Exam 3 : April 28.

Reverse Delete algorithm

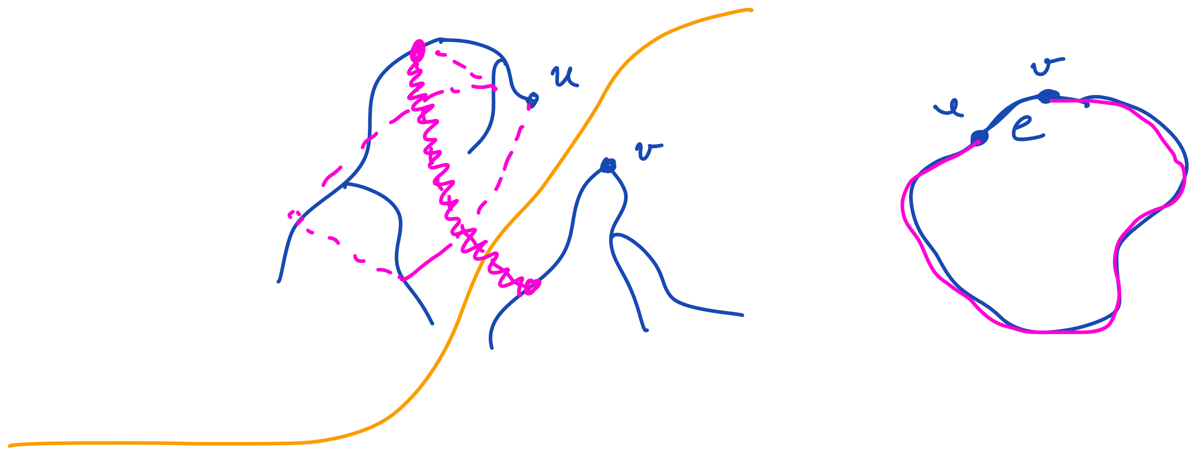
1. Sort edges in ↘ order of wts.

2. Process edges in the above order and delete the edge if removing it does not disconnect the graph.

Proof of Correctness

Lemma (Cycle Property): Let C be a cycle in an undirected graph $G=(V,E)$. Let e be an edge in C with the heaviest weight. Then e does not belong to any MST.

Proof : Assume for contradiction that there exists a MST, T, that contains e=(u,v).



Consider $T' = T - e$.

T' has <u>two</u> c c.

Follow the path (longer) going from u to v in C. Since u & v are on opposite sides of the cut, there must be an edge, f that crosses the cut and connects the two

Conn. components.

$\boxed{T'+f}$ is another

spanning tree whose total wt

$<$ wt(T), a contradiction!!

(Since $w_f < w_e$)

<u>Theorem</u>: Reverse Delete algorithm yields a MST.

<u>Proof</u>: Let $e=(u,v)$ be an edge about to be deleted by the reverse delete algorithm. We will prove that even "God" (opt. soln) will not include e.

Note that our alg. deletes e because removing e does not disconnect the graph. This means there is already a path between u & v in G that does not contain e. This path + e forms a cycle in G. Since our alg. processes edges in ↓ order of wts, e must be the edge with the heaviest wt in C. By the prev. lemma, no opt. soln. contains e.

At the end of our alg, we get a connected, acyclic graph. ✓.

Dynamic Programming.

Input: n intervals — 1, 2, ..., n

interval i

- start time S_i

- finish time f_i
- wt p_i denoting the profit

<u>Objective</u> : To output a set of non-overlapping intervals whose total profit is maximized.

We saw how to solve the problem when all p_is were equal to 1.

<u>Scratch work</u>.

- Consider intervals in $\nearrow$ order of their finish times. Let this order be $1, 2, \ldots, n$.

- Let OPT $[n]$ be the max. profit obtained by considering intervals $1, 2, \ldots, n$.

