Day 2: Problem Analysis

07.05.2014

Problem A. Balance

Problem A. Balance

Problem statement

- You are given n coins, it's known that one of these coins is fake, it is different in weight, but it's unknown if the fake coin is lighter or not
- You have a balance which allows you to compare two groups of coins. You are to determine the fake coin in such a way that the number of weighings in the worst case is minimum possible

Solution

- Suppose you know, if the fake coin is lighter or not
- Then solution would be the following:
  - Take two groups of $\lfloor \frac{n}{3} \rfloor$ or $\lceil \frac{n}{3} \rceil$ coins
  - Then after weighing you know, which of three groups contains fake coin
  - Solve problem for this group
- The number of weighings you need is $\lceil \log_3 n \rceil$

Solution

- When we don't know the type of fake coin, we only know the subset z of genuine coins
- What can we do?

Move 1

- Let's try to weigh coins we know nothing about: we choose $2x$ coins
- If their weights are the same, then $n - 2x$ coins left, and $z + 2x$ coins are genuine for sure
- If different and $z \geqslant x$ we can weigh x real coins with one group of x coins we tried just before and learn the type and group of fake coin

Move 2

- Another move we can make is weigh group from known coins with group of unknown coins
- If they are different then we learn the group and the type of fake coin
- If not $n - x$ coins left

Solution

- Dynamic programming approach: $f(n)$ — number of weighings one needs to make to find fake coin
- Try make the moves described above to make transitions
- Memoize the optimal moves made to get the answer

Problem B. Cipher

Problem B. Cipher

Problem statement

- You are given $(H-1) \times (W-1)$ matrix B, that is made of $H \times W$ matrix A as $B_{ij} = A_{ij} + A_{i+1,j} + A_{i,j+1} + A_{i+1,j+1}$
- Find any A that produces given B or say that there is no such

Solution

- Suppose you already know A_{i0} and A_{0j}
- Then you can find any other $A_{ij} = B_{i-1,j-1} - A_{i-1,j} - A_{i,j-1} - A_{i-1,j-1}$
- If you look more carefully and represent each A_{ij} in terms of A_{i0} and A_{0j}, then you can see that $A_{ij} = \sum\limits_{x<i,y<j} (-1)^{y-j+x-i} B_{xy} + (-1)^{i+j} A_{00} + (-1)^{i} A_{0j} + (-1)^{j} A_{i0}$

Solution

- Let's split the problem to two cases: $A_{00} = 0$ and $A_{00} = 1$
- The only two summands not known in this formula are A_{0j} and A_{i0} and allowed values of A_{ij} are only 0 and 1
- So you have $H + W - 2$ boolean variables and for every A_{ij} there are restrictions for A_{i0} and A_{0j}
- Solve 2-SAT problem

Problem C. Hyperboloid Distance

Problem C. Hyperboloid Distance

Problem statement

- You are given two points on $x^2 + y^2 - z^2 = 1$ hyperboloid
- Find the distance between these points on hyperboloid surface

Solution

- Since allowed error is 0.1, one can make a grid on hyperboloid and find the shortest path in this graph
- Not any grid would work. The more optimizations you make, the less error solution would have

Some optimizations

- Make point as pair of angle and z-coordinate
- You can rotate hyperboloid, so one of the angles is 0 and the other is less that π, and path doesn't contain angles greater
- You can't say that about z-coordinate, sometimes it's optimal to go closer to $z = 0$
- Try eight directions from every point, not only four
- You can use your geometry skills to find the distance between neighbouring points more accurately

Problem D. Real Fun

Problem D. Real Fun

Problem statement

- You are given points on the plane
- You need to find minimal d, so that there are three $d \times d$ squares with sides parallel to coordinate axes that every point is covered by at least one of the squares

Solution idea

- Define m_x, M_x are minimal and maximal x-coordinate of given points respectively
- Define m_y and M_y similarly
- Main idea: solution always contains square, angle of which is in one of four points: (m_x, m_y), (M_x, m_y), (m_x, M_y), (M_x, M_y)
- Why? Suppose no square covers one of these points, so every square covers at most one of these: m_x, M_x, m_y, M_y
- But we have only three squares, so pigeon hole principal says that the statement above is correct

Solution

- Binary search for d
- Now have to check, if there are three squares to cover
- $f(s, A)$ — checks, if there are s squares to cover all points from A
- if $s = 0$ check whether A is empty
- Try all four corner points to make square S, and check $f(s - 1, A \setminus S)$

Problem E. Hippopotamus

Problem E. Hippopotamus

Problem statement

- You are given n, m and k
- You have to find how many different n-bit strings, that every m consecutive bits contain at least k 1-s

Solution

- Dynamic programming approach
- $f(i, S)$ — is number of i-bit strings, that last m bits are look like S
- The number of states and transitions in DP is $O(n2^m)$.

Problem F. Ice-cream Tycoon

Problem F. Ice-cream Tycoon

Problem statement

- You are given queries of two types:
  - There is a sell of k items each at price of c
  - A request to buy k items having m money, if the cheapest k items now being sold cost not more than m, then the trade happens
- The number of queries doesn't exceed 10^5

Solution

- You need to have data structure that contains sells as pair (c, k) sorted by c
- When new sell arrives, you have to be able to add it
- When there is a new buy, you have to check how many smallest pairs are there to make at least k items
- Check whether cost exceeds m and remove sells one by one
- You can use any binary search tree to make every query run in $O(\log n)$

Alternative solutions

- To make things easier to implement, you can solve the problem using segment tree, reading all queries first
- One could use $O(\log^2 n)$ solution for single query using Fenwick tree and binary search, which is very easy to implement

Problem I. Shortest Paths

Problem I. Shortest Paths

Problem statement

- You are given directed graph with non-negative edge cost
- You need to find k shortest paths between two vertices

Solution

- Author's solution was based on paper written by David Eppstein, which describes solution of this problem
- Link to the paper