Codeforces #172 Tutorial

xiaodao

Contents

1 Problem 2A. Word Capitalization

Brief Description

Capitalize a given word.

Tags

implementation, strings

Analysis

As the first problem in the problemset, it intended to be simple so that any one could solve it. Just implement what the problem said under your favorite language.

Negative Example

h6363817, A.nahavandi, Sigizmynd, addou: There are always some competitors started write down the code before they got comprehensive understanding on the statement and the restriction. There is more than one single case and I am not going to list all of them here.

LordArian, Cyber.1: Another part of our compatriots are still unfamiliar with some basic builtin-function. They are going to reinvent the wheel and it will definitely cost more of their time on implementation. And at the same time this means more possibility of making mistakes.

Suggested Code

astep, navi, shloub, Hohol: For situation like this, we believe that **the shorter the better**! There are the shortest codes on **Ruby**, **Perl**, **Python** and **Haskell**. And I hope there could be at least one of them to your taste.

2 Problem 2B. Nearest Fraction

Brief Description

Find the nearest fraction to fraction $\frac{x}{y}$ whose denominator is no more than n.

Tags

brute-force, greedy, math, number theory

Analysis

This problem can be solved by a straight forward method since we are at Div II, simply enumerate all possible denominator, and calculate the nearest fraction for each denominator. As you would expect, we can do better. Actually this is the **limit_denominator()** method implemented by the Python fractions module. Since it is written in **Python** and you can just look at the source code to find the truth behind it if you are interested.

Here I'd like to quote the top comment of the snippet:

```
# Algorithm notes: For any real number x, define a *best upper
# approximation* to x to be a rational number p/q such that:
#
#   (1) p/q >= x, and
#   (2) if p/q > r/s >= x then s > q, for any rational r/s.
#
# Define *best lower approximation* similarly.  Then it can be
# proved that a rational number is a best upper or lower
# approximation to x if, and only if, it is a convergent or
# semiconvergent of the (unique shortest) continued fraction
# associated to x.
#
# To find a best rational approximation with denominator <= M,
# we find the best upper and lower approximations with
# denominator <= M and take whichever of these is closer to x.
# In the event of a tie, the bound with smaller denominator is
# chosen.  If both denominators are equal (which can happen
# only when max_denominator == 1 and self is midway between
# two integers) the lower bound---i.e., the floor of self, is
# taken.
```

Negative Example

clonoboyka, somay.jain: There were some competitor have used strange algorithms that their complexity is hard to evaluate, and others got into trouble with precision issues. I hope this problem could give you a lesson, they are just some rules that everybody need to know.

Suggested Code

wafrelka: A clean **C++** implementation for our brute-force algorithm during the contest. You can see wafrelka were using 64-bit multiplication against float division to avoid precision issues.

havaliza: A **Python** code which using the **limit_denominator()** method as we mentioned above. As you can see here, choosing a suitable language sometimes may greatly decrease the amount of the code.

