ITMO University
Peking University
Training Camp

ITMO大学北京大学训练营

Day 01: Problem Analysis

Maxim Buzdalov

ITMO University,
St. Petersburg, Russia

26.02.2014

Problem Origin

- Problem compilation from SWERC contests
  - 2010
  - 2011
  - 2012

Problem A. Periodic Points

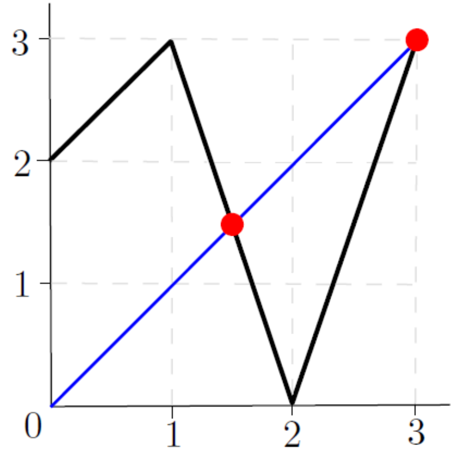
- Given a function f from $[0;m]$ to $[0;m]$
  - values are given for $f(0), f(1), \ldots, f(m)$
  - f is piecewise linear in intervals [0;1], [1;2], …
- $f^n(x) = f(f(\ldots(f(x))\ldots))$
- Find the number of solutions to $f^n(x) = x$
  - either “Infinity”
  - or the number of solutions modulo *mod*

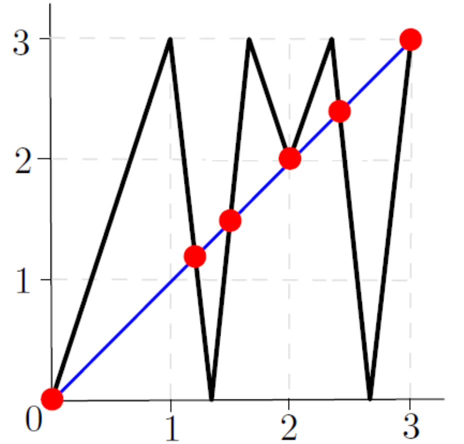
Ideas

- If the answer is finite for f, it is finite for f^n
  - will be proven later by construction
- The answer is "Infinity" for f $\Leftrightarrow$
  the answer is "Infinity" for f^n
- Check if for any interval $[k; k+1]$ holds $f(x) = x$
  - if this is true, the answer is "Infinity"
  - otherwise, it is not.

Ideas

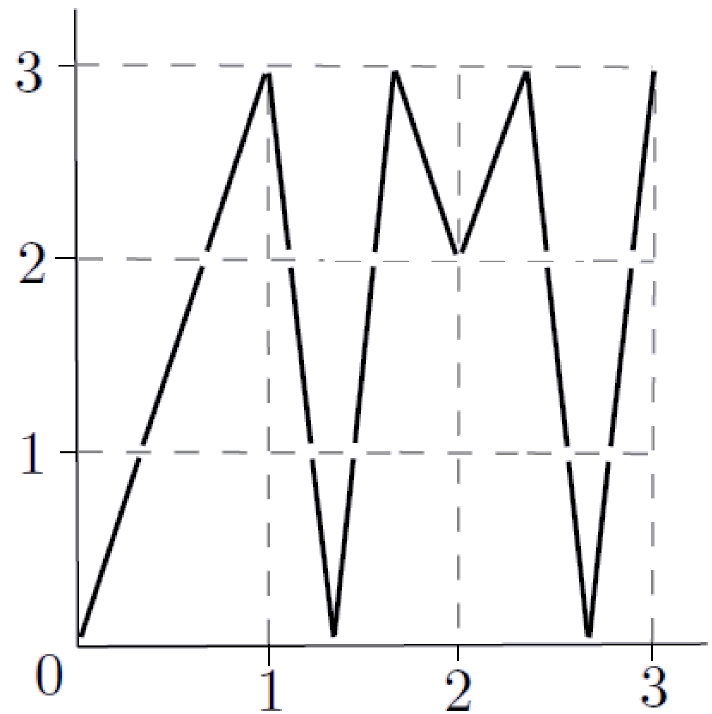
- f^n is piecewise linear
  - but the number of pieces may grow exponentially
- number of solutions to $f^n(x) = x$ in $[0;m]$ is the number of intersections of the f^n plot with the diagonal of $[0;m]$x$[0;m]$





