Analysis of Problem Sequence

Sinya Lee

Jinshan High School, Swatow, China

Table of Contents

A Simple DP Algorithm

The description of the problem seems very easy to understand. And I came up with a DP solution 1 second after I understood the problem. Let $dp[i][j]$ be the minimum cost to let Y_1 to Y_i match the conditions and let $Y_i = j$. So the state transition equation is very easy to get:

$$dp[i][j] = min\{ (1 \le k \le Q) \& (j - B \le k \le j - A) \mid dp[i-1][k] \} + |j - X_j|$$

Neither Dongdong nor his test data are friendly to us students. As a result, the range of X and Q are extremely large so that the DP algorithm above will cost about 1 billion years to get the solution for the worst case on my poor laptop computer. Because $N \le 500000$ but $Q \le 10^9$, we need an algorithm whose time complexity doesn't have a factor Q.

Family of Functions and the Recursive equation

As a student who loves mathematics very much, I tried to observe the family of functions $f_i(x)$, which means the minimum cost to let Y_1 to Y_i match the condition and let $Y_i = x$. The domain of function $f_i(x)$ is $[1 + (i-1) * A, Q]$. It's is similar to $dp[i][j]$, we can get a recursive equation like the state transition equation:

$f_i(x) = | x - X_1 |$ | for $i = 1$
$f_i(x) = min \{ x - B \le x' \le x - A \mid f_{i-1}(x') \} + | x - X_i |$ | for $i > 1$

A Simple Sample

We cannot get any observation immediately after we got the equation. So, keep it simple and stupid! Considering the following input:

3 6 1 2
1 4 6

It is obvious that $f_1(x) = | x - 1 |$. **(Figure 1)**

And then let's try to find out the analytic expression of $f_2(x)$. From the recursion expression, we know:

$f_2(x) = min \{ x - 2 \le x' \le x - 1 \mid f_1(x') \} + | x - X_2 |$
$= min \{ x - 2 \le x' \le x - 1 \mid | x' - 1 | \} + | x - 4 |$

The function is the sum of $min \{x - 2 \le x' \le x - 2 \mid | x' - 1 | \}$ and $| x - 4 |$. $| x - 4 |$ is a very simple function. **(Figure 2)**

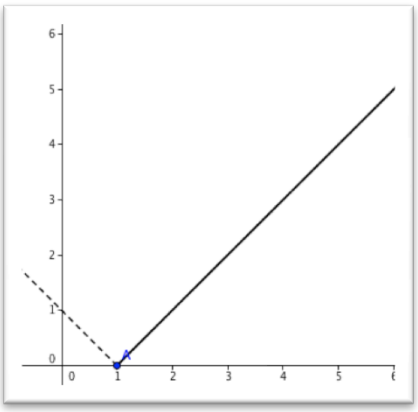

Figure 1

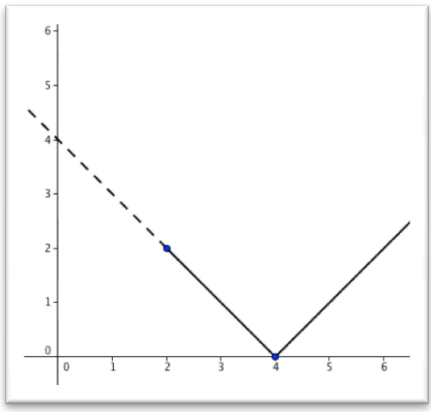

Figure 2

But $min \{x - 2 \le x' \le 2 - a \mid | x' - 1 | \}$ is not easy to figure out. So let's try to part the problem into 3 cases:

i. If $x \le 1 + a = 2$, $x' \le x - A \le 1$. So the function become $min \{x - B \le x' \le x - B \mid 1 - x'\}$. It is obvious that the bigger x' is, the smaller $(1 - x')$ is. So $min \{x - B \le x' \le x - A \mid 1 - x'\}$ is equal to $1 - (x - A)$, i.e. $2 - x$.

ii. If $x \geq 1 + B = 3$, $x' \geq x - B \geq 1$. So it is $min\{x - B \leq x' \leq x - A \mid x' - 1\}$. It is obvious that the smaller x' is, the smaller $(x' - 1)$ is. So $min\{x - B \leq x' \leq x - A \mid x' - 1\}$ is equal to $(x - B) - 1$, i.e. $x - 3$.
iii. If $2 = 1 + A < x < 1 + B = 3$, $x - B < 1$ and $x - A > 1$. So x' can be 1. When x' is 1, $| x' - 1 |$ will be 0, which is its minimum value. So $min\{x - B \leq x' \leq x - A \mid x' - 1\}$ is 0.