3 Problem A. Rectangle Puzzle

Brief Description

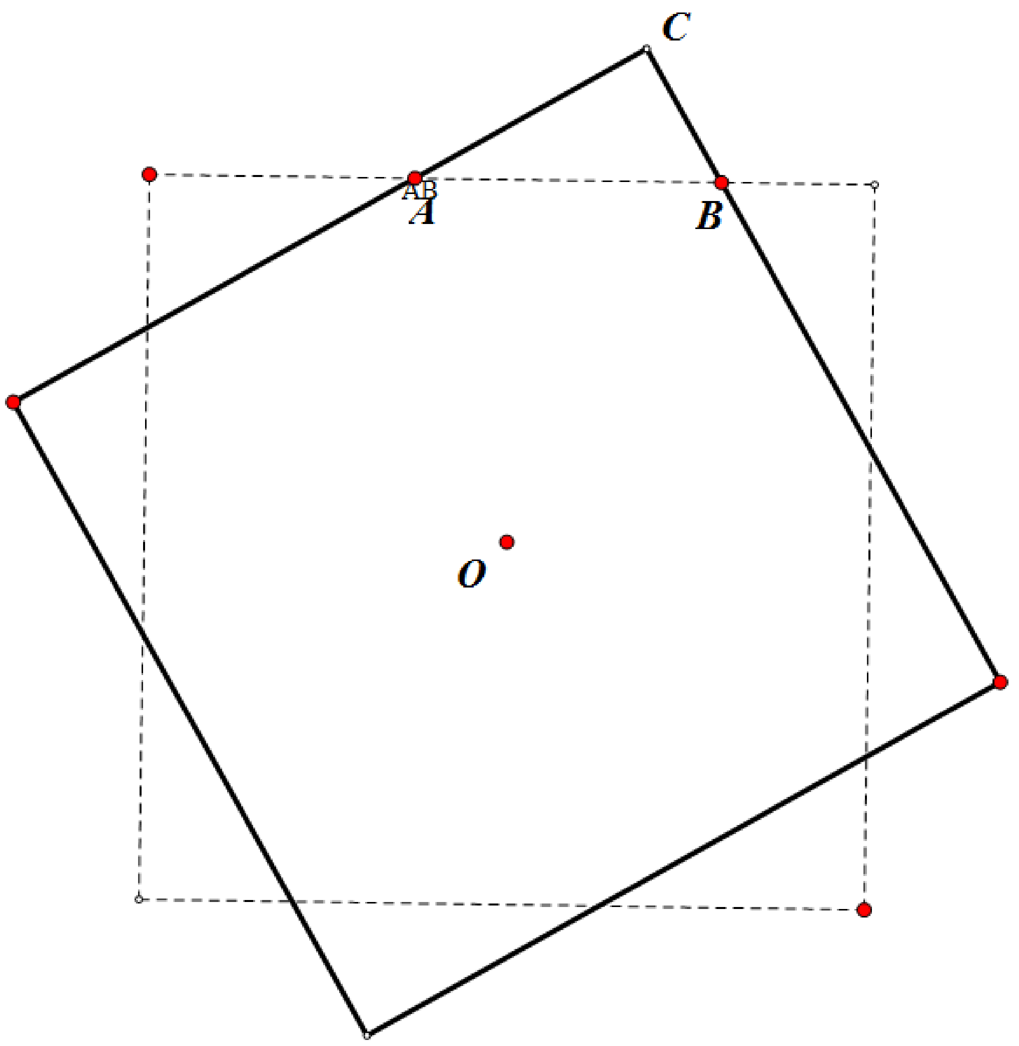
Calcuate the overlapping area with the rectangle before rotated.

Tags

geometry

Analysis

It may always be a good idea to try a simplified version before the situation gets into complicated. In this problem, a simplified version you might to be noticed is the **square** case.

