Dynamic Programming

The most important algorithmic technique covered in CSE 417

Key ideas:
- Express solution in terms of a polynomial number of sub problems
- Order sub problems to avoid recomputation

Optimality Condition

Opt[ j ] is the maximum weight independent set of intervals I_1, I_2, ..., I_j

- Where p[ j ] is the index of the last interval which finishes before I_j starts

Algorithm

MaxValue(j) =
if j = 0 return 0
else
    return max( MaxValue(j-1), w_j + MaxValue(p[ j ]))

Worst case run time: 2^n
A better algorithm

\[
M[j] \text{ initialized to } -1 \text{ before the first recursive call for all } j
\]

\[
\text{MaxValue}(j) = \\
\quad \text{if } j = 0 \text{ return 0; } \\
\quad \text{else if } M[j] \neq -1 \text{ return } M[j]; \\
\quad \text{else } \\
\quad \quad M[j] = \max(\text{MaxValue}(j-1), w_j + \text{MaxValue}(p[j])); \\
\quad \quad \text{return } M[j];
\]

Iterative Algorithm

Express the MaxValue algorithm as an iterative algorithm

\[
\text{MaxValue }
\]

Iterative Algorithm

\[
\text{int[]} M = \text{new int}[n+1]; \\
\text{char[]} R = \text{new char}[n+1]; \\
M[0] = 0; \\
\text{for (int } j = 1; j < n+1; } j++ \\
\text{if } (v1 > v2) \\
\quad M[j] = v1; \\
\quad R[j] = 'A'; \\
\text{else } \\
\quad M[j] = v2; \\
\quad R[j] = 'B';
\]

Fill in the array with the Opt values

Opt[ j ] = max (Opt[ j − 1], wj + Opt[ p[ j ] ])

Computing the solution

Opt[ j ] = max (Opt[ j − 1], wj + Opt[ p[ j ] ])

Record which case is used in Opt computation

Optimal linear interpolation

\[
\text{Error} = \sum (y_i - ax_i - b)^2
\]
What is the optimal linear interpolation with three line segments

What is the optimal linear interpolation with two line segments

What is the optimal linear interpolation with $n$ line segments

Notation

• Points $p_1, p_2, \ldots, p_n$ ordered by x-coordinate ($p_i = (x_i, y_i)$)
• $E_{ij}$ is the least squares error for the optimal line interpolating $p_i, \ldots, p_j$

Optimal interpolation with two segments

• Give an equation for the optimal interpolation of $p_1, \ldots, p_n$ with two line segments

• $E_{ij}$ is the least squares error for the optimal line interpolating $p_i, \ldots, p_j$

Optimal interpolation with $k$ segments

• Optimal segmentation with three segments
  - $\min_{i,j} (E_{1j} + E_{ij} + E_{jn})$
  - $O(n^2)$ combinations considered
• Generalization to $k$ segments leads to considering $O(n^{k-1})$ combinations
Opt\textsubscript{k}[j] : Minimum error approximating \(p_1\ldots p_j\) with \(k\) segments

How do you express Opt\textsubscript{k}[j] in terms of Opt\textsubscript{k-1}[1],...,Opt\textsubscript{k-1}[j]?

**Optimal sub-solution property**

Optimal solution with \(k\) segments extends an optimal solution of \(k-1\) segments on a smaller problem

**Optimal multi-segment interpolation**

Compute Opt\textsubscript{k}[j] for \(0 < k < j < n\)

\[
\text{for } j = 1 \text{ to } n \\
\quad \text{Opt}[1, j] = E_{1,j};
\]

\[
\text{for } k = 2 \text{ to } n-1 \\
\quad \text{for } j = 2 \text{ to } n \\
\quad \quad t = E_{1,j} \\
\quad \quad \text{for } i = 1 \text{ to } j-1 \\
\quad \quad \quad t = \min(t, \text{Opt}[k-1, i] + E_{i,j}) \\
\quad \quad \text{Opt}[k, j] = t
\]

**Determine the solution**

- When Opt\textsubscript{k}[j] is computed, record the value of \(i\) that minimized the sum
- Store this value in an auxiliary array
- Use to reconstruct solution

**Variable number of segments**

- Segments not specified in advance
- Penalty function associated with segments
- Cost = Interpolation error + \(C \times \#\text{Segments}\)

**Penalty cost measure**

\[
\text{Opt}[j] = \min(E_{1,j}, \min_i(\text{Opt}[i] + E_{i,j} + P))
\]