- Suppose we know that $n \notin$ OPT $[n]$

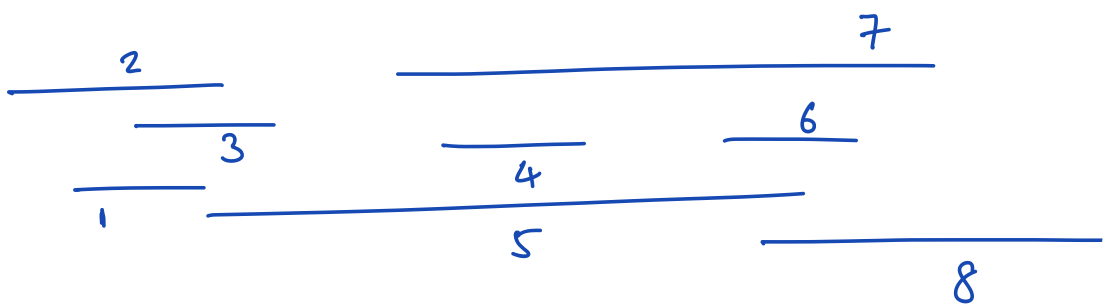
$OPT[n] = OPT[n-1]$.

• Suppose we know that $n \in OPT(n)$

prev(n)

$OPT(n) = OPT(\boxed{n-1}) + p_n$

$OPT(n) = \max \{ \quad , \quad \}$



$OPT[8] = p_8 + OPT[\ \underline{\ \ 4\ \ }\ ]$.

prev(j): interval with the largest finish time that does not overlap with interval j.

Consider intervals in $\nearrow$ order of their finish times. Let the numbering be $1, 2, \ldots, n$.

Subproblems

$P[j]$: max profit obtained by considering intervals $1, 2, \ldots, j$.

Our solution : $P[n]$.

Recurrence :

$$P[j] = \begin{cases} 0, & \text{if } j = 0 \\ \max\{P[j-1], P[\text{prev}(j)] + p_j\}, & \text{otherwise} \end{cases}$$

Alg:

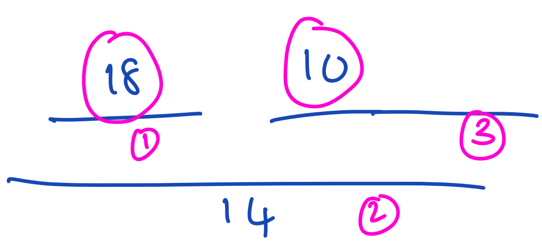
Profit (j)

if j = 0 then

return 0

else

return $\max\left\{ \overset{IH}{\text{Profit}(j-1)},\ \underset{IH}{\text{Profit}(\text{prev}(j))} + p_j \right\}$