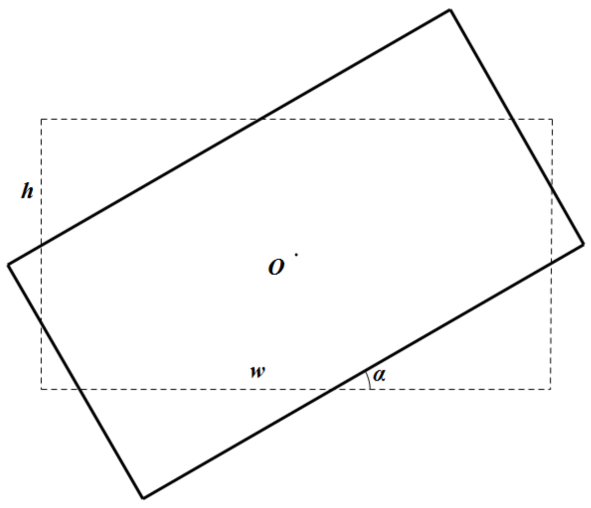


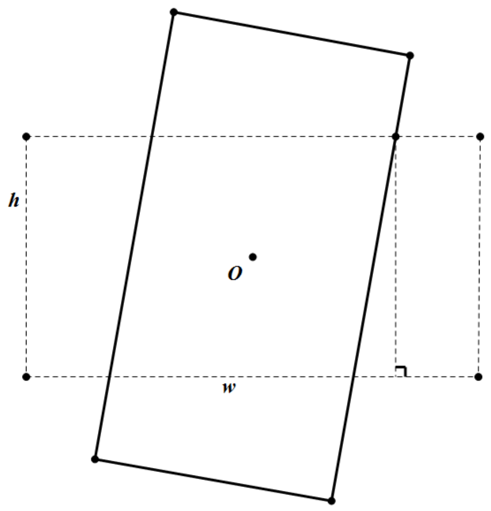
So here it is. you can easily see the area can be calculated by

$$S - 4 \times T$$

Here S is the area of the large square, and T is the area of those ambient triangles.

In the rectangle situation, however, things will turn out to be more complex. But if you come back to the problem after draw some pictures, you can found there are only two cases we need to dig them out:





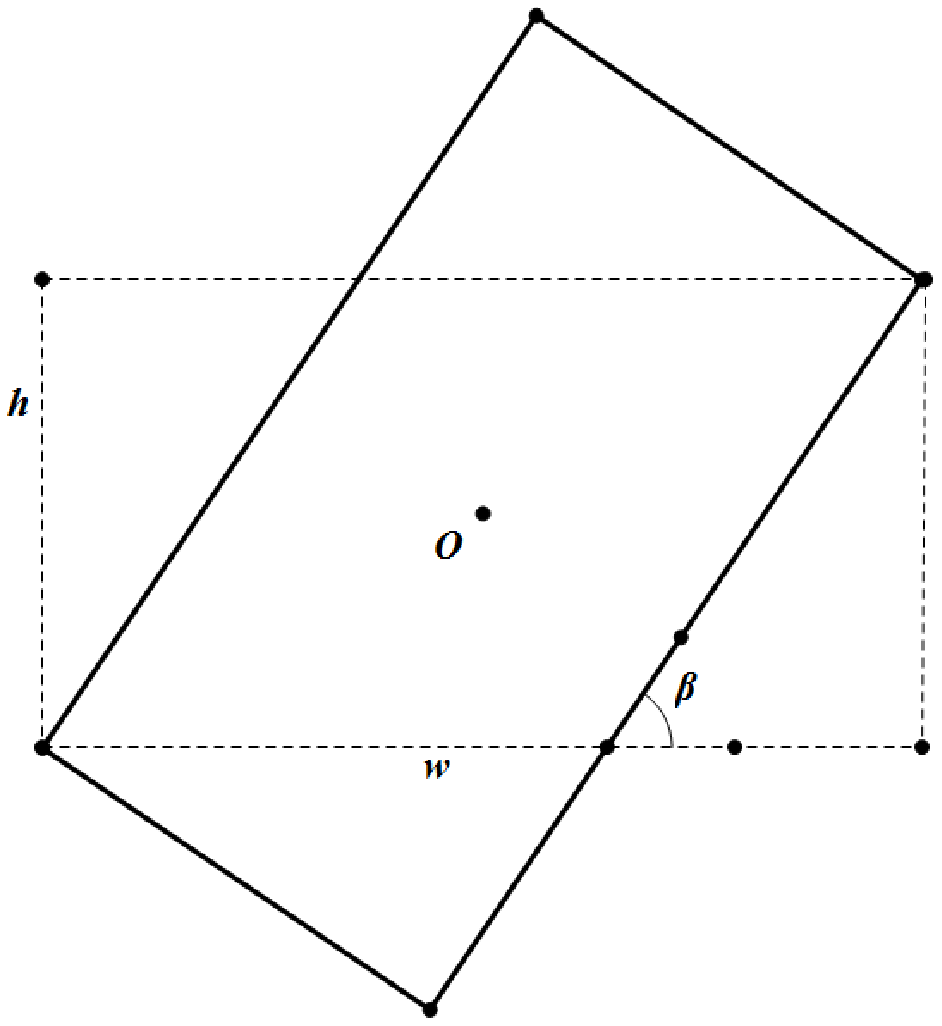
The first case is quiet similar to the **square** which we discussed just now. The second case is rather easy if you consider the parallelogram. And here the key point is come ... .