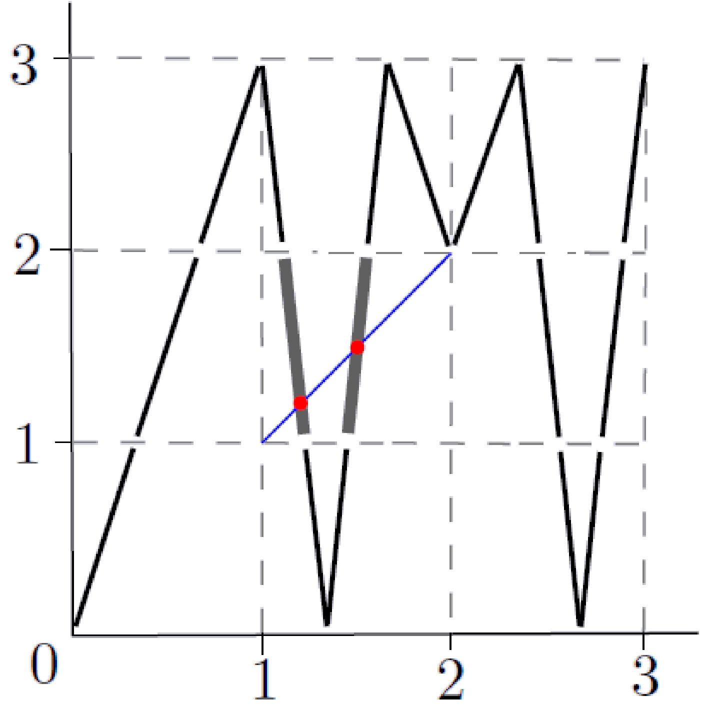
Ideas

- f^n consists of intervals going from $y = a$ to:
  - $y = a - 1$
  - $y = a + 1$



Ideas

- Intersections between f^n and the diagonal in the interval $(k; k + 1)$ = sub-intervals going from $y = k$ to $y = k + 1$ or vice-versa.



Solution

- A_{ij} = number of subintervals between $y = j$ and $y = j+1$ contained in the graph of f in $(i; i+1)$
- $(A^{n})_{ij}$ = number of subintervals between $y = j$ and $y = j+1$ contained in the graph of f^{n} in $(i; i+1)$
  - Use fast exponentiation
- The answer for non-integer points is $\mathrm{Trace}(A^n)$
- + For all x = 0, 1, …, m check if $f^n(x) = x$

Problem B. Palindromic DNA

- Transform a given DNA string (chars: A, G, C, T)
  - several pairs of characters should be equal
  - each character can be unmodified or changed:
    - cyclic order: A → G → C → T
    - A → G, T
    - G → A, C
    - C → G, T
    - T → C, A
  - cannot modify consequent characters

Observation

- For each pair of positions that should be equal:
  - if s[i] = s[j], need to apply same operation to both;
  - if dist(s[i]; s[j]) = 1, exactly one of them has to change (in the right direction);
  - if dist(s[i]; s[j]) = 2, both need to change in reverse directions.

Solution: 2SAT

- Variables:
  - x_i – s[i] is changed
  - y_i – s[i] is increased in the cyclic order
- For all pairs of positions to be equal:
  - s[i] = s[j] → $(x_i = x_j)$ & $(y_i = y_j)$ → $(!x_i \mid x_j)$ & $(!x_j \mid x_i)$ & $(!y_i \mid y_j)$ & $(!y_j \mid y_i)$
  - …
- No two consecutive positions are changed:
  - $(!x_1 \mid !x_2)$ & $(!x_2 \mid !x_3)$ & …
- Dependencies of y_i on x_i:
  - $(x_1 \mid !y_1)$ & $(x_2 \mid !y_2)$ & …

Problem C. Jumping Monkey

- There is a graph and a monkey in an unknown vertex
- You shoot in a vertex
  - the monkey is killed, or
  - the monkey moves using a graph edge
- What is the shortest sequences of shoots to kill the monkey for sure?

Solution

- Store the vertex set where the monkey can be
  - recalculation: O(N) shoots, O(N^2) moves for each shoot, O($2^N N^3$) in total
- Bitmasks for possible moves – O($2^N N^2$) in total
- For current state $\{V_1, V_2, \ldots, V_k\}$, compute the neighbors of the sets $\{V_1\}$, $\{V_1, V_2\}$, … and the sets $\{V_k\}$, $\{V_k, V_{k-1}\}$, … Use it for everything else
  - O(N) operations for a state, O($2^N N$) in total

Problem D. 3-sided Dice

- There are three dice A, B, C with the given probabilities for sides 1, 2, 3
- Is it possible to simulate the given die using the dice A, B and C?
  - by choosing fixed **nonzero** probability for each die

Solution

- Dice are points in 2D
  - coordinates the probabilities for sides 1 and 2
- Dice A, B, C → a triangle
  - the triangle is degenerate (a segment) →
    test if the given die is strictly inside the segment
  - the triangle is not degenerate →
    test if the given die is strictly inside the triangle

Problem E. Assembly Line

- Given a list of pieces of different types
- Assembly table
  - given two pieces of type T_i and T_j
  - it takes C_{ij} time to assemble them
  - the resulting piece is of type R_{ij}
- What is the optimum time to assemble all the pieces?
  - cannot change their order