$$OPT(3) = \max\{OPT(2),\ OPT(1) + 10\}$$

$$OPT(2) = \max\{\underset{18}{OPT(1)},\ \underset{8}{OPT(0)} + 14\}$$

$$OPT(1) = \max\{OPT(0),\ OPT(0) + 18\}$$

$$OPT(1) = 18.$$

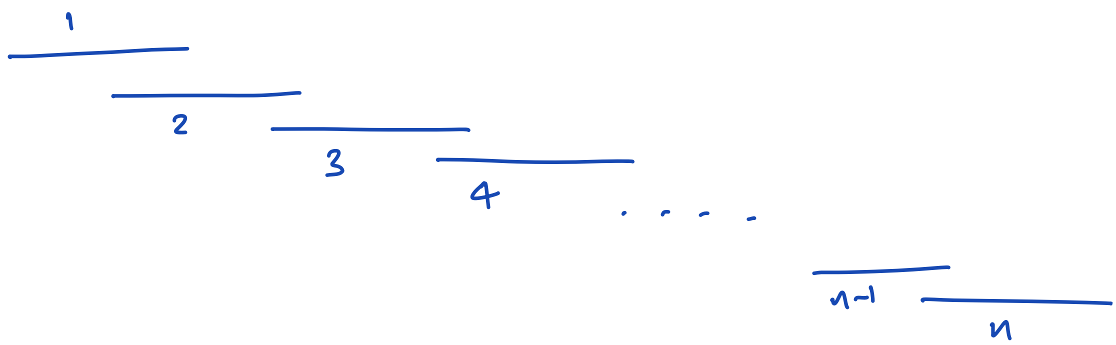
$$OPT(2) = OPT(1) = 18.$$

$$OPT(3) = \max\{18,\ 18 + 10\}$$

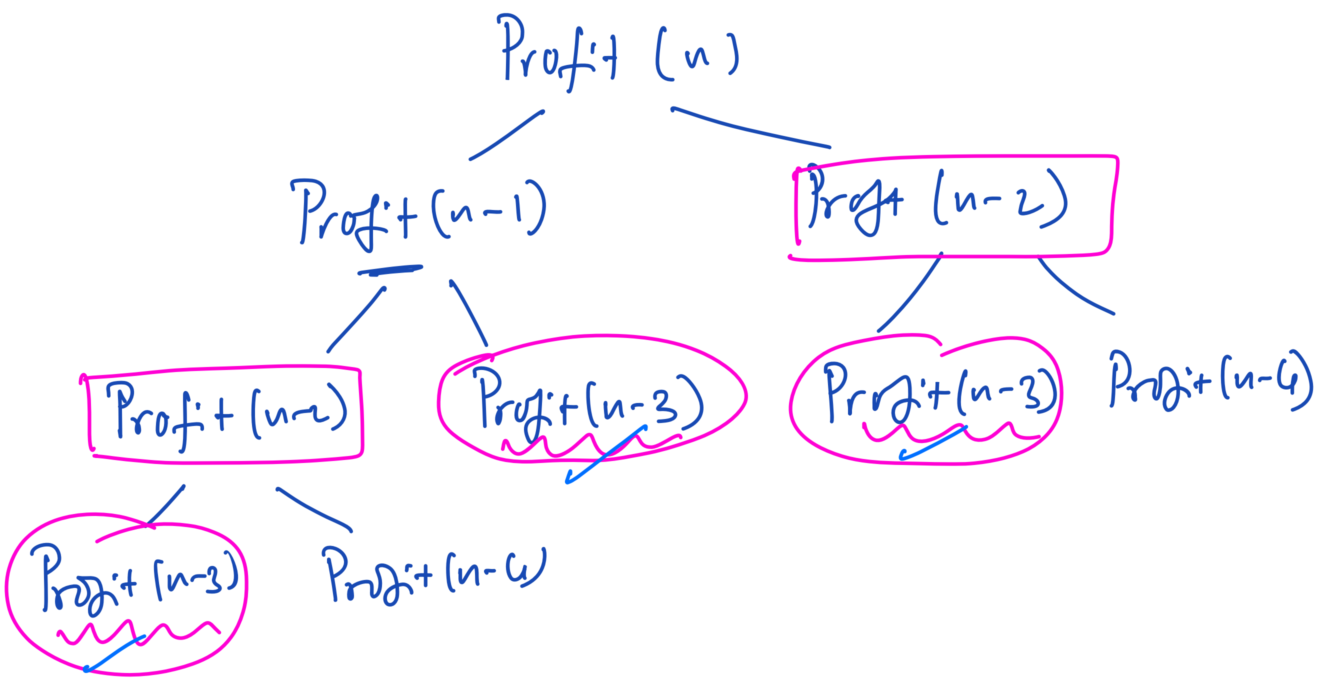
$$= \underline{\underline{28}}$$

It is easy to see that the algorithm always works. However, the running time of the alg. is exponential.

Bad example.



Note that $prev(j) = j-2$, $\forall\ 2 \le j \le n$.



$$
\begin{aligned}
T(n) &= T(n-1) + T(n-2) \\
&> 2T(n-2) \\
&> 2^2\, T(n-4)
\end{aligned}
$$

$> 2^3 \, T(n-6)$

$\vdots$

$> 2^k \, T(n - 2 \cdot k)$

Recursion bottoms out when $n - 2k = 0$ or $k = n/2$.