..

..

Where is the **watershed** between them?

..



You can calculate β by exterior-interior angles, similar triangle, solving a system of linear equations of two unknowns even binary search on the angle or any other way you could imagine. In the end, youĺl find $\beta = 2\arctan\frac{h}{w}$.

As long as you can find the **watershed**, the greater part of the problem has been finished.

Maybe you have noticed, this problem can also be solved by some **computational geometry** method, such as **half-plane intersection** or **convex-hull**.

Negative Example

UESTC_Nocturne: A negative example hacked by scottai1, which didn't consider the second case mentioned above. This become a common phenomenon around other negative examples. (scottai1 himself failed because he use cout without setprecision.)

liouzhou_101, wuzhengkai: Same as UESTC_Nocturne, Peter50216 hacked wuzhengkai's solution in the halfway, so wuzhengkai can struggle himself out during the remain contest. Other two guys, unfortunately, do not have even that comfort.

Suggested Code

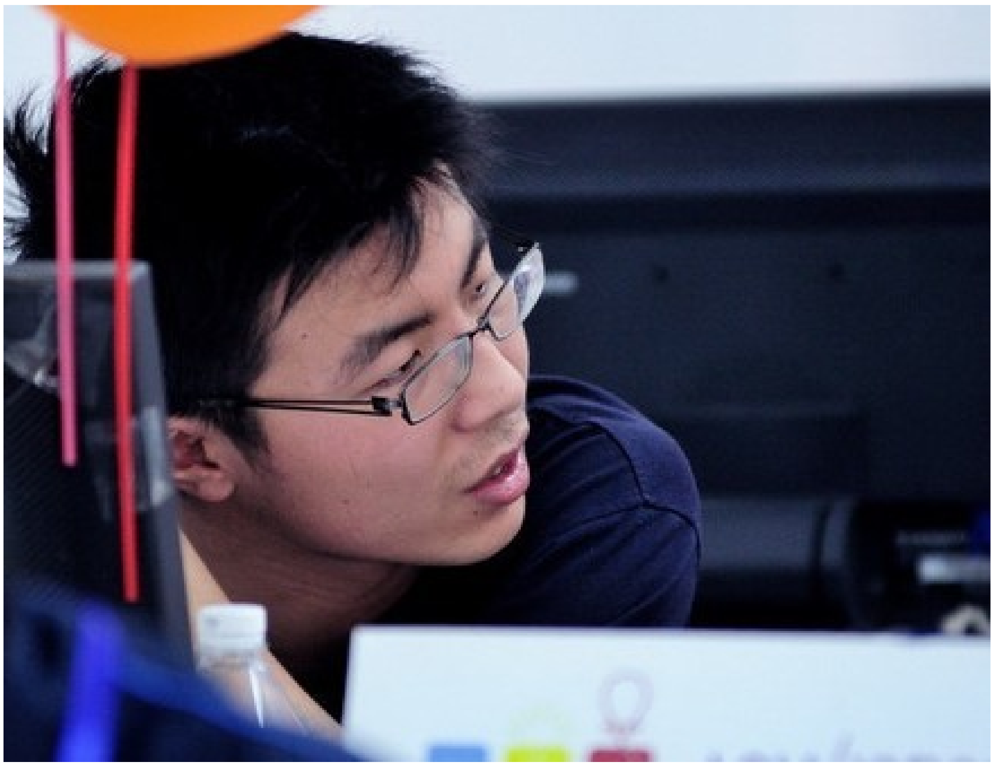
peter50216: The first math solution during the contest.

