Algorithm Design Techniques

- Induction (Self-Reduction)
  - Divide and Conquer
  - Dynamic Programming continued

Outline

- Hum-Soc reading problem
- String edit problem
- Neat printing problem

Hum-Soc reading

- You are assigned books $b_1, \ldots, b_n$ to read in $K$ days (typically $K < n$)
  - The books have to be read in order
  - You must finish any book you start on the same day
  - Book $b_i$ has $p_i$ pages
- Objective: minimize the maximum number of pages you need to read in any day

Hum-soc reading

Definition:
Let $M(i,j)$ denote the maximum number of pages you have to read in any day in an optimal solution to the problem of reading books $b_1, \ldots, b_i$ in $j$ days

- Base Case: $j=1, i=0\ldots n$
  $M(i,j)=\sum_{m=1}^{i} p_m$
- Inductive Step: $j=2\ldots k, i=0\ldots n$
  $M(i,j)=\min_{m=0\ldots i} \max(M(m,j-1), \Sigma_{m=1}^{i} p_j)$
Hum-soc reading

- Analysis
  - Compute $M(i,j)$ for $i=0 \ldots n$, $j=1 \ldots K$
  - Each computation can be done in $O(n)$
  - Overall running time is $O(n^2K)$

String Edit

- Input: Strings $X=x_1x_2 \ldots x_n$ and $Y=y_1y_2 \ldots y_n$
- Output: Edit distance between $X$ and $Y$
- Edit operations are insert/delete, match and substitute. Insert/deletes cost $c_1$ and substitutes cost $c_2$. Matches are free.
- Edit distance is the minimum cost of edits needed to transform $X$ into $Y$.

String Edit

- Let $D(i,j)$ denote the edit distance between $x_1 \ldots x_i$ and $y_1 \ldots y_j$

String Edit

- Base case: $D(i,j)$ where $i=0$ or $j=0$
  - $D(0,j) = c_j$
  - $D(i,0) = c_i$
- Inductive step: $D(i,j)$ where $i>0$ and $j>0$
  - If $x_i=y_j$:
    - $D(i,j) = \min(c_1D(i-1,j), c_1 + D(i,j-1), c_2 + D(i-1,j-1))$
  - Else:
    - $D(i,j) = \min(c_1D(i-1,j), c_1 + D(i,j-1), c_2 + D(i-1,j-1))$

String Edit

- Analysis:
  - $D(i,j)$ takes constant time to compute given $D(i-1,j)$, $D(i,j-1)$ and $D(i-1,j-1)$.
  - $D(i,j)$ is computed for $i=0 \ldots n$, $j=0 \ldots m$
  - The running time is $O(nm)$

Neat Printing

- We want to print words $w_1 \ldots w_n$
- Word $w_i$ is $c_i$ characters long
- Adjacent words on a line must be separated by a blank
- $M$ characters on a line must be separated by a blank
- No hyphenation is allowed
- We want to minimize the sum of the squares of the trailing blanks on the printed lines.
Neat Printing

• Definition 1:
  – For $j > i$ let $B(i, j)$ denote the number of trailing blanks when words $w_i \ldots w_j$ are printed on a single line provided the words fit on a single line. If the words don’t fit then $B(i, j) = \infty$.
  – Note: if $B(i, j)$ is finite then $j - i + 1 \leq M/2$, which is constant.
  – We can compute $B(i, j)$ for $i, j$ such that $j - i + 1 \leq M/2$ in $O(n)$ time.

• Definition 2:
  – Let $C(i)$ denote the minimum cost of printing words $w_1 \ldots w_i$.

  • Base Case: $i = 0$
    – $C(0) = 0$
  • Inductive Step: $i > 0$
    – $C(i) = \min_{j < i, j > 0, j \geq i - M/2} C(j) + B(j + 1, i)$

• Analysis:
  – $C(i)$ takes $O(1)$ time to compute given $B(i, j)$ for $j - i + 1 \leq M/2$
  – We need to compute $C(i)$ for $i = 0, \ldots, n$
  – The running time is $O(n)$