Day 1: Problem Analysis

Version 0.9: All problems except for D and G

Maxim Buzdalov

April 14, 2015

Preliminaries

- Contest origin – Norwegian Collegiate Programming contest
  - NCPC 2005: B, C, F, G $\rightarrow$ A, B, C, D
  - NCPC 2006: A, B, D, F, G $\rightarrow$ E, F, G, H, I
  - NCPC 2007: D, E, F $\rightarrow$ J, K, L
  - hardest problems from each set

Problem A. Funny Games

Statement

- Planet of initial size X
- K weapons, i-th reduces the planet by a factor of F_i
- Two players make moves (applying a weapon) in turns
- Who made the planet less than 1, wins

Problem A. Solution

Solution idea

- This is a game on a graph
- Vertices = sizes $\rightarrow$ exponential size, TL
- Vertices = size intervals!
  - winning/losing intervals
  - a point from a winning interval = a winning point

Problem A. Implementation

My implementation

- Construct intervals from 1 above
  - the first winning interval: $(1; \frac{1}{F_{\min}})$
- Consider them as a collection of winning intervals
  - overlapping winning intervals can be united
- When a winning interval ends
  - if a losing interval starts (at t), add beginnings of winning intervals: $\frac{t}{F_i}$
- When a winning interval begins
  - if a losing interval ends, add endings of winning intervals

Problem A. Implementation

Further details

- A priority queue to store interval beginnings and endings
- Running time: $O(Z \log Z)$ where Z is the number of intervals
- What is the bound on Z?
  - I don't know :(
  - Somehow connected with X, maximum $F_i \leq 0.9$ and K

Problem B. Nullary Computer

Statement

- Given a computer with 26 registers and simplistic instructions
- Sort first 24 registers
- Size limit: 5432 instructions

Solution

- A comparator for a and b: `a(Yb(Z)a)z(Az)y(By)`
- Bubble sort network: $n(n-1)/2$ comparators, size 5244

Problem C. Worst Weather Ever

Statement

- Data: in year Y_i it was R_i mm of rain
- Queries: does year X have the most rain since year Y?
  - $R(Y) \geq R(X)$
  - if $Y < Z < X$ then $R(Z) < R(X)$
- Answers: true, false, maybe

Problem C. Solution

My (plain) segment tree solution

- Segment tree for maximum only on known years
- (X, Y) query processing:
  - Find closest known years:
    - $Y_i \leftarrow$ upper bound for Y
    - $X_i \leftarrow$ lower bound for X
  - Get a maximum from a segment tree (without X and Y)
  - Check the statements
    - all years are known: $X - Y = X_i - Y_i$

Problem D. Kingdom

Problem E. Shoot-out

Statement

- N cowboys, i-th shoots dead with probability P_i
- Shoot in turn using optimal strategies until only one remains
- What are the probabilities of remaining the only one?

Problem E. Solution

Solution

- Dynamic programming: $A(M, i, j)$ is the probability of the cowboy j to remain if there is a set of M living cowboys and the cowboy i shoots now
- $A(M, \_, j)$ have a circular dependency loop (i.e. all cowboys may miss), so should be evaluated at once
- Complexity: $O(2^N \cdot N^3)$

Problem F. Tour Guide

Statement

- N oldies each move along a straight line
- You need to run onto each of them and motivate to go to $(0, 0)$
- Minimize the time when everyone is at $(0, 0)$

Solution

- Test all $N!$ permutations
- Act greedily

Problem G. Jezzball

Problem H. Traveling Salesman

Statement

- Countries: broken closed polylines in space
- Some countries have common borders (polyline segments)
- Find minimum number of border crossings to get from country A to country B

Solution

- Build a graph (vertices = countries, edges = common borders)
- Find a path length (BFS)

Problem I. Whac-a-Mole

Statement

- $N \times N$ field with moles appearing
- Hammer moves: straight line movements from integer point to integer point
- Maximize number of whacked moles

Solution

- Dynamic programming: $A(x, y, t)$ is the answer at the end of t when finishing at (x, y)
- Can get outside of $[0; N-1] \times [0; N-1]$!

Problem J. Copying DNA

Statement

- Source DNA string S
- Target DNA string T
- Operations
  - get substring from S, optionally reverse, stick into T
  - get substring from partially built T, optionally reverse, stick into T
- Find minimum number of operations to build T

Problem J. Solution

My solution

- Precompute $P(s, t)$
  - if $T[s, t]$ is as a (reversed) substring of S
- Precompute $Q(s_1, s_2, l)$
  - if $T[s_1, s_1 + l]$ is the (reversed) same as $T[s_2, s_2 + l]$
- Dynamic programming: $A(M)$ – the minimum number of operations to construct a subset M of positions from T
  - test all $U = [s, t]$ such that $M \cap U = \emptyset$
  - find using P and Q if you can construct U

Problem K. Circle of Debt

Statement

- A, B, C owes some money to each other
- Each of them has money units of nominations: 100, 50, 20, 10, 5, 1
- Find minimum number of money unit movements to clear debts

Problem K. Solution

Idea

- For each nomination, possible movements are $(X \to Y)$; $(X \to Y, Z)$; $(X, Y \to Z)$

Solution

- Dynamic programming: $D(x, y, k)$ is the minimum number of moves to achieve x money for A and y money for B after exchange of k smallest nominations
- Almost all nominations are multiples of each other. Don't check all x, y!

Problem L. Full Tank?

Statement

- Graph: vertices are fuel stations with price p_i per unit, edges are roads where you spend d_i units of fuel
- Find the cheapest way to get from A to B

Solution

- Author solution: Dijkstra with heap on implicit graph
  - If one quits Dijkstra when target is hit first: 0.4 seconds
  - Otherwise, 2.8 seconds