Lecture 5: Overview

- Dynamic Programming (Chapter 15)
- Programming Assignment 1 due next week
Case Study

[Diagram with numbers 2, 3, 4, 7, 5, 2, 2, 8 connected in a network pattern]
Assembly Line

- Each stage has a cost
- Cost to move from one line to another
- \(2^{\text{stage}}\) possible routes
- For large numbers of stages, this search space is too large
Structure of Optimal Solution

- If optimal solution contains station $S_{i,j}$ then the path from the beginning to $S_{i,j}$ must be optimal and the path from $S_{i,j}$ to the end must be optimal (can’t make up for a bad route)
- Idea: start at stage 1 and compute optimal cost/path, then compute at stage 2....
- $f_{i}[j]=\min\{f_{i}[j-1], f_{m}[j-1]+t_{m, (j-1)}\}$
Optimal Solution

- Can write this as a recursive program
  - Still exponential (re-evaluating the same expressions many times) - need to store computed values
- Does not track path - store path in array
- Iterative version given in book
- Runtime is Theta(n)
Matrix-Chain Multiple

- Matrix multiply is associative
  - \((AB)C = A(BC)\)
  - Computation cost can be different (dimensions differ)
- Problem: How to optimize placement of parentheses to minimize cost
Optimal Structure

• Consider the product: $A_1A_2...A_n$
• Let $A_{ij}$ denote the product $A_iA_{i+1}...A_{j-1}A_j$
• We want to choose $k$ such that computing $A_{ik}A_{k+1}j$ is the minimal way to compute $A_{ij}$
• Let $M(i,j)$ denote the cost of computing $A_{ij}$
• $M(i,j)=0$ if $i=j$ or $\min(M(i,k)+M(k+1,j)+p_{i-1}p_kp_j)$ for $i\leq k<j$ if $i<j$
Recursive Procedure

\[ M(i, j) \]
if \( i = j \)
    then return 0
else
    \[ \text{min} = M(i, i) + M(i + 1, j) + p_{i-1}p_ip_j \]
    for \( k \leftarrow i + 1 \) to \( j-1 \)
        if \( \text{min} > M(i, k) + M(k + 1, j) + p_{i-1}p_kp_j \)
            \[ \text{min} = M(i, k) + M(k + 1, j) + p_{i-1}p_kp_j \]
    return min
Problem

- This recursive implementation is expensive
- We need to cache $M(i,j)$’s instead of re-computing them
- Can compute from the bottom up
Longest Common Subsequence

• Consider the sequences $X=\{A, B, C, B, D, A, B\}$. A subsequence of $X$ is $Y=\{A, C, D, A, B\}$.

• Subsequences have a subset of the elements in order

• Formally, given $X=\langle x_1, x_2, \ldots, x_m \rangle$, another sequence $Z=\langle z_1, z_2, \ldots, z_k \rangle$ is a subsequence of $X$ if there exist a strictly increasing sequence $\langle i_1, i_2, \ldots, i_k \rangle$ such that for all $j=1, 2, \ldots, k$, $x_{i_j}=z_j$ where $n=i_j$.
Problem

- Given two sequence X and Y, what is the longest common subsequence
- Naive approach: compute all subsequences of X and Y, find the long sequence in the intersection
Structure of the solution

Let $X=\langle x_1, x_2, \ldots, x_m \rangle$ and $Y=\langle y_1, y_2, \ldots, y_n \rangle$ be sequences. Let $Z=\langle z_1, z_2, \ldots, z_k \rangle$ be a LCS of $X$ and $Y$.

1. If $x_m = y_n$ then $z_k = x_m = y_n$ and $Z_{k-1}$ is a LCS of $X_{m-1}$ and $Y_{n-1}$
2. If $x_m \neq y_n$, then $z_k \neq x_m$ implies that $Z$ is a LCS of $X_{m-1}$ and $Y$
3. If $x_m \neq y_n$, then $z_k \neq y_n$ implies that $Z$ is a LCS of $X$ and $Y_{n-1}$
Recursive Solution

c[i,j] = 0 if i=0 or j=0
  c[i-1, j-1] +1 if i,j>0 and x_i=y_j
  max(c[i,j-1], c[i-1,j]) if i,j>0 and x_i≠y_j
• Compute c[i,j] table bottom up
Optimal Binary Search Trees

- Want to optimize average search time for binary search trees
- Some searches may be more common
- Not all search terms may be in the tree - want to optimize for failed searches also
Problem

Input n keys $K = \langle k_1, k_2, \ldots, k_n \rangle$

$n+1$ dummy keys $D = \langle d_0, d_1, \ldots, d_n \rangle$

$d_0 < k_1 < d_1 < k_2 < \ldots < k_n < d_n$

Dummy key $d_i$ represents all searches between $k_i$ and $k_{i+1}$

Probability of search for $k_i$ is $p_i$

Probability of search for $d_i$ is $q_i$

$$\sum_{i=1}^{n} p_i + \sum_{i=0}^{n} q_i = 1$$
Cost

\[ T[\text{search cost for } T] = \sum_{i=1}^{n} (\text{depth}(k_i) + 1)p_i + \sum_{i=0}^{n} (\text{depth}(d_i) + 1)q_i = 1 \]
Structure

- Subtree will contain keys in contiguous rank \( k_i, \ldots, k_j \) for some \( 1 \leq i \leq j \leq n \).
- Dummy keys \( d_{i-1}, \ldots, d_j \) must be leaves.
- Optimal structure - both sub-trees are optimal.
- Cost increase of lowering subtree one level

\[
\begin{aligned}
w(i, j) &= \sum_{l=i}^{j} p_l + \sum_{l=i-1}^{j} q_l \\
e[i,j] &= q_{i-1} \quad \text{if } j = i-1 \\
&= \min\{e[i,r-1]+e[r+1,j]+w(i,j)\} \quad \text{if } i \leq j \text{ for } i \leq r \leq j
\end{aligned}
\]