ABC 059 / ARC 072 解説

writer : hogloid

2017 年 4 月 22 日

A : Three-letter acronym

「小文字を大文字にする」という操作は、ライブラリ関数を用いるか、ASCII 文字コードで大文字や小文字は連続して配置されていることを利用して文字コードの差を足し算して求めることもできます。

C++ のコード例

```
int main(){
  string a, b, c;
  cin >> a >> b >> c;
  char dif='A' - 'a';
  printf("%c%c%c\n", a[0] + dif, b[0] + dif, c[0] + dif);
        return 0;
}
```

B : Comparison

まず、2 つの数の桁数が異なる場合、桁の多い数の方が大きいです。桁数が等しい場合、数の大小は大きい桁から順に並べた文字列の辞書式比較と等しくなることをプログラムすればよいです。

C : Sequence

偶数番目と奇数番目どちらを正にするかで 2 通り考え、小さい方を答えることにします。番号の小さい順に数を確定させていきます。

i 桁目までの和の符号が欲しい符号と同じ場合、i 番目の数を変更する必要はありません。すでに i 番目までの和の符号が合致しているのに i 番目の数を変更して最適解が得られるなら、i 番目の数に対する変更を $i+1$ 番目の数に持ってくることで同じ最適解が得られるからです。

i 桁目までの和の符号が欲しい符号と異なる場合、まず欲しい符号のうち絶対値最小までは変更しなくてはいけません (1 か -1)。これ以後は上と同様の議論が成り立つので、1 か -1 まで変更すれば十分です。

この規則で前から順に決めコストを求めていくことで答えが求まります。

D : Alice&Brown

結論としては、$|X-Y| \leq 1$ のとき Brown、そうでなければ Alice が勝ちます。以下の帰納法を $X+Y$ の昇順に回すことで、この性質が成り立つことがわかります。

- $X+Y \leq 1$ のとき有効な操作がないため、Brown が勝ちます。
- $X+Y>1, |X-Y| \leq 1$ のときどんな操作をしても、$|X-Y|>1$ で相手に手番を渡してしまうため、帰納法の仮定により相手が勝ち、自分は負けます。
- $X+Y>1, |X-Y|>1$ のとき山のうち石が多くある方の山から適切な数取ることで、$|X-Y| \leq 1$ で相手に手番を渡すことができます。帰納法の仮定により相手は負け、自分は勝ちます。

E : Alice in linear land

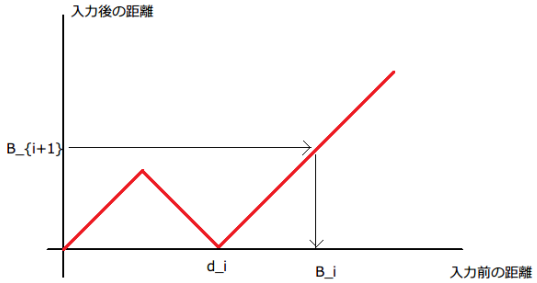
i 番目までの数を