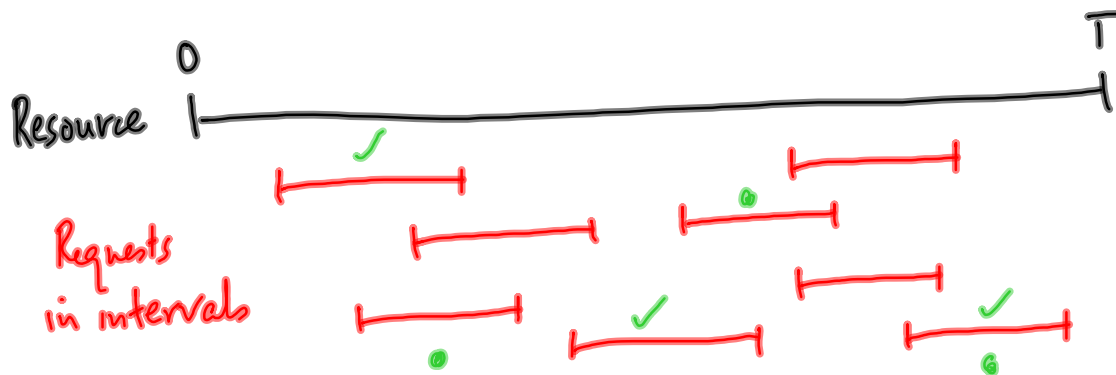


Dynamic Programming

Interval scheduling



Compute the largest set of requests that can be served

Request i has a start time $s_i <$ finish time f_i

$$\forall i. 0 \leq s_i < f_i \leq T$$

Greedy

Sort by f_i

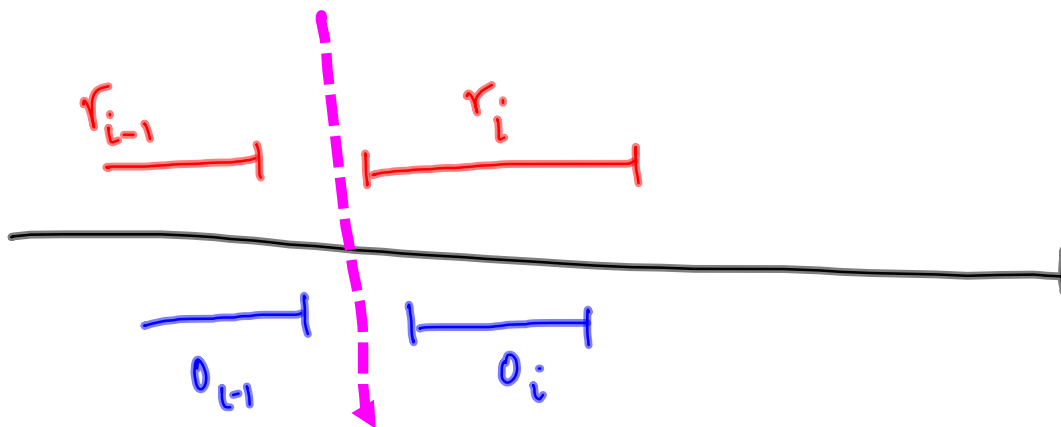
Process in increasing order of f_i

Greedy solution $r_{i_1} < r_{i_2} < r_{i_3} \dots < r_{i_k}$

Other optimal solution $o_{j_1} < o_{j_2} < o_{j_3} \dots < o_{j_m}$

To show $k \geq m$

$f_{i_1} \leq f_{j_1}$ By induction $f_{i_\ell} \leq f_{j_\ell} \forall \ell$



If $f_{o_i} < f_{r_i}$, the greedy
 algorithm would pick o_i , not r_i
 $\therefore f_{r_k} \leq f_{o_m}$ & $\therefore m > k$ is a contradiction

Weighted Interval Scheduling

Each request comes with a "rent" (non negative)

Choose a set of jobs to satisfy that maximizes rent

No known greedy solution

Revert to trying all possibilities, efficiently

First sort by starting times



exclude i : $\text{best}(i+1)$

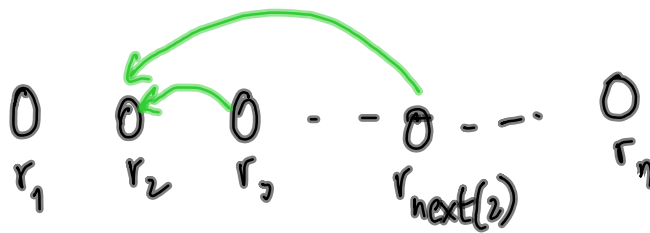
$$\text{Best}(n) = c_n$$

$$\forall j < n \quad \text{Best}(j) = \max \left(c_j + \text{Best}(\text{next}(j)), \text{Best}(j+1) \right)$$

