LECTURE #12

CS 170

Spring 2021

Last time:

- introduction to dynamic programming ↙ ≈ recursion + memoization (with explicit solving order)
- examples
  - shortest paths in a DAG
  - longest increasing subsequence
  - edit distance
  - Knapsack (w/o repetition)

DP Recipe:

1. define problems
2. set boundaries
3. give recurrence
4. specify order (and give time)

Today:

More examples of DP solutions:

- Knapsack with repetition
- chain matrix multiplication
- all-pairs shortest paths
- traveling salesperson problem

Knapsack With Repetition

Given a maximum weight W and value-weight pairs $(v_1, w_1), \ldots, (v_n, w_n)$, output a multi-set of weight $\leq W$ of highest value.

(ok to pick the same item more than once)

- Problems: $K(w) :=$ max value achievable with capacity $\leq w$
- Boundaries: $K(0) = 0$
- Recurrence: $K(w) := \max_{i=1,\ldots,n} \{ K(w - w_i) + v_i \mid w_i \leq w \}$
- Efficiency:
  - solve in increasing weight w: $K(0), K(1), \ldots, K(W)$
  - each computation takes $O(n)$ time
  - total time is $O(n \cdot W)$

The algorithm is only weakly polynomial time.
Polynomial time would have been poly$(n, \log W)$,
but we don't expect this because the problem is NP-complete

Chain Matrix Multiplication

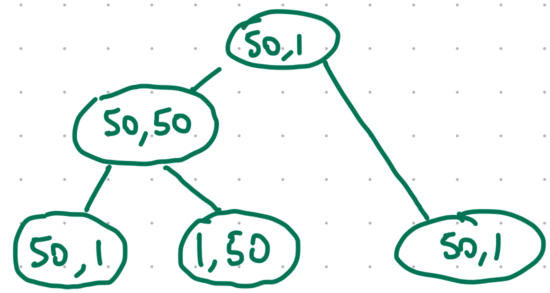
Multiplying x-by-y matrix A and y-by-z matrix B takes xyz ops.
(We have also seen how to do better but this is not important for now.)

Today: find strategy to multiply matrices $A_1, A_2, \dots, A_n$ as cheaply as possible.

Example: 50-by-1 A, 1-by-50 B, 50-by-1 C. The result is 50-by-1 R.

- option 1: $(A \cdot B) \cdot C = D \cdot C$

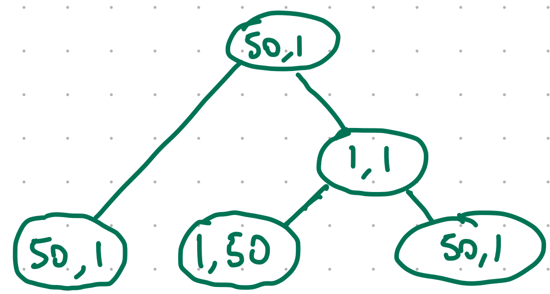
  $50 \cdot 1 \cdot 50 + 50 \cdot 50 \cdot 1 = 5000$



100x better!

- option 2: $A \cdot (B\,C) = A \cdot E$

  $1 \cdot 50 \cdot 1 + 50 \cdot 1 \cdot 1 = 100$



The association order matters!

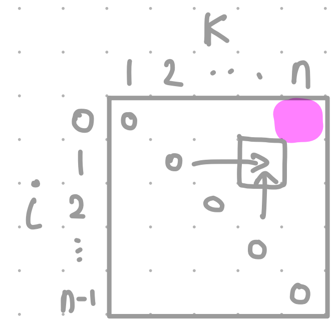
In general, the goal is to find the optimal association order for

$$A_1 \;,\; A_2 \;,\; \dots \;,\; A_n$$
$$m_0\text{-by-}m_1 \quad m_1\text{-by-}m_2 \quad \quad m_{n-1}\text{-by-}m_n$$