So we can get the graph of $min\{x - B \leq x' \leq x - A \mid x' - 1\}$. **(Figure 3)**

After adding $| x - 4 |$, it will be $f_2(x)$. **(Figure 4)**

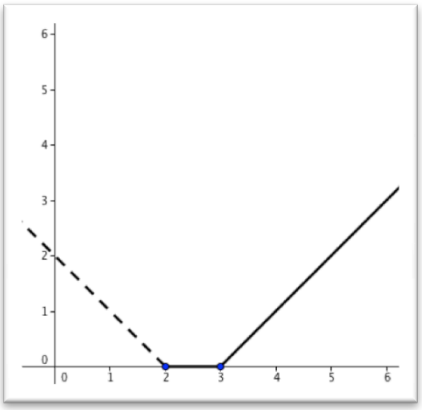

Figure 3

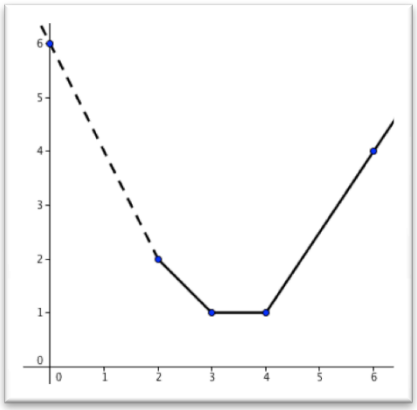

Figure 4

And then **Figure 5** is the graph of $min\{x - 2 \leq x' \leq x - 1 \mid f_2(x')\}$.

After adding $| x - 6 |$, it will be $f_3(x)$. **(Figure 6)**

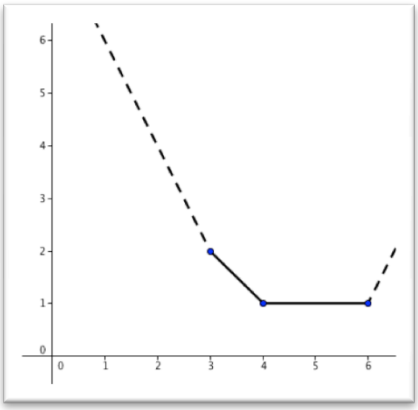

Figure 5

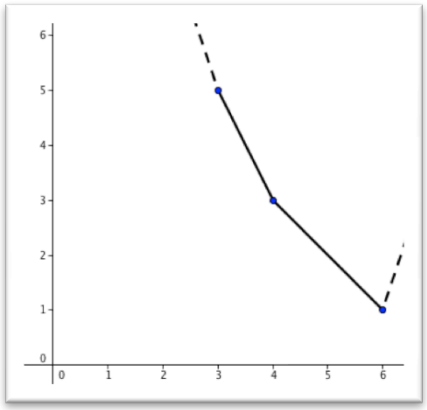

Figure 6

General Algorithm

If we have worked on the sample above, we can easily find a general algorithm to get the analytic expression of $f_i(x)$.

Suppose $f_{i-1}(x)$ is a concave function, and $f_{i-1}(t)$ is the minima, if we move the graph of $f_{i-1}(x)$ in the left of t A unit(s) rightwards and the graph in the right B unit(s) rightwards and fill the space between $t + A$ to $t + B$ with a horizon segment, we will get the graph of $min\{x - B \le x' \le x - A \mid f_{i-1}(x')\}$. (Like the transition from **Figure 1** to **Figure 3**, or the one from **Figure 4** to **Figure 5**)

To keep the article brief, I don't provide the proof here. You can go to the sample above to see the proof of the transition from **Figure 1** to **Figure 3**.

And then if we add $| x - X_i |$ to the graph, we will get the graph of $f_i(x)$. Since $| x - X_i |$ is also a concave function, so $f_i(x)$ is also a concave function. Another fact is that $f_1(x)$ is a concave function. So with mathematical induction we can prove that all the functions are all concave. With mathematical induction we can prove that the graphs of all the function are all consist segment(s) too.

So we can get a algorithm easily. At first we can get the function $f_1(x)$. And then we try to get $f_2(x)$ to $f_i(x)$ one by one with the operations above.

It is obvious that we have to pick up one data structure because if we transform $f_{i-1}(x)$ to $f_i(x)$ directly, the total time complexity is $O(N \wedge 2)$, which will cost too much time.

Using Binary Search Tree

The first idea that came to me is to use BST. Use it to record the slope and endpoints of segments in the graph of the function. If we want to transform $f_{i-1}(x)$ to $f_i(x)$, this data structure can easily find the minima (binary searching by comparing the slope) in $O(log\ N)$. And then move all the segments before it A unit(s) rightwards and segments after it B unit(s) rightwards. If we use lazy propagation, we can do this in $O(log\ N)$. And then we can insert a segment. After that we should add $| x - X_i |$ the function. We can find X_i, and add -1 to all the slopes of the segments before X_i and add 1 to all the slopes of the segments after X_i. These can also be done in $O(log\ N)$ with lazy propagation.

Focus on the Slopes

I am very weak in data structures and I hate BST very much! If I use BST for this problem I need to have record 2 values for lazy propagation, it will be very troublesome when rotations are needed. So let's try to simplify the problem.

Let's ignore the fact that the domain of function $f_i(x)$ is $[1 + (i - 1) * A , Q]$. The leftmost part of the function is a ray which has a slope $-i$, and the rightmost part is a ray which has a slope i. It can be easily proved using the recursive equation. And because of the slopes of rays and segments are all integers; we can record all the points where the slope of the function increases by 1 unit.