dhh1995: The shortest **C++** code written after the contest.

tourist: A light half-plane intersection function written by hand during the contest.

sune2: The first correct solution during the contest. This is the best geometry template I have ever seen.

daizhenyang: I am very honored to recommend another well implemented geometry template to you. This template is belong to the famous Chinese ACMer Zhenyang Dai.[1]



[1]Zhenyang Dai (中文名：戴振阳) has been known as the man who stand on the top of the graph-theory in Chinese Mainland since the last decade. He is also the man who put forward the well-known theory which been titled as "*All the things are Network-Flow(一切皆网络流)*" on the 2008 Beijing Congress of Mathematicians(北京数学家大会) and introduce it to the public.

4 Problem B. Maximum Xor Secondary

Brief Description

You've been given a sequence of distinct positive integers. Your task is to find a interval [l, r], so that the maximum element **xor** the second maximum element within the interval is as large as possible.

Analysis

The interval's meaning can only be reflected on its maximum element and second maximum element, so apparently, there must be a lot of meaningless interval which we needn't check them at all. But how can we skip them? Maintain a **monotone-decreasing-stack** can help us. While a new element came into the view, pop the top element in the stack, and check the corresponding interval, until the new element is greater than the top element in the stack. We can easily see it is correct since we won't lost the answer as long as it exists.

All the element at most push and pop once, and only been checked when popped. So the time complexity turn to be $O(n)$.

Negative Example

ramtin95: Some competitors were misunderstanding on the term "bitwise excluding OR". We fell so sorry to heard about this but we can't give back your points on this particular case.

watashi: Some competitors used strange algorithm that we didn't know what they are thinking ... ...

Suggested Code

havaliza: The first correct solution during the contest, which is a clean explanation for the algorithm we described above.

xiaodao: Our standard program is slightly different with most of submissions during the contest, we avoided to run the main algorithm on the reversal. :-)

5 Problem C. Game on Tree

Brief Description

You are played a single-player game on a rooted tree. On each step, you choose a remaining node randomly, remove the subtree rooted by the node until all of them have been removed. Return the expectation of the number of steps in this game.

Tags

probabilities, trees

Analysis

The answer is

$$\sum_{i=1}^{n} \frac{1}{Depth[i]}$$

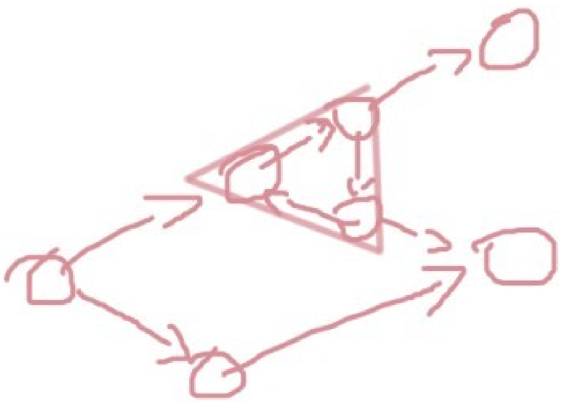
Here we denoted the $Depth[root] = 0$.

And why it works?

Well, let us observe each vertex *independently*, when one vertex has been removed, y-ou need either to remove it directly, or remove one of its ancestors. All these choices are equiprobable, so the probability of direct removal is $\frac{1}{Depth[i]}$. And the sum of them is our result.

