CS124 Lecture 11

March 2nd, 2020

Dynamic Programming, continued

Requirements:

- what is being computed
- clearly written + stated recursion ☆ very important
- Analysis (runtime and space)

continued

activate → caveat

$A[1 \ldots n]$

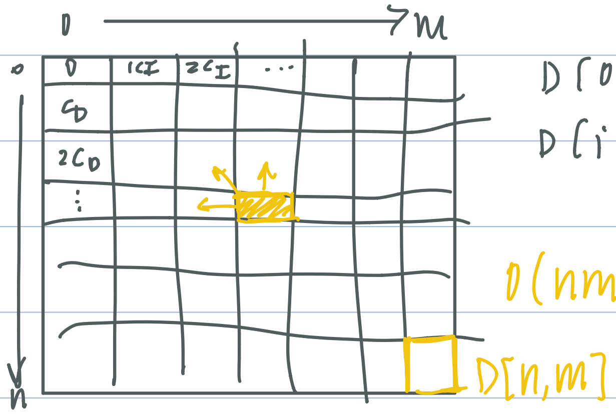
$B[1 \ldots m]$

$D[i,j]$ = edit distance from $A[1..j]$ to $B[1..j]$

$$D[i,j] = \min \begin{cases} D(i-1, j) + C_D & \text{delete} \\ D(i, j-1) + C_I & \text{insert} \\ D(i-1, j-1) + C_R & \text{replace} \\ D(i-1, j-1) + C_m & \text{match if } A[i] == B[j] \end{cases}$$

$D[0,0]$ base case $= 0$

memorizing / array filling



$D[0,i]$ = insertions

$D[i,0]$ = deletions

$O(nm)$ time algorithm: each cell looks at const # of other cells, $O(1)$ per cell

$O(nm)$ space: $O(n)$ or $O(m)$ w/ optimization

All pairs shortest path

(positive edge weights)

$D[i,j]$: shortest path from i to j

$D_0[i,j]$: original distance matrix

$D_k[i,j]$: shortest path from i to j using only vertices $\{1, 2, \ldots, k\}$ as intermediary points

$$D_k[i,j] = \min \begin{cases} D_{k-1}[i,j] & \text{when } k \text{ is not on path (or } k = i,j) \\ D_{k-1}[i,k] + D_{k-1}[k,j] & \text{when } k \text{ is on path} \end{cases}$$

for $k = 1$ to n:

  for $i = 1$ to n:

    for $j = 1$ to n:

      $D_k[i,j] = \min(D_{k-1}[i,j],\ D_{k-1}[i,k] + D_{k-1}[k,j])$

$O(n^3)$ running time

Traveling Salesman Problem (TSP)

n cities, d_{ij} = distances > 0

"tour" that visits all n cities and returns home w/ minimal length

__Dummy Solution__: try all tours, $(n-1)!$ tours $\rightarrow 2^{O(n \log n)}$

__DP Solution__:

shortest path from i to j that must pass through additional vertices in S.

$C(S, j)$ = paths that start at 1, end at j, and go through all vertices of S. $j \in S$.

Base case: $C(\{j\}, j) = d_{1j}$

* Recursive case: $C(S, j) = \min_{i \in S, i \neq j} C(S - \{j\}, i) + d_{ij}$

similar to last q on HW

recursion over all last vertices before j

Array filling on $O(2^n \cdot n)$ size array

for all j: $C(\{j\}, j) = d_{1j}$

for size set s = 2 to $n-1$:

for all subsets of size s:

$C(S, j) = \min_{i \in S, i \neq j} C(S - \{j\}, i) + d_{ij}$

$O(2^n \cdot n^2)$ runtime algorithm

OPT tour: $\min_{i} C(\{2, \dots n\}, i) + d_{i1}$.