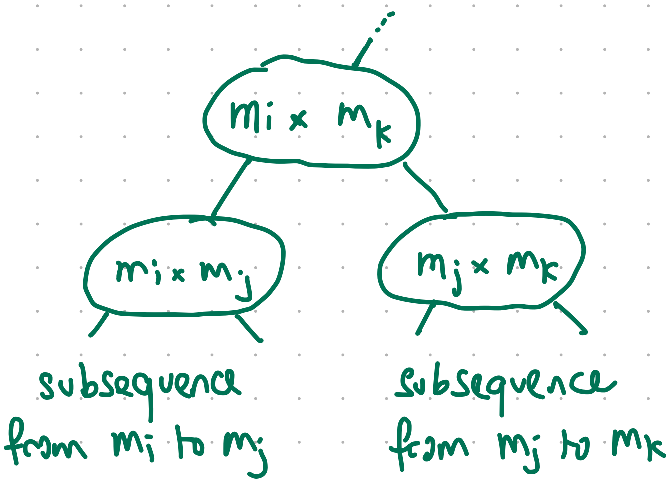
in the example the input is $(m_0, m_1, m_2, m_3) = (50, 1, 50, 1)$

input: list of positive integers $m_0, m_1, \dots, m_n$ representing matrix dimensions (the matrices themselves don't matter)

output: parenthetization of the list (a tree on it)



We solve the problem via dynamic programming:



The cost so far is:

$m_i \cdot m_j \cdot m_k$ + cost_left + cost_right

- subproblems:

  $C(i,k) :=$ optimal cost for subsequence $m_i, \dots, m_k$
  [multiplying $A_{i+1} \cdots A_k$]

- boundaries:

  $C(0,1) = C(1,2) = \cdots = C(n-1,n) = 0$

- recurrence: $A_{i+1} \cdots A_j$ $\quad A_j \cdots A_k$ $\quad$ combine

  $$C(i,k) := \min_{i<j<k} \{ C(i,j) + C(j,k) + m_i m_j m_k \}$$

- cost: $O(n^2)$ subproblems, computing each is $O(n)$

  $\Rightarrow O(n^3)$ cost to find $C(0,n)$.

All-Pairs Shortest Paths

distance from u to v

Given graph $G=(V,E)$ and lengths $\ell: E \to \mathbb{Z}$, find $\{\text{dist}(u,v)\}_{u,v \in V}$.

Idea: Run Bellman-Ford for every possible source: $|V| \cdot O(|V| \cdot |E|) = O(|V|^2 \cdot |E|)$.

Better: $O(|V|^3)$ via dynamic programming ($|E| \geq |V|-1$ if G is connected)

Floyd-Warshall algorithm

- The subproblems are:

  $d(u,v,i)$ = shortest path from u to v using intermediate nodes in $\{1,2,\ldots,i\}$

- Boundaries are:

$$d(u,v,0) = \begin{cases} \text{if } (u,v) \in E: & \ell(u,v) \\ \text{if } (u,v) \notin E: & \infty \end{cases}$$

- The recurrence is:

$$d(u,v,i) = \min\{d(u,v,i-1),\ d(u,i,i-1) + d(i,v,i-1)\}$$

- Efficiency:
  - init boundaries: $O(|V|^2)$
  - table has $|V|^3$ entries and filling each is $O(1)$

  total cost is $O(|V|^3)$.

Traveling Salesperson Problem

cycle visiting each vertex once

Given graph $G=(V,E)$ and lengths $\ell: E \to \mathbb{Z}$, output a shortest tour of G.

Straightforward approach: try all tours.

WLOG start at vertex 1. There are $\leq (n-1)!$ tours. Evaluating a tour costs $O(n)$.

$\Rightarrow O(n!) \approx O\left(\frac{n^n}{e^n}\right)$.

Better: $O(2^n \cdot n^2)$ via dynamic programming (cannot expect poly-time algo because TSP is NP-complete)

- The subproblems are: $\boxed{\text{answer is } \min_{j=2,\dots,n} \{C(\{2,\dots,n\}, j) + \ell(j,1)\}}$

  $C(S,j) =$ shortest path from 1 to $j \in S$ visiting each vertex in $S \subseteq \{2,\dots,n\}$ once

- Boundaries: $C(\{j\}, j) = \ell(1,j)$

- Recurrence: $C(S,j) = \min_{i \in S \setminus \{j\}} \{C(S-\{i\}, i) + \ell(i,j)\}$.

- Efficiency: compute in order of increasing $|S|$

  table has $O(2^n \cdot n)$ entries and each takes $O(n)$ to compute $\Rightarrow O(2^n \cdot n)$.