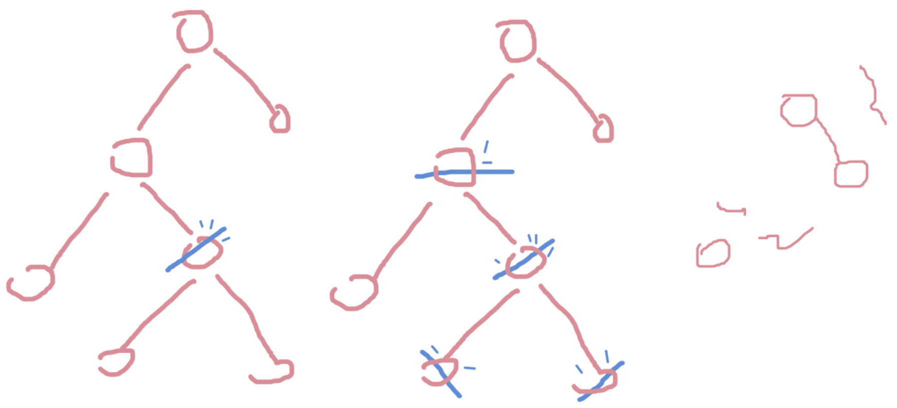
This is been known as the *Linearity of the Expected Value*[2]. This property arises almost every way in those problems related with probability.

This method will also work when the objects of the delete operation is under a partial order relation.



[2] Wikipedia - Linearity of the Expected Value

For example, on a DAG or even a digraph(reduce it into a DAG[3]).
Finally It will be reduce to *descendant counting*[4].



But when the delete option is look like .. .

```
 ...  remove the node and all its neighbours ..  .
```

such disordered form, the method can do nothing.[?][5]

Suggested Code

jonathanpaulson: A elegant **Java** which is reading in the graph into a nice data structure.[6]

Related Problem

Codeforces #146 DIV 1 Problem D. Graph Game
Topcoder Open Round 1C Level 3. TheKnights
ACM-ICPC Amritapuri onsite regionals 2012. Problem F. The Loyalty of the Orcs

[3]Wikipedia - Strongly Connected Component
[4]Paper - Size-Estimation Framework with Applications to Transitive Closure and Reachability
[5]Open Problem - Solvable?...
[6]Issue - This code was suggested by Daniel Sleator. Here is the original message. Well, as in Algorithm Competition, people usually use some mature template against rewrite them from the sketch. So most of time you can't assessing the code's quality according to their redundance. However in this particular, maybe professor is right. I did not recommend his code because Ocaml is quiet unfamiliar with most of us, not out of the preconception on language. On the contrary, I do think Ocaml is a beautiful computer language that everyone should have it a try ;)

6 Problem D. k-Maximum Subsequence Sum

Brief Description

Giving a number sequence A_i, on this sequence you need to implement the following two operations:

- `0 x v`: Change A_x to v.
- `1 l r k`: Query the **k-MSS** in [l, r].

Tags

data structures, dp, flows

Analysis

Let's back to the static problem and review how can we use `dynamic programming` to solve it.

- $f_0[i][j]$: the **j-MSS** in [0, i].
- $f_1[i][j]$: the **j-MSS** in [0, i], which the element A_i must be selected.

The state transition is enumerating whether the ith element is selected or not. So now we have our initial *brute-force* method — Doing a *dynamic programming* for each query which is $O(1)$ for `Modify` and $O(nk)$ for `Query`. It is just too common to mention and have no chance to get **Accepted**.

Our main motivation here is using a data structure to speed up the `Query` operation. But is a interval tree enough? Let's have a try! In each node, we maintain the following information:

- the **i-MSS**
- the **i-MSS** which the first element must been selected
- the **i-MSS** which the last element must been selected
- the **i-MSS** which both first and last element must been selected

Let's see how can we concatenate two adjacent interval into a single one, which is the most important part in this data structure and usually has been named as `Update`.

Denoted the two intervals as L and R, then for each **i-MSS** in the result interval, we could enumerate how many subsequences will be chosen in L(Denoted j), and then we will get how many subsequences we will chosen in R (Which is $i - j$) and get the target **i-MSS** in $O(1)$. Then overall time complexity for concatenating two intervals is $O(k^2)$.

