Dynamic Programming

- Weighted Interval Scheduling
- Given a collection of intervals I₁,...,Iₙ with weights w₁,...,wₙ, choose a maximum weight set of non-overlapping intervals

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

Computing the solution

```
Opt[j] = max (Opt[j - 1], wᵢ + Opt[p[j]])
```

Record which case is used in Opt computation

```
2 4 9 9 9 16 16
B B B A A B A
```

Optimality Condition

- Opt[j] is the maximum weight independent set of intervals I₁, I₂, ..., Iⱼ
- Opt[j] = max( Opt[j - 1], wᵢ + Opt[p[j]])
  - Where p[j] is the index of the last interval which finishes before Iⱼ starts

Announcements

- Reading:
  - 6.1-6.2, Weighted Interval Scheduling
  - 6.3 Segmented Least Squares
  - 6.4 Knapsack and Subset Sum
Optimal linear interpolation

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
  - \( \min_{i,j} (E_{1,j} + E_{i,j} + E_{j,n}) \)
  - \( O(n^2) \) combinations considered

• Generalization to k segments leads to considering \( O(n^{k-1}) \) combinations

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 \( \text{Opt}[k,j] \) for \( 0 < k < j < n \)

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

Determining the solution

• When \( \text{Opt}[k,j] \) is computed, record the value of \( i \) that minimized the sum
• Store this value in a 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

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