$\therefore \boxed{T(n) = \Omega(2^{n/2})}$.

```
po2(n)
  if n=0 then
     return 1
  else
     x ← po2(n-1)
     return x + x.
```

```
po2(n)
  if n=0 then
     return 1
  else
     return po2(n-1) +
              po2(n-1).
```

MemoizedProfit (j)

if $j = 0$ then

  return 0

else

  if $P[j]$ exists then

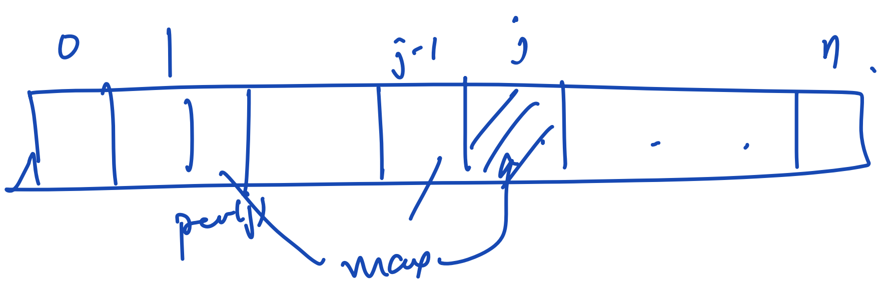
    return $P[j]$

  else

    $P[j] \leftarrow \max \left\{ \text{MemoizedProfit}(j-1),\ \text{MemoizedProfit}(\text{prev}(j)) + p_j \right\}$

    return $P[j]$

Running time: $O(n)$.



Dynamic Prog.

- Bottom-up approach.

DP_Profit (I)

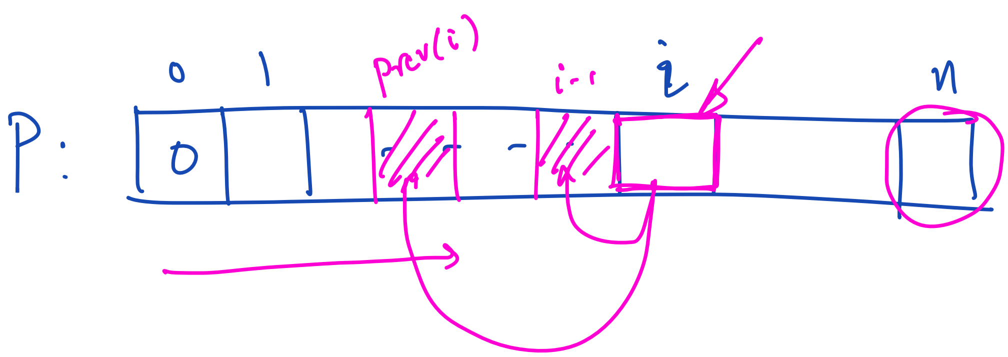
  $P[0] \leftarrow 0$

  for $i \leftarrow 1$ to n **do**

    $P[i] \leftarrow \max\{P[i-1],\ P[prev(i)] + p_i\}$

    constant

  return $P[n]$.



Int MaxProfit (P)

$S \leftarrow \{ \quad \}$

$j \leftarrow n$

while $j > 0$ do

if $P[j] > P[j-1]$ then

$S \leftarrow S + \{j\}$

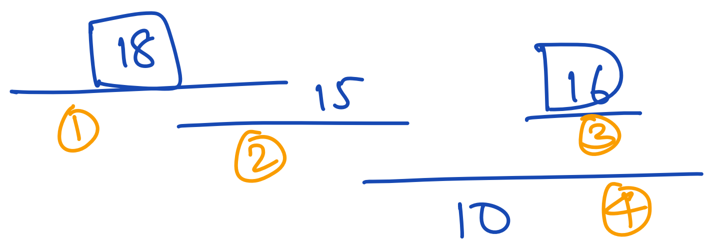
$j \leftarrow prev(j)$

else

$j \leftarrow j - 1$

return S

Example.



P:

0	1	2	3	4
0	18	18	28	34

$$P[1] = \max\{P[0],\ P[0] + 18\}$$

$$P[2] = \max\{P[1],\ P[0] + 15\}$$

$$P[3] = \max\{P[2],\ P[1] + 10\}$$

$$P(4) = \max\{P(3),\ P(2) + 16\}$$

$$= \max\{28,\ 18 + 16\}$$

$= 34$

$S = \{4, 1\}$ ✓