Since both `Modify` and `Query` is involve `Update`, both the complexity of `Modify` and `Query` is $O(k^2 \log n)$.

It seems hard to be optimized using such thought. Now we are going to use another totally different thought. We construct a **max-cost max-flow** model to solve this problem. The source is S and the sink is T. There are other $n+1$ vertices. A bidirectional edge between Vertex i and Vertex $i+1$ has cost A_i and capacity 1. The unidirectional edge from S to Vertex i has cost 0 and capacity 1 and the unidirectional edge from Vertex i to T has cost 0 and capacity 1. It's obvious for correctness. We can use **Successive shortest path** algorithm to solve the flow problem. But it's as slow as the **brute-force** method, or even worse.[7]

From now this problem has been reduce to GSS3. Considering how the flow problem is solved, that is, how **Successive shortest path** algorithm works. We find the longest path(**max-cost** instead of **min-cost**) between S and T, and then augment. We can simplify the algorithm due to the special graph. The new algorithm is:

- find the subsegment with maximum sum
- negate the elements in the subsegment
- repeat the two steps k times
- the answer is the sum of the sum of the subsegment you found each time.

So, the key point is doing these two operations quickly. Segment tree can also be used. To find the **MSS** you need to store

- the sum of a interval
- the maximum partial sum from left
- the maximum partial sum from right
- the **MSS** in the interval

To negative a subsegment, the minimum elements must be stored. To find the position of **MSS**, all the positions are supposed to kept. The complexity of `Modify` is $O(\log n)$, and the complexity of `Query` is $O(k \log n)$. It can pass all the test data easily but it has a really large constant.

[7]Open Problem: Can we prove its correctness without using MCMF?

Honor Hall

UESTC_Nocturne: The first correct code among the game. It is this submission, acclaimed him as the champion.

FattyPenguin: The fastest code among the contest.

tclsm2012: The only solution made by purple.

liouzhou_101's solution: A $O(mk^2logn)$ solution with dramatic optimizations that get pass during the contest.

Related Problem

SPOJ 1716. Can you answer these queries III

APIO 2007 Problem B. backup ...

HDU 1024. Max Sum Plus Plus

POJ Challenge 2011.04.10 Problem B. Birthday Present(生日礼物)

Reference

From Maximum Subsequence Sum to Maximum Submatrix Sum(Chinese!)(从连续最大和到最优子矩阵, 史沛)

Optimization on m-MSS, Haoqiang Fan (Chinese!)(M段最大和的优化, 范浩强)

7 Problem E. Sequence Transformation

Brief Description

You've got n, Q, A, B and a non-decreasing sequence $X_1, X_2, \ldots, X_n (1 \le X_1 \le X_2 \le \ldots \le X_n \le Q)$. Your task is transform sequence $X_1, X_2, \ldots, X_n$ into some sequence $Y_1, Y_2, \ldots, Y_n (1 \le Y_i \le Q; A \le Y_{i+1} - Y_i \le B)$, and minimizes the transformation price which defined as:

$$\sum_{i=1}^{n} (Y_i - X_i)^2$$

Tags

data structures, dp, math

Analysis

The description of the problem seems quite easy to understand. And anyone can put it into the mathematical as follow:

$$w(i, x) = (x - X_i)^2$$

$$g(i, x) = \begin{cases} 0 & i = 0 \\ min\{f(i-1, x') \mid x - b \le x' \le x - a\} & \text{otherwise} \end{cases}$$

$$f(i, x) = g(i, x) + w(i, x)$$

Here $w(i, x)$ stands for the price on the ith element, and $f(i, x)$ is the minimum transformation price if we set Y_i equal to x, and $g(i, x)$ is the auxiliary function to calculate $f(i, x)$.

Now, we can approach those functions from different angles. One is take them as dynamic programming states and their transition function. The states are $O(n\infty)$[8] and the transition is $O(\infty)$ in the worst case and surely it'll never work. We need come up with an algorithm whose time complexity does not involve ∞ as a factor.