Compute $\text{Best}(n), \text{Best}(n-1) \dots \text{Best}(1)$

Final answer is $\text{Best}(1)$

Simultaneously compute $\text{Included}(j)$, $n \geq j \geq 1$



Symmetrically

define $prev(j)$

process right to left

Only virtue is that base case is $Best(1)$

All $c_i = 1 \Rightarrow$ this is the same as the
older problem, can use this DP
to solve that

Spelling correction

Suggested word should be "close" to the word you typed

How many changes to transform one to another

Insert a letter

Delete a letter

Change a letter

Edit distance between words

Levenshtein distance

M A R C H

A P R I L

Try to align them

M A • R • • C H
• A P R I L • •
1 2 3 4 5 6

M A • R C H
• A P R I L
1 2 3 4

$$u = u_1 u_2 \dots u_m$$

$$v = v_1 v_2 \dots v_n$$

Edit Distance (u, v)

$$u_1 \neq v_1$$

$$\begin{array}{cc}
 u_1 & \begin{bmatrix} u_2 \dots u_m \\ v_1 \dots v_n \end{bmatrix} \\
 \bullet & \\
 & \bullet \quad v_1 \begin{bmatrix} u_1 \dots u_m \\ v_2 \dots v_n \end{bmatrix}
 \end{array}$$

$$\begin{array}{c}
 \begin{array}{|c|} \hline u_1 \\ \hline v_1 \\ \hline \end{array} \begin{array}{|c|} \hline u_2 \dots u_m \\ \hline v_2 \dots v_n \\ \hline \end{array} \\
 \text{lost of 1}
 \end{array}$$

$$u_1 = v_1$$

 u_1 v_1

✓

$$\begin{array}{c} u_2 \dots u_m \\ v_2 \dots v_n \end{array}$$

"greedy"

To be cautious, also allow

 u_1

•

$$\begin{array}{c} u_2 \dots u_m \\ v_2 \dots v_n \end{array}$$

•

 v_1

$$\begin{array}{c} u_1 \dots u_m \\ v_2 \dots v_n \end{array}$$

$ED(i, j)$ is edit distance of

$$\begin{array}{c} u_i u_{i+1} \dots u_m \\ v_j v_{j+1} \dots v_n \end{array}$$

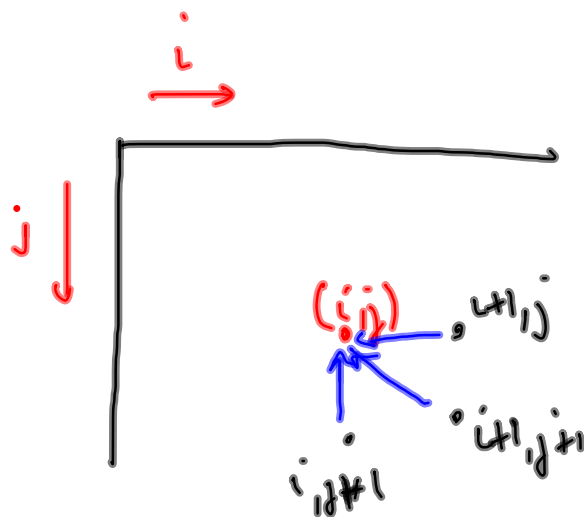
$$ED(i, j) = \min \left(\begin{array}{l} 1 + ED(i+1, j), \\ 1 + ED(i, j+1), \\ \text{diff}(i, j) + ED(i+1, j+1) \end{array} \right)$$

$$\text{where } \text{diff}(i, j) = \begin{array}{ll} 0 & \text{if } u_i = v_j \\ 1 & \text{if } u_i \neq v_j \end{array}$$

