Kashish Jethmalani, 034302188

CECS 528, Homework Assignment-2

$\boxed{1.}$ Edit distance when substitution is disallowed and adjacent swap is allowed.

Taking $u = u_1 u_2 \dots u_m$ and $v = v_1 v_2 \dots v_n$.

$d(i,j)$ = minimum number of edits to transform $u[1 \dots i]$ into $v[1 \dots j]$

→ delete u_i → cost 1

→ insert v_j → cost 1

→ match when $u_i = v_j$ → cost 0.

→ swaping two adjacent letters $u_{i-1} u_i \rightarrow u_i u_{i-1}$ → cost 1

Here, while swapping, both pairs i.e. u_{i-1} & u_i to match v_{j-1}, v_j, these are consumed, all subsequent edits occur to the left. The recurrence enforces this by jumping from (i,j) to $(i-2, j-2)$

$\boxed{1a}$ Recurrence, $d(0,j) = j$, $j = 0, \dots, n$

$d(i,0) = i$, for $i = 0, \dots, m$

$$d(i,j) = \min \begin{cases} d(i-1,j)+1 & \text{delete } u_i \\ d(i,j-1)+1 & \text{insert } v_j \\ d(i-1,j-1) & \text{match, if } u_i = v_j \\ d(i-2,j-2)+1 & \text{swap if } i \geq 2, j \geq 2, u_{i-1} = v_j, u_i = v_{j-1} \end{cases}$$

(No term for substitution as its not allowed)

This runs in $O(mn)$ time and space.

1b $u = baababab$, $v = abbbabba$

$\downarrow u \backslash v \rightarrow$	λ	a	b	b	b	a	b	b	a
λ	0	←1	←2	←3	←4	←5	←6	←7	←8
b	↑1	2↑	↖1	↖2	↖3	←4	↖5	↖6	←7
a	2↑	↖1	⇖1	←2	←3	↖3	←4	←5	↖6
a	3↑	↖2	↑2	↑3	↑4	↖3	←4	←5	⇖5
b	↑4	3↑	↖2	↖2	↖3	⇖3	↖3	↖4	←5
a	↑5	4↖	↑3	↑3	↑4	↖3	↑4	↑5	↖4
b	↑6	5↑	↖4	↖3	↖3	⇖3	↖3	↖4	⇖4
a	↑7	↖6	↑5	↑4	↑4	↖3	↑4	↑5	↖4
b	↑8	↖7	6	⇖5	↖4	↑4	↖3	↖4	⇖4

↑ delete u_i

← = insert v_j

⇖ = swap u_{i-1}, u_i to match v_{j-1}, v_j

↖ = match $u_i = v_j$

Total cost $d(8,8) = 4$.

1. swap baababab → baababba
2. Delete 'a' at position 3: baababba → bababba
3. Insert b at position 3: bababba → babbabba
4. Swap: babbabba → abbbabba

$\boxed{2}$ $e_j = (s_j, f_j, b_j)$ sorted by increasing finish time.

$\boxed{2a}$ Let $p(j)$ be the index of the rightmost event that finishes no later than S_j (i.e. $f_{p(j)} \leq S_j$); if none exists, $p(j) = 0$

OPT(j) = Maximum total bid achievable using only events $\{e_1, \dots, e_j\}$

Base case: $OPT(0) = 0$

$OPT(j) = \max(b_j + OPT(p(j)), OPT(j-1))$

Interpretation: either take e_j, & then adding best profit up to $p(j)$, or skip it & keep best among the best $j-1$.

→ To recover the optimal set, backtrack from $j=n$ taking e_j ~~wh~~ exactly when $b_j + OPT(p(j)) \geq OPT(j-1)$.

2b Events already sorted by f:

$e_1 = (2,3,40)$, $e_2 = (0,4,60)$, $e_3 = (2,4,30)$, $e_4 = (1,6,60)$,

$e_5 = (3,7,70)$, $e_6 = (5,7,20)$ $e_7 = (5,8,60)$ $e_8 = (7,8,50)$

$e_9 = (6,10,40)$

computing compatibility links $p(j)$:

$p(1) = 0$ (no earlier finish ≤ 2)

$p(2) = 0$ (no finish ≤ 0)

$p(3) = 0$ (no finish ≤ 2)

$p(4) = 0$ (no finish ≤ 1)

$p(5) = 1$ (largest $i < 5$ with $f_i \leq 3$ is $i = 1$)

$p(6) = 3$ (largest finish ≤ 5 is $f_3 = 4$)

$p(7) = 3$ (largest finish ≤ 5 is $f_3 = 4$)

$p(8) = 6$ (largest finish ≤ 7 is $f_6 = 7$)

$p(9) = 4$ (largest finish ≤ 6 is $f_4 = 6$)

Build the DP :

j	$e_j = (s_j, t_j, b_j)$	$p(j)$	included value $b_j + OPT(p(j))$	excluded value $OPT(j-1)$	$OPT(j)$	decision
1	(2,3,40)	0	40	0	40	take
2	(0,4,60)	0	60	40	60	take
3	(2,4,30)	0	30	60	60	skip
4	(1,6,60)	0	60	60	60	any - take or skip
5	(3,7,70)	1	70+40=110	60	110	take
6	(5,7,20)	3	20+60=80	110	110	skip
7	(5,8,60)	3	60+60=120	110	120	take
8	(7,8,50)	6	50+110=160	120	160	take
9	(6,10,40)	4	40+60=100	160	160	skip.

Backtracking from $j=9$ yields accepted bids:

$\{e_1, e_5, e_8\}$.

$e_1 = (2,3,40)$

$e_5 = (3,7,70)$

$e_8 = (7,8,50)$

They are pairwise compatible (end $\leq$ next start),

& the maximum total profit is

$\boxed{40 + 70 + 50 = 160}$