But how can we achieve that?

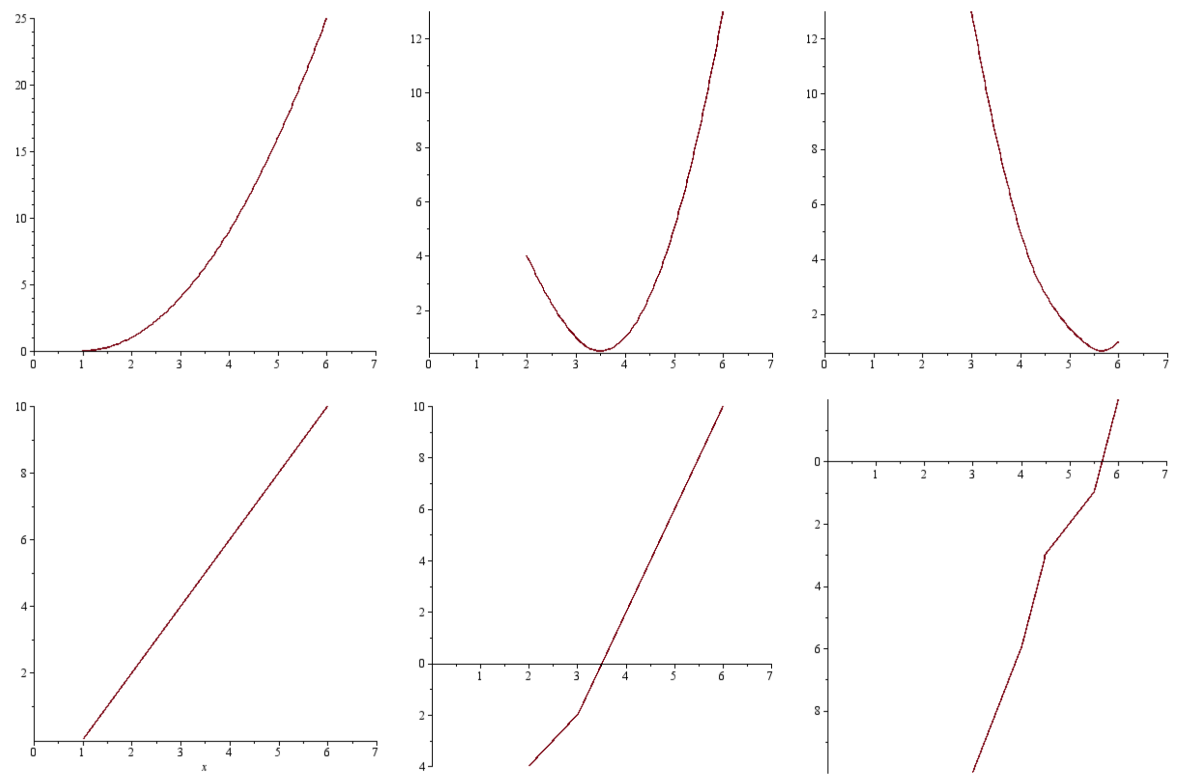
...

[8] The state space is continuous.

We cannot make any progress until we regard those equations as true `mathematical functions`.

For example ...

```
3 6 1 2
1 4 6
```



The graph of f and f'.

(Comming Soon ...)

$O(?n^2)$ - Finding the minimum - Ternary Search

...

$O(n^2)$ - Maintain the Derived Function

Seter

$O(nlogn)$ - Map the Derived Function into a Binary Search Tree

Seter

$O(n^2)$ - Jacob's Adjusting algorithm

Jacob

$O(n^2)$ - Daniel Sleator's DP algorithm

Darooha

Python C++

Jacob

...

Related Problem

CTSC 2009. Sequence Transform(序列变换) ...

Reference

Analysis of Problem Sequence[Sinya Lee].pdf

Another Solution of Problem Sequence[Sinya Lee].pdf