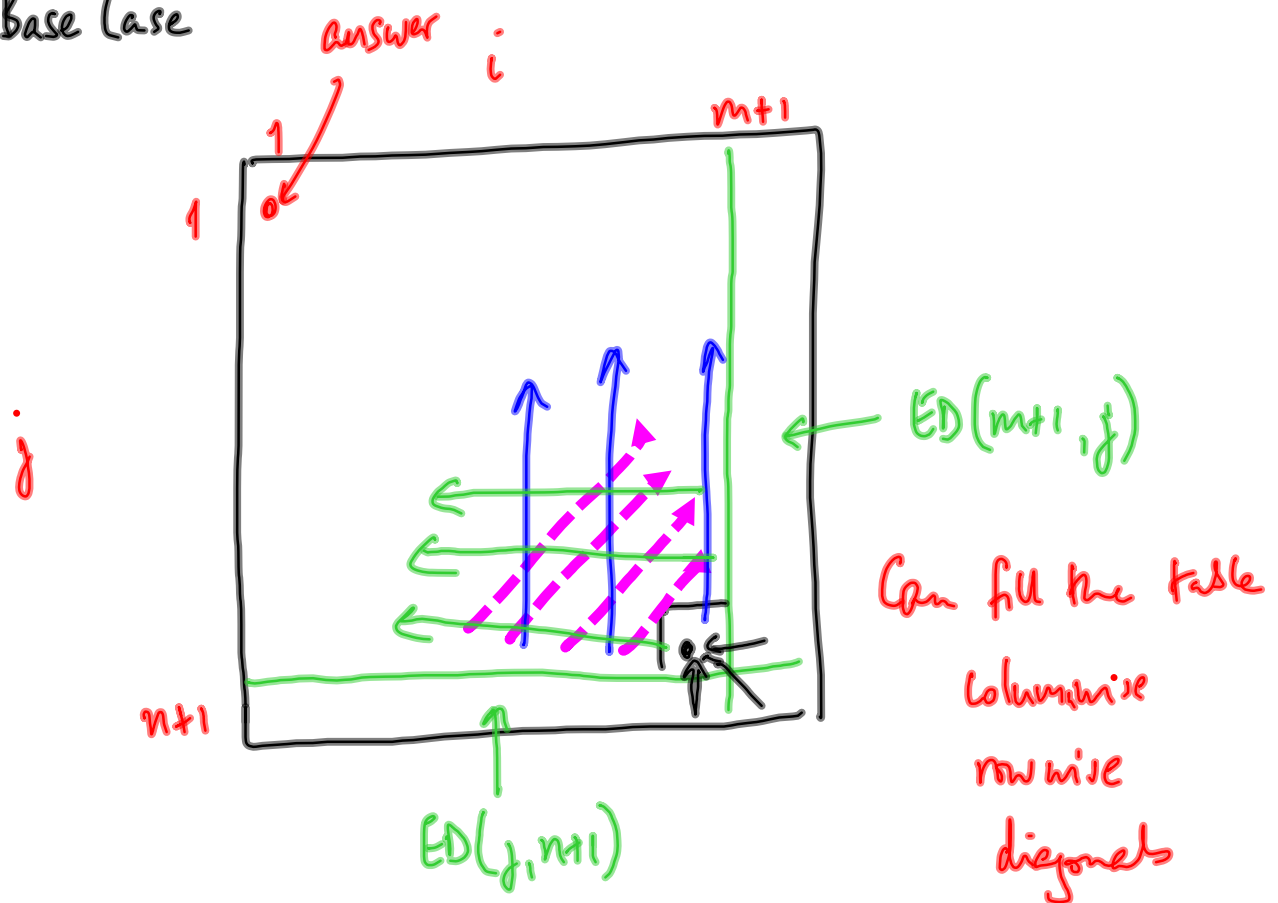
Convenient to let arguments to ED to run up to $m+1, n+1$

$$\forall j \quad \text{ED}(m+1, j) = n - j + 1 \quad \text{Insert } v_j v_{j+1} \dots v_n$$

$$\forall j \quad \text{ED}(j, n+1) = m - j + 1 \quad \text{Insert } u_j u_{j+1} \dots u_m$$



Base Case



What if all operations are wrt of equal cost?

C_i insert cost

C_d delete cost

C_r replace cost

$$ED(i,j) = \min \left(\begin{aligned} &C_d + ED(i+1,j), \\ &C_i + ED(i,j+1), \\ &\text{diff}(i,j) + ED(i+1,j+1) \end{aligned} \right)$$

$$\text{diff}(i,j) = C_r \text{ if } u_i \neq u_j$$