We can use **Figure 1, 3, 4, 5, 6** as examples. We don't ignore graph out of the domain. For example, in $f_1(x)$ **(Figure 1)**, at first the slope is -1, and then we can find that when $x = 1$ the slope increases by 2 units. So all the points that the slope increases by 1 unit are:

1 1

1 appears twice because when $x = 1$ the slope increase by 2 units.

And in **Figure 3 to 6**, the points where the slope increase in 1 unit are:

$min\{ x - b \le x' \le x - a \mid f_1(x) \}$	**(Figure 3):**	2 3
$f_2(x)$	**(Figure 4):**	2 3 4 4
$min\{ x - b \le x' \le x - a \mid f_2(x) \}$	**(Figure 5):**	3 4 6 6
$f_3(x)$	**(Figure 6):**	3 4 6 6 6 6

So when we know all the points of $f_{i-1}(x)$, we can move the first i points A unit(s) rightwards and the other i points B unit(s) rightwards. And then if we add two X_i to the list it will be the points where the slope increase by 1 unit in $f_i(x)$. The correctness can be proved.

Finally we will get the list of points where the slope of the graph of $f_N(x)$. Actually this list is a bit like the second derivative. But we need to get the graph of $f_N(x)$ itself. Because we have got the list of points where the slope of $f_N(x)$ increase by one, we also know that the slope in the leftmost part of the function is $-N$, we can get $f_N'(x)$. After that we can get $f_N(x)$ itself because value of $f_N(1 + A * (N - 1))$ can be determined. There is only a way to let $Y_N = 1 + A * (N - 1)$.

Using Heaps

There are many ways to record the points where the slope of $f_i(x)$ increases by 1 unit. For example, BST is a choice. But I hate it very much, so I found that we could record

the list with two heaps, a max-heap for the first (smaller) i points and a min-heap for the other i points. With these two heaps we can transform $f_{i-1}(x)$ to $f_i(x)$ in $O(\log N)$.

Program

sequence_1.cpp

```
// For Junde Huang.
// sequence, CTSC 2009.
// Written by Sinya in September, 2009.

#include <iostream>
#include <cstdlib>

#define maxn 500010
#define inf 1000000000000000 // 10^15

using namespace std;

int n;
long long q, a, b;
long long x[maxn];

int size[2];
long long heap[2][maxn], delta[2];

long long p[maxn];

int comp( const void *a, const void *b) {
  long long temp= *(long long*)a - *(long long*)b;
  if ( temp== 0) return 0;
  return ( temp> 0) ? 1: -1;
}

void up(const int t,const int x) {
  if (x== 1) return;
  const int y= x >> 1;
  if ( (heap[t][x]> heap[t][y])^ t ) {
    swap( heap[t][x], heap[t][y]);
    up( t, y);
  }
}

void down( int t, int x) {
  int y= x << 1;
  if ( y> size[t]) return;
  if ( ( y+1<= size[t] ) && ( ( heap[t][y+1]> heap[t][y]) ^ t) )
    y++;
  if ( ( heap[t][y]> heap[t][x])^ t ) {
    swap( heap[t][x], heap[t][y]);
    down( t, y);
```

```
  }
}

void add( int t, long long data) {
  heap[t][++size[t]]= data;
  up( t, size[t]);
}

void del( int t) {
  heap[t][1]= heap[t][size[t]--];
  down( t, 1);
}

int main() {
  freopen("sequence.in", "r", stdin);
  freopen("sequence.out", "w", stdout);

  scanf( "%d%lld%lld%lld", &n, &q, &a, &b);
  for ( int i= 1; i<=n; i++)
    scanf( "%lld", &x[i]);

  long long val= 0;
  for ( int i= 1; i<= n; i++)
    val += abs( 1+ a*(i - 1) - x[i] );

  size[0]= size[1]= 1;
  heap[0][1]= heap[1][1]= x[1];
  delta[0]= delta[1]= 0;
  long long left= 1;
  for ( int i= 2; i<=n; i++) {
    delta[0] += a;
    delta[1] += b;
    int t;
    if ( x[i]<= heap[0][1]+ delta[0]) t= 0;
    else t= 1;
    left+= a;
    if ( x[i]< left) x[i]= left;
    add( t, x[i]- delta[t]);
    add( t, x[i]- delta[t]);
    add( t^ 1, heap[t][1]+ delta[t]- delta[t^ 1]);
    del( t);
  }

  for ( int i= 1; i<= n; i++)
    p[i]= heap[0][i]+ delta[0];
  qsort( &p[1], n, sizeof(long long), comp);
  p[0]= left;

  long long k= -n;
  int i;
  for ( i= 0; i< n; i++) {
    if ( p[i+1]> q) break;
    val+= ( p[i+1]- p[i]) * ( k++);
  }
  long long sol= min( p[i+1], q);
```

```
  sol= val+ (sol- p[i])* k;
  printf( "%lld\n", sol);

  return 0;
}
```