Improving space using divide-&-conquer

- Compute the sequence with min edit distance in $O(mn)$ time and $O(m+n)$ space.

Shortest path from $(0,0)$ to (n,m)

$$= \min_i \left\{ \text{shortest path from } (0,0) \text{ to } (i, m/2) + \text{shortest path from } (i, m/2) \text{ to } (n,m) \right\}$$

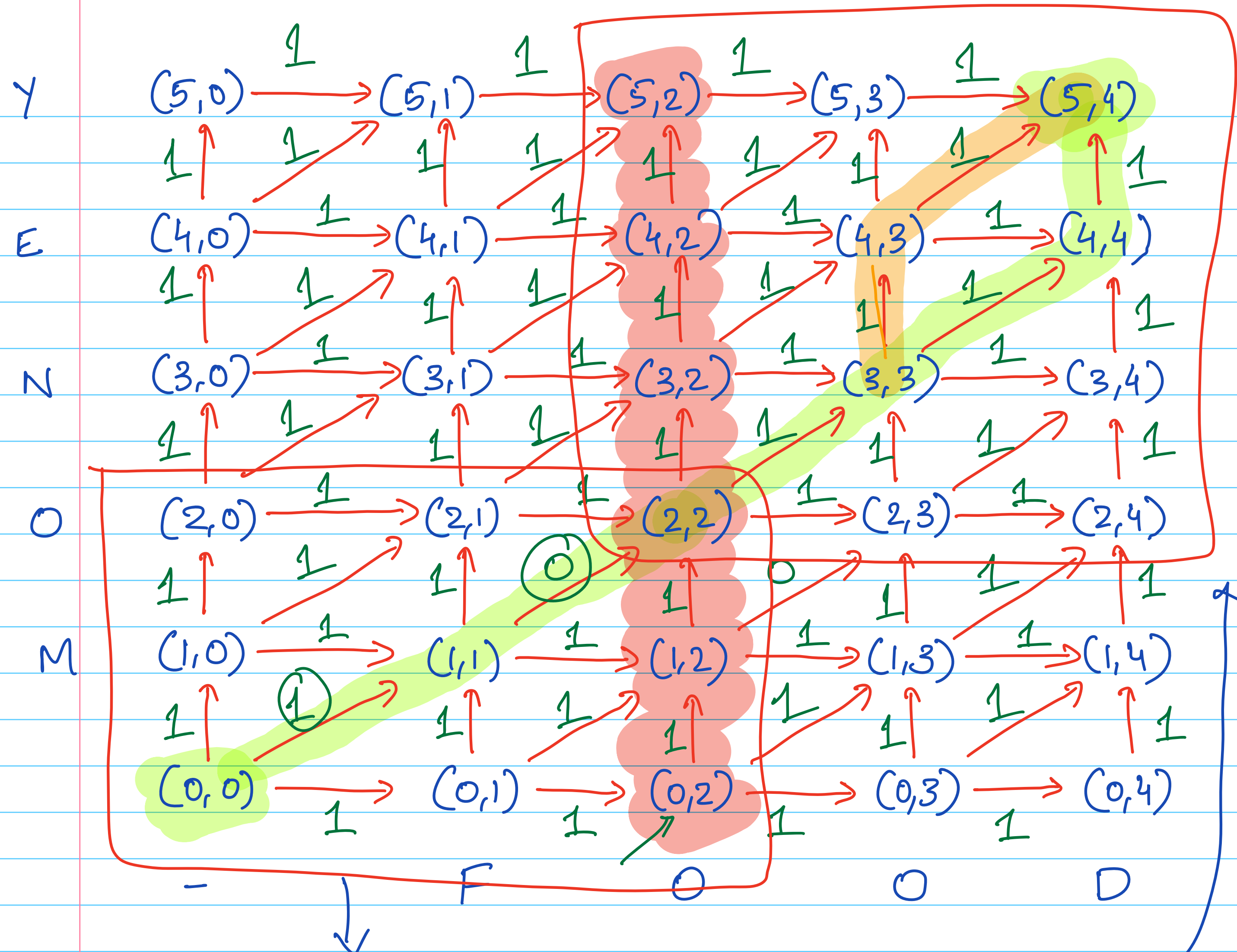
Shortest path from $(0,0)$ to $(i, m/2)$

