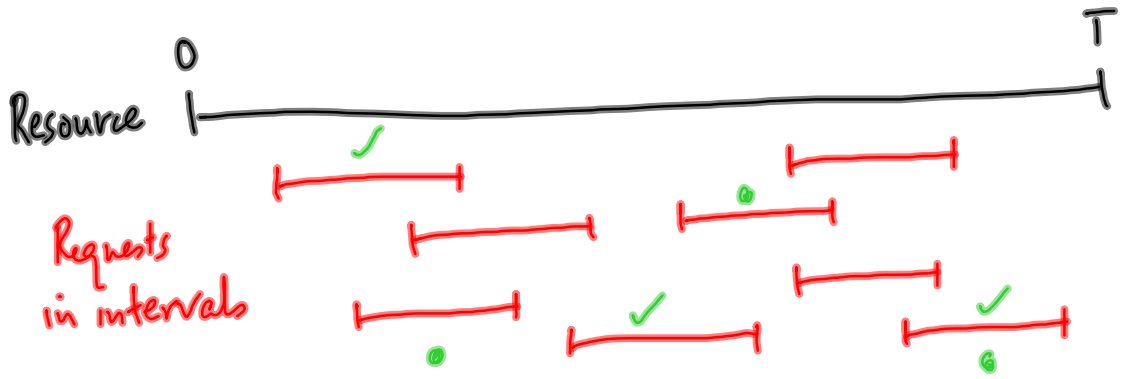
Dynamic Programming

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

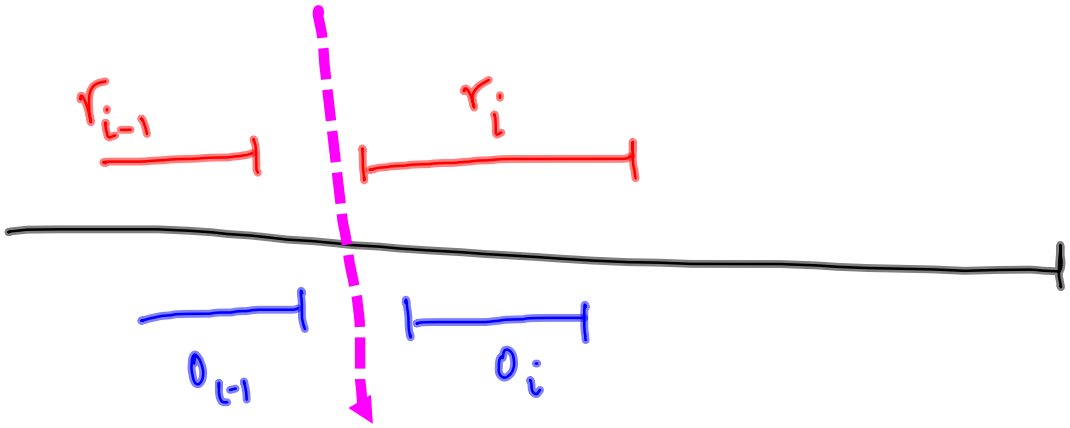
Process in increasing order of f_i

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

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

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

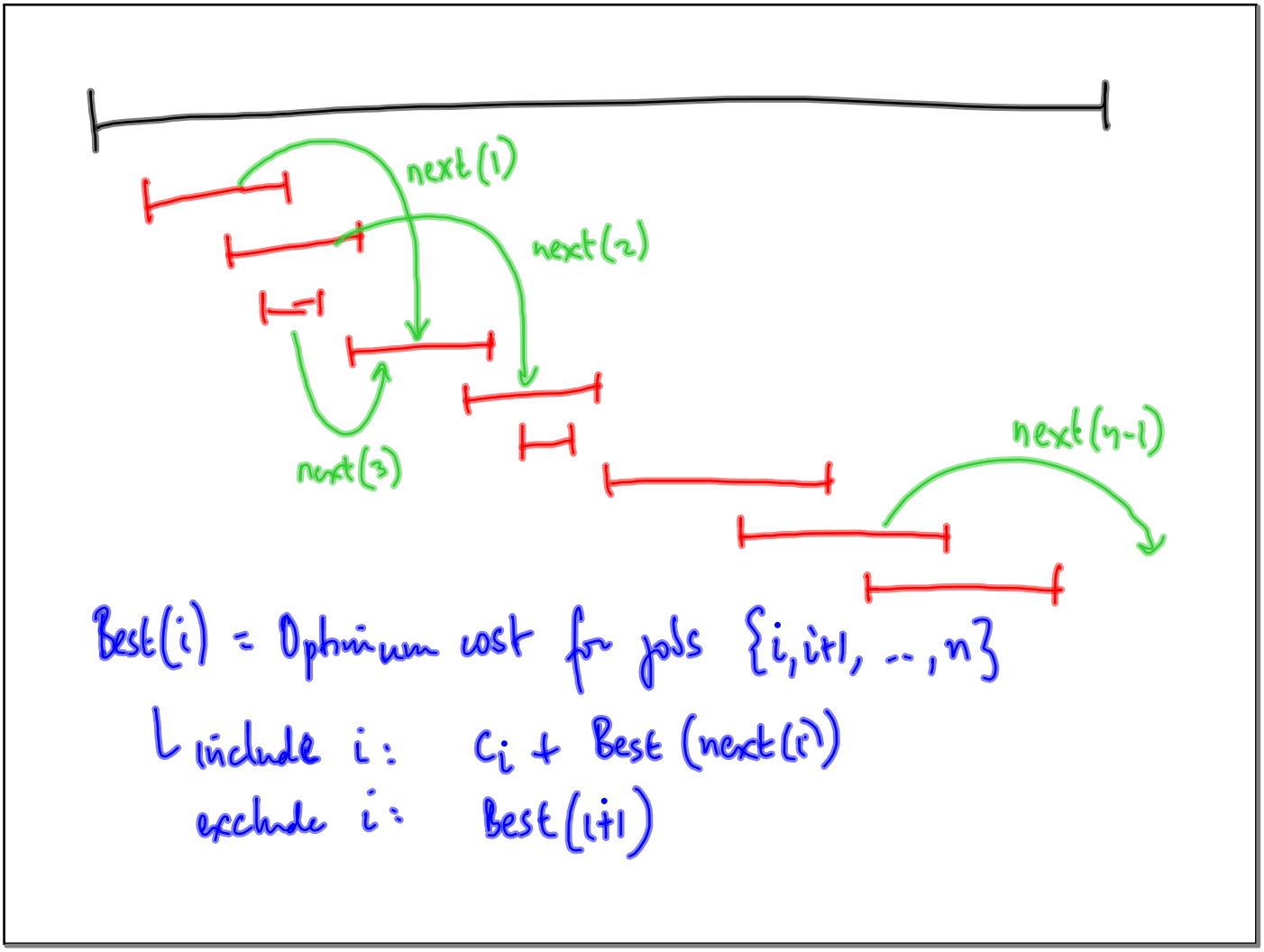
Each request comes with a "rent" (non negative)

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


next(1)
next(2)
next(3)
next(n-1)
Best(i) = Optimum cost for jobs {i, i+1, .., n}
include i: c_i + Best(next(i))
exclude i: Best(i+1)

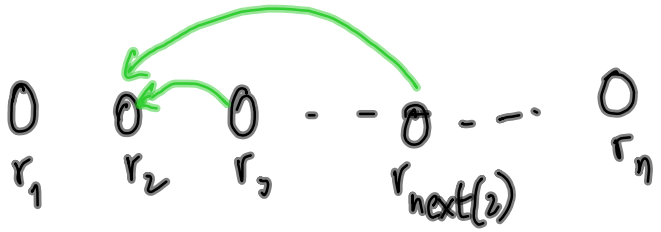
$Best(n) = c_n$

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

Compute $Best(n), Best(n-1) \ldots Best(1)$

Final answer is $Best(1)$

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



Symmetrically

define prev(j)

process right to left

Only nitpick 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 — Levenstein distance

MARCH APRIL

Try to align them

```
MA.R..CH
.APRIL..
1   2  3456
```

```
MA.RCH
.APRIL
1   2  34
```

$u = u_1 u_2 \cdots u_m$

$v = v_1 v_2 \cdots v_n$

Edit Distance (u, v)