Solution

- Dynamic programming
  - $D_{i\,i\,k} = 0$ if the type of *i*-th piece is *k*
  - $D_{i\,i\,k} = \infty$ if the type of *i*-th piece is not *k*
  - $D_{i\,j\,k} = \min \{D_{i\,m\,a} + D_{m\,j\,b} + C_{a\,b} \mid R_{a\,b} = k, i \leq m < j\}$

Problem F. Alphabet Soup

- There are *P* points on a circle, each may have *S* possible types
- Compute the number of assignments of types to points, if two assignments which can be rotated one into another are equal
  - modulo 100000007

Solution: Burnside lemma

- Compute the smallest rotation that move all points onto some other points
  - string of angle differences *D: D*[*i*] = *P*[*i*+1] – *P*[*i*]
  - find an entity of *D* into *DD*
    which is not a prefix or a suffix
- Full circle rotation → S^P mod 100000007
- Rotation by *k* → $\frac{k}{P}\sum_{i=0}^{\frac{P}{k}-1} S^{k\,\gcd\left(i,\frac{P}{k}\right)}$

Problem G. Game, Set and Match

- Compute the probability of winning a game, a set and a match in tennis if each point is won with the probability of P

Rule simplification

- The simple form of rules
  - To win a game, you need to win 4 points by a margin of 2.
  - To win a set, you need to win 6 games by a margin of 2, except if the score reaches 6–6 (tiebreak).
  - To win the tiebreak, you need to win 7 points by a margin of 2.
  - To win the match, you need to win 2 sets.

Subproblems

- $a(p)$ – the probability of being the first to win two consecutive points when tied, and each point is won with the probability of p
- $b(n, p_1, p_2)$ – the probability of winning n points with a margin of two if each point is won with the probability of p_1 and when the score is n–n, the probability of winning is p_2.

Winning probabilities

- $Pr[\text{game}] = b(4, p, a(p))$
- $Pr[\text{tiebreak}] = b(7, p, a(p))$
- $Pr[\text{set}] = b(6, Pr[\text{game}], Pr[\text{tiebreak}])$
- $Pr[\text{match}] = r^2 + 2r(1 - r)$ where $r = Pr[\text{set}]$

Computing $a(p)$

- With the probability of p^2 we win
- With the probability of $(1-p)^2$ we lost
- With the probability of $2p(1-p)$ we return to the initial state
- $a(p) = p^2 + 2p(1-p)\, a(p)$
- $a(p) = \frac{p^2}{1-2p(1-p)}$

Computing $b(n, p_1, p_2)$

- $f(i, j)$ –probability of winning with score $i - j$.
  - $f(i, j) = 1$ if $i \geq n$ and $i \geq j + 2$
  - $f(i, j) = 0$ if $j \geq n$ and $j \geq i + 2$
  - $f(n, n) = p_2$
  - $f(n, n - 1) = p_1 + (1 - p_1)\mathrm{f}(n, n)$
  - $f(n - 1, n) = p_1 f(n, n)$
  - $f(i, j) = p_1 f(i + 1, \mathrm{j}) + (1 - p_1) f(i, j + 1)$
- $b(n, p_1, p_2) = f(0, 0)$

Problem H. Non-negative Partial Sums

- Count the number of cyclic shifts of the given array such that all partial sums are non-negative
  - $a[i]$ – the input array
- Compute in linear time:
  - $A[i]$ – sum of prefix of length i, $B[i]$ – of suffix
  - $C[i] = \min\{j \le i\} A[j]$
  - $D[i] = \min(a[i], a[i] + a[i + 1], \ldots)$
- Answer: count $\{i \mid D[i] \ge 0; B[i] + C[i - 1] \ge 0\}$

Problem I. Beehives

- Problem: find a shortest cycle in the undirected graph
- Solution:
  - start a BFS from each vertex
  - stop when you find an already visited node

Problem J. RNA Secondary Structure

- Given RNA – a string of {A, C, G, U} encoded in RLE
- A can pair with U
- C can pair with G
  - no more than K times
- Pairings should not overlap
- Find how to make maximum number of pairings

Solution

- Consider the first and last chains of equal symbols
  - unpairable – cut most of the chains
  - pairable – make pairings
- What remains?
  - inner chains of equal symbols
  - small pieces of first and last chains
- Run dynamic programming
  - A[L][R][C] – number of pairings from symbol L to symbol R when at most C pairs C-G can be paired

Problem K. The Moon of Valencia

- Given a graph with costs on vertices and on edges
- You need to find a path starting and ending in given times at given vertices with cost equal to the given one.

Problem K. The Moon of Valencia

- SWERC judges say…
  - the problem can be solved using A* algorithm
  - the priority heuristic is $|S^* - G + H|$
    - S^* – the required cost
    - G – the current cost
    - H – the distance to the target in the path graph

Thank you!

Thank you for your attention!