Dynamic Programming

August 23, 2025

Review of Coin Exchange

Input

- n coins with values $p_1, \ldots, p_n$;
- a number t.

Question

Could we find some coins with a total value of t?

Recurrence Relation

Let $F[i, j]$ be a Boolean array such that $F[i, j] = 1$ if and only if we can get value j by $p_1, \ldots, p_i$.

- For all $i \in [n]$, $F[i, 0] = 1$;
- For all $i \in [n], j \in [t]$,
  $F[i, j] = 1$ if and only if $F[i-1, j] = 1$ or $F[i-1, j-p_i] = 1$
- $F[n, t]$ is the solution.

Pseudo-code for Coin Exchanges

- Input $p_1, \ldots, p_n, t$;
- For $i = 1$ to n
  $F[i, 0] = 1$
- For $i = 1$ to n
  For $j = 1$ to t
  If $(j \geq p_i)$ Then $F[i, j] = (F[i-1, j]$ or $F[i-1, j-p_i])$
  If $(j < p_i)$ Then $F[i, j] = F[i-1, j]$
- Output $F[n, t]$

Review Edit Distance

Question

Number of times we need to insert, remove, or rewrite characters to make x and y identical. More specifically, inserting/deleting a character costs A, and overwriting a character costs B.

Recurrence Relation

Let $D[i, j]$ be the edit distance of $(x_1 x_2 \ldots x_i)$ and $(y_1 y_2 \ldots y_j)$

- For all $i \in [n]$, $D[i, 0] = A \cdot i$;
- For all $j \in [m]$, $D[0, j] = A \cdot j$;
- ...
- $F[n, t]$ is the solution.

Pseudo-code for Minimum Edit

- Input x and y;
- Let $n = |x|$ and $m = |y|$;
- For $i = 1$ to n
  $D[i, 0] = A \cdot i$
- For $j = 1$ to m
  $D[0, j] = A \cdot j$
- For $i = 1$ to n
  For $j = 1$ to m
  If $x_i = y_j$
  Then $D[i, j] = \min(D[i-1, j-1], A + D[i-1, j], A + D[i, j-1])$
  Else
  $D[i, j] = \min(D[i-1, j-1] + B, A + D[i-1, j], A + D[i, j-1])$
- Output $D[n, m]$

Review of Weighted Interval Scheduling

Input

- n tasks;
- a start time s_i for each task i;
- a finish time f_i for each task i;
- a weight w_i for each task i

Desired output

A set S of non-overlapping tasks maximizing the total weight $\sum_{i \in S} w_i$

Pseudo-code for Weighted Interval Scheduling

- $A[0] = 0$
- For $t = 1$ to t_{max}
  - $A[t] = A[t-1]$
  - For all i with $f_i = t$
    - If $w_i + A[s_i - 1] \geq A[t]$ Then $A[t] \leftarrow w_i + A[s_i - 1]$
- Output $A[t_{max}]$

Knapsack Problem

Input

- n items;
- size s_i for each item $i \in [n]$
- value v_i for each item $i \in [n]$
- the size S of the knapsack.

Output

The maximum value we can get of using size S.

Recurrence Relation

Recurrence Relation

Recurrence Relation

An Important Step of Writing Solutions

Explicitly write down your ideas!

Thanks!