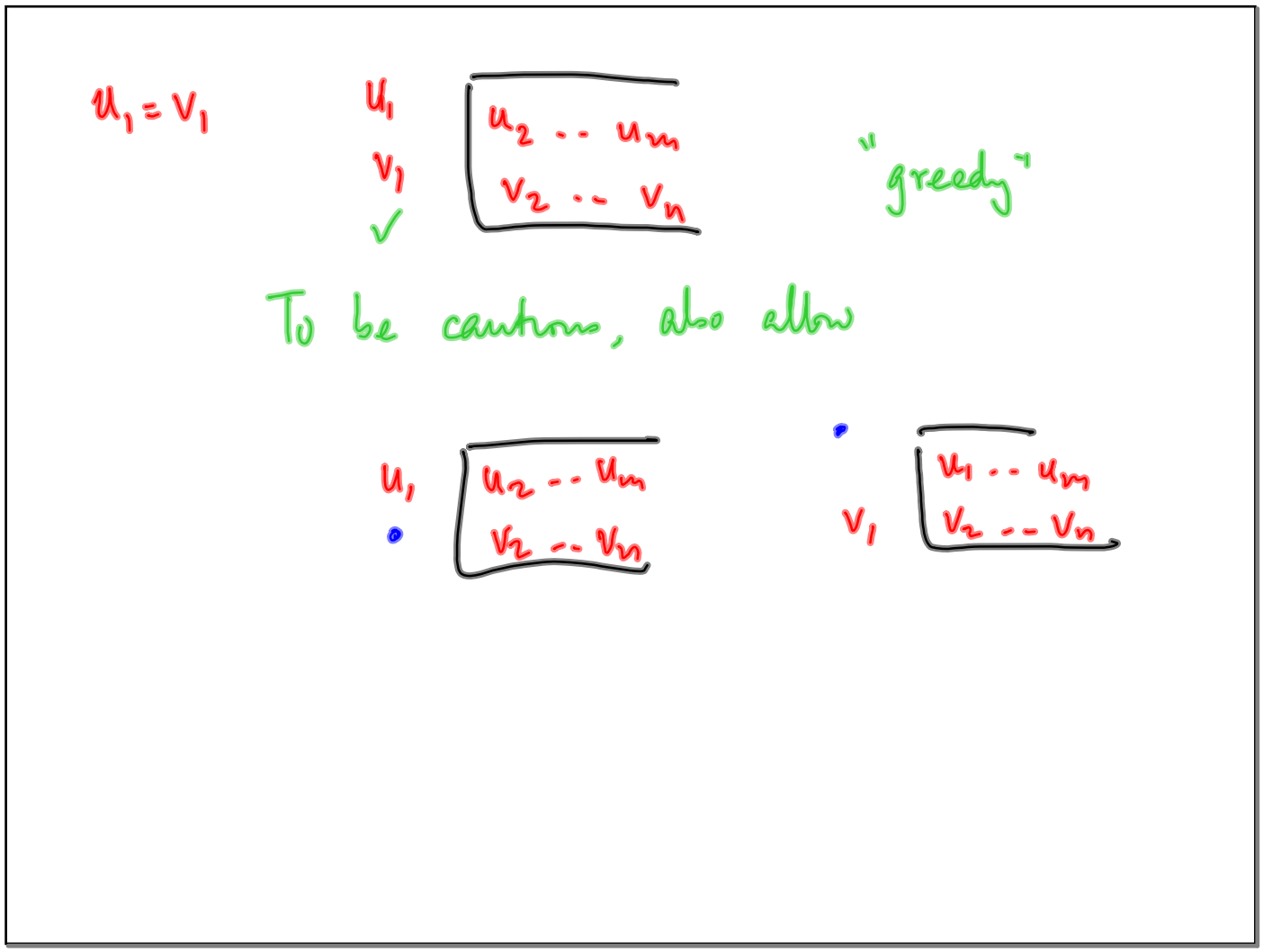
$u_1 \neq v_1$

u_1 | $u_2 \cdots u_m$ / $v_1 \cdots v_n$

v_1 | $u_1 \cdots u_m$ / $v_2 \cdots v_n$

u_1 / v_1 | $u_2 \cdots u_m$ / $v_2 \cdots v_n$

Cost of 1


$u_1 = v_1$
u_1
$u_2 \dots u_m$
v_1
$v_2 \dots v_n$
✓
"greedy"
To be cautious, also allow
u_1
$u_2 \dots u_m$
$v_2 \dots v_n$
v_1
$u_1 \dots u_m$
$v_2 \dots v_n$

$ED(i,j)$ is edit distance of $u_i u_{i+1} \ldots u_m$ and $v_j v_{j+1} \ldots v_n$

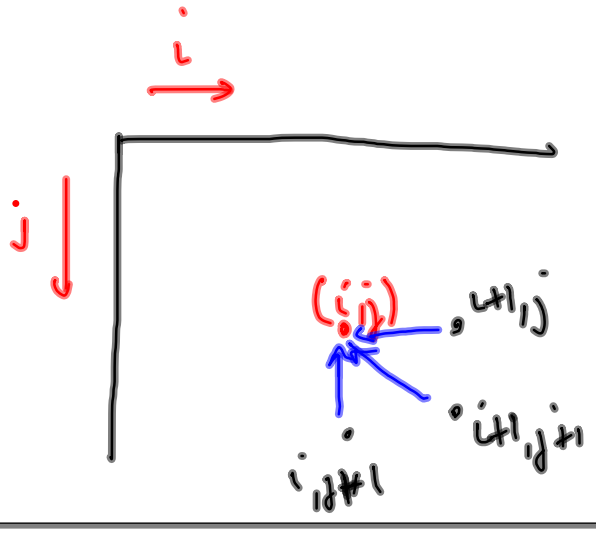
$$ED(i,j) = \min\Big(1 + ED(i+1,j),\ 1 + ED(i,j+1),\ diff(i,j) + ED(i+1,j+1)\Big)$$

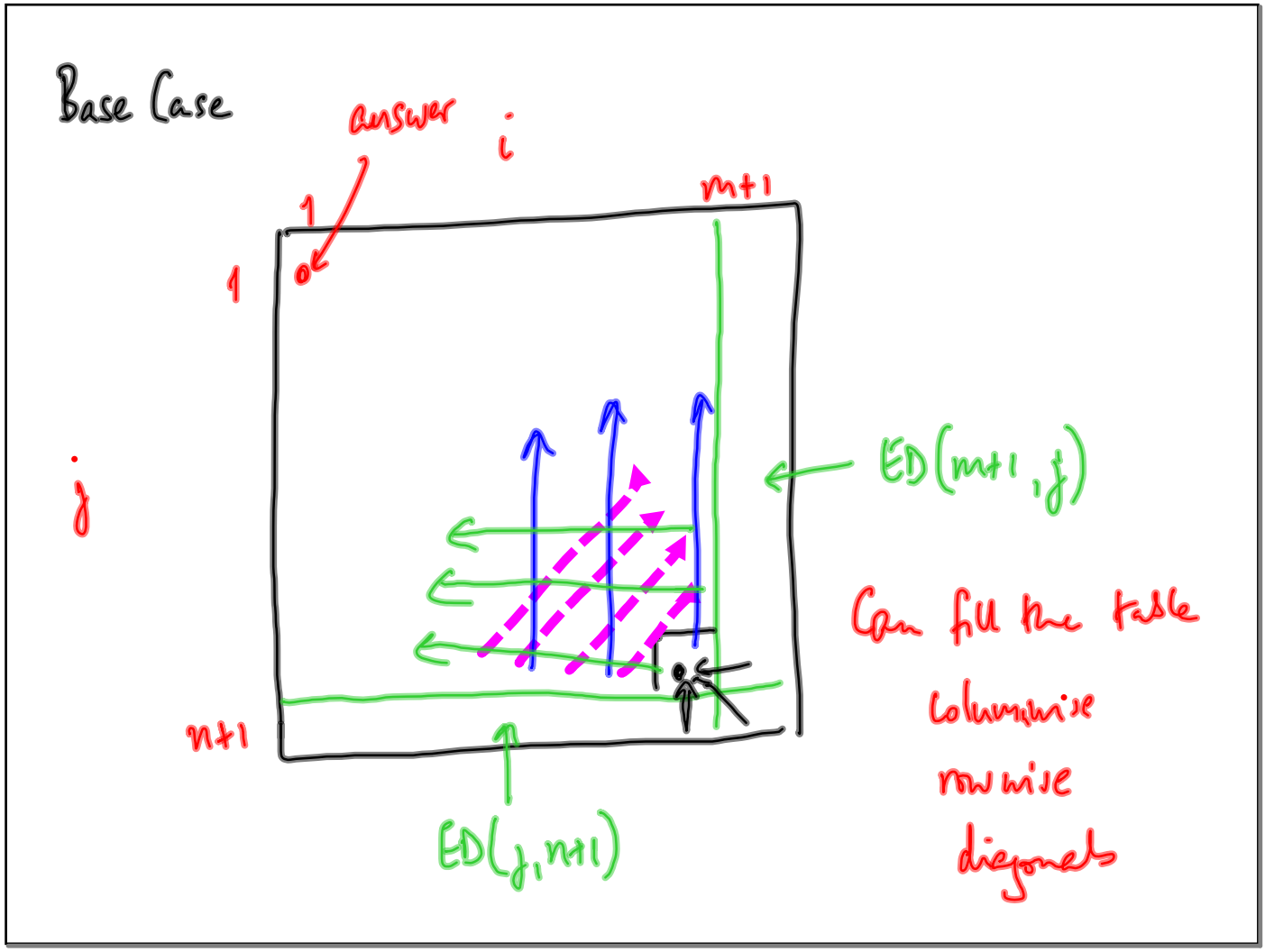
where $diff(i,j) = 0$ if $u_i = v_j$, 1 if $u_i \neq v_j$.

Convenient to let arguments to ED to run upto m+1, n+1

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

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




Base Case
answer
i
m+1
1
1
0
j
ED(m+1, j)
Can fill the table
columnwise
rowwise
diagonals
n+1
ED(j, n+1)

What if all operations are not of equal cost?

c_i insert cost

c_d delete cost

c_r replace cost

$$ED(i,j) = \min \begin{cases} c_d + ED(i+1,j), \\ c_i + ED(i,j+1), \\ diff(i,j) + ED(i+1,j+1) \end{cases}$$

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