Algorithms for Programming Contests SS20 - Week 08

Mikhail Raskin, Tobias Meggendorfer, Stefan Jaax, A.R.Balasubramanian, Christoph Welzel

June 16, 2020

A: Making Change - Sample Solution

Problem

Find the fewest number of coins to make a certain amount of change.

A: Making Change - Sample Solution

- $a[i]$: fewest number of coins needed to pay a value of i
- $a[i] = \min_{1 \le j \le n} \{a[i - v_j] + 1 \text{ if } i - v_j \ge 0\}$
  with v_j the value of the j-th coin/note
- Return $a[c]$

B: Bracelets - Sample Solution

- Longest common subsequence
- $a[i][j]$: longest increasing subsequence using the first i characters of one word and the first j characters of the other word
- $a[i][j] = \max\{a[i-1][j-1]+1 \text{ if } word1[i] = word2[j], a[i-1][j], a[i][j-1]\}$
- Dimension may be reduced by one by saving only one column at a time
- Return $a[|word1|][|word2|]$
- Take maximum of all possible rotations and reflections

C: Packing Cases - Sample Solution

Problem

Can Lea stack boxes high enough to reach the top counter in her kitchen?

C: Packing Cases - Sample Solution

- Each box may only be used two times $\Rightarrow$ always enough boxes.
- Add each rotation of each box (six/three if properly sorted).
- Sort the boxes by decreasing area on the top, then use DP.
- $a[i]$: maximum height using boxes 1 to i with box i on top.
- $a[i] = \max_{1 \leq j < i}\{a[j] + h_i \text{ if box } i \text{ fits on top of box } j\}$.
- Return $\max_i a[i] \geq h$.

D: Poker - Sample Solution

Problem

What is the maximum amount of money Lea can win without attending overlapping poker tournaments?

D: Poker - Sample Solution

- Weighted Interval Scheduling Problem
- Sort intervals by finishing time.
- Use dynamic programming over the intervals (1-based).
- Let $p(i)$ be the index of the latest finishing interval that is still compatible with interval i.

$$\begin{aligned} \texttt{dp}[0] &= 0 \\ \texttt{dp}[i] &= \max(\texttt{dp}[i-1], \texttt{dp}[p(i)] + \mathit{weight}(i)) \end{aligned}$$

- Compute $\texttt{dp}[n]$.

E: Escaping the Paradox - Sample Solution

- Generate bidirectional graph for future Lea
- Run Dijkstra to find out how much time she needs to reach each cave/the surface
- Remove Edges that Lea cannot take
- Use dynamic programming to find a way with maximum value that can reach the surface before future Lea can.
- If you reach the surface before future Lea does, she cannot have met you on the way.