$$= \text{Edit}(i, m/2)$$

Shortest path from (i,j) to (n,m)

$$= \widetilde{\text{Edit}}(i,j)$$ → edit distance for $A[i+1, \ldots, n]$, $B[j+1, \ldots, m]$

$$\widetilde{\text{Edit}}(i,j) = \begin{cases} n-i & \text{if } j=m \\ m-j & \text{if } i=n \\ \min \begin{cases} \widetilde{\text{Edit}}(i+1,j)+1 \\ \widetilde{\text{Edit}}(i,j+1) + \\ \widetilde{\text{Edit}}(i+1,j+1) + [A[i] \neq B[i]] \end{cases} \end{cases}$$



Edit(2,2)

$\underbrace{\text{MO , FO}}_{\text{edit distance}}$

$\underbrace{\text{NEY} \quad \text{OD}}_{\text{edit distance}}$

find

Column $m/2$: $\min_i \{ Edit(i, m/2) + \widetilde{Edit}(i, m/2) \}$

$O(m+n)$ space

$\Rightarrow$ The shortest path goes through $(i, m/2)$

divide step: Recursively compute the shortest path from $\boxed{(0,0) \text{ to } (i, m/2)}$ & from $\boxed{(i, m/2) \text{ to } (n, m)}$

reuse space for the recursive calls

$$T(n,m) \le cmn + \underbrace{T(i, m/2)} + \underbrace{T(n-i, m/2)}$$

constant of the original edit dist alg

recursive calls

$$T(n,m) = \Theta(mn)$$

Space complexity $= \Theta(m+n)$

Knapsack problem → NP-hard

Set of items $1, 2, 3, \ldots, n$ → exactly one copy of the items
↗ infinite copies

each item i has value v_i & weight w_i

Knapsack → hold weight W

find subset $S \subseteq [n]$ s.t $\sum_{i \in S} w_i \leq W$

and $\sum_{i \in S} v_i$ is maximized

$KP(w)$ = max value attainable with KS of capacity w

$$KP(W) = \max_i \{ v_i + KP(W - w_i) \}$$

↳ not keeping track of the item i that was included

$KP(i, w)$ = max value attainable with KS of capacity w using items $\{1, 2, .., i\}$

$KP(n, W)$ ← soln. to knapsack problem

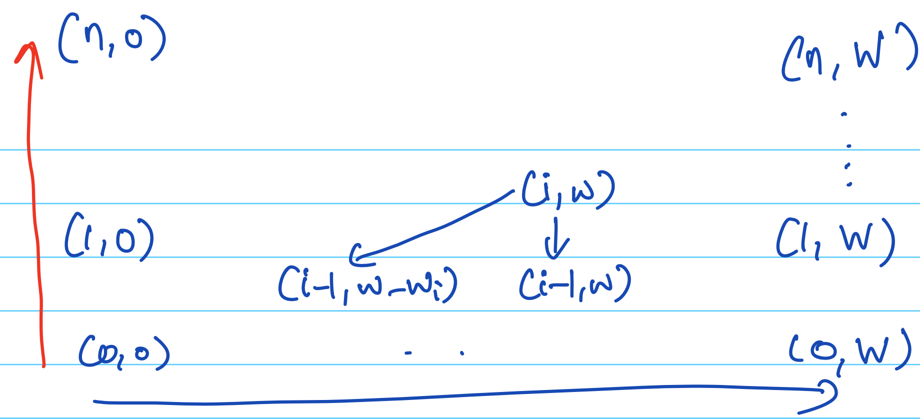
$$KP(i, w) = \max \{ KP(i-1, w), KP(i-1, w - w_i) + v_i \}$$

↓ exclude i from the knapsack

↳ include i in the knapsack

Base cases: $KP(0, w) = KP(i, 0) = 0$

Dependency graph: # of vertices: nW



Running time: $O(nW)$ → value, not the size of the repr.

↳ pseudo-polynomial