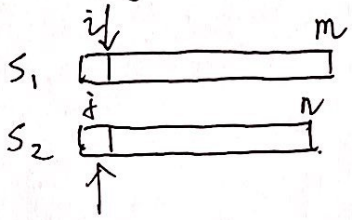
(1) $K=1$.

check: change S_1 to S_2 in K moves $\Leftrightarrow$ check if edit distance btw S_1 & S_2 is $K=1$

$\Rightarrow$ simultaneously traverse both strings (maintain two pointers i, j to S_1, S_2) and keep track of the count of different characters.

Let : $|S_1| = m$, $|S_2| = n$ $(1 \le m, n \le N)$

$k_{i,j}$: count of different chars. at (i, j)



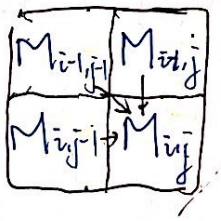
$$k_{i,j} = \begin{cases} k_{i+1,j+1}, & \text{if } S_1[i] = S_2[j] \\ 1 + k_{i+1,j}, & \text{if } S_1[i] \neq S_2[j] \text{ and } m > n \\ 1 + k_{i,j+1}, & \text{if } \quad " \quad \text{and } m < n \\ 1 + k_{i+1,j+1}, & \text{if } \quad " \quad \text{and } m = n \\ 1 + k_{i+1,n}, & \text{if } j = n,\ i < m \\ 1 + k_{m,j+1}, & \text{if } i = m,\ j < n. \end{cases}$$

The goal is to see if $k_{m,n} \le K = 1$.

$\Rightarrow$ takes $m+n$ steps $\le 2N = O(N)$.

(3) Discuss with 賴言禮.

Following the typical edit distance problem.



$$M_{i,j} = \begin{cases} j, & i = 0 \ (j \text{ additions}) \\ i, & j = 0 \ (i \text{ deletions}) \\ \min(\underbrace{M_{i-1,j}+1}_{\text{delete}}, \underbrace{M_{i,j-1}+1}_{\text{addition}}, \underbrace{M_{i-1,j-1}+C_{i,j}}_{\text{possibly replace}}), & \text{otherwise.} \end{cases}$$

$$C_{i,j} = \begin{cases} 1, & S_1[i] \neq S_2[j] \\ 0, & S_1[i] = S_2[j]. \end{cases}$$

* $M_{i,j}$ = Count of moves for changing $S_1[1 \cdots i]$ to $S_2[1 \cdots j]$

except that the DP table only records those $M_{i,j} \le K$, values greater than k are excluded. (a cut-off).

∴ For i-th string $S_1[1 \cdots i]$, the valid range would be at most $2K+1$ cells, from $(i, i-K)$, ~, (i, i), ~, $(i, i+K)$

The total cells we have to compute $\le |S_1| \times (2K+1) \le 2NK = O(NK)$.

(4) $\Sigma = \{a, b, c, d, e, f\}$, $Dis(c_1, c_2) = |ord(c_1) - ord(c_2)|$.

S_1 = "adefc", S_2 = "accfd"

S_2 \ S_1	0	a	d	e	f	c
0	0 →	0 →	3 →	4 →	5 →	8
a	0 ↓	0 ↘	3 →	4 →	5 →	8
c	2 ↓	2 ↓	3 →	4 →	5 ↘	8
c	2 ↓	2 ↓	3 →	4 →	5	8
f	5 ↓	5 ↓	5 ↓	5 ↓	5 →	8
d	7	7	7	7	7 →	8

S = "accdefdc"

$D(S) = 8$

(b) The S must be one of the shortest common supersequence (SCS) of S_1 & S_2. (∵ S_1 & S_2 are subsequence of S and $D(S)$ is minimum)

let $D_{i,j}$ be the minimum $D(S_{i,j})$ btw $S_1[1 \sim i]$, $S_2[1 \sim j]$ and $e_{i,j}$ be the last element of $S_{i,j}$.

$$D_{i,j} = \begin{cases} D_{i-1,j} + Dis(S_1[i], S_2[i-1]), & j = 0 \\ D_{i,j-1} + Dis(S_2[j], S_2[j-1]), & i = 0 \\ D_{i-1,j-1} + Dis(S_1[i], e_{i-1,j-1}), & \text{if } S_1[i] = S_2[j] \\ \min\left(D_{i-1,j} + Dis(S_1[i], e_{i-1,j}),\ D_{i,j-1} + Dis(e_{i,j-1}, S_2[j])\right), & \text{if } S_1[i] \neq S_2[j] \end{cases}$$

a variant of SCS.

$D_{i-1,j} + Dis(S_1[i], e_{i-1,j})$: try to append $S_1[i]$ to $S_{i-1,j}$.

$D_{i,j-1} + Dis(e_{i,j-1}, S_2[j])$: try to append $S_2[j]$ to $S_{i,j-1}$.

⇒ To build this DP table, we have $|S_1| \cdot |S_2|$ steps to do.

∴ $|S_1||S_2| \leq N^2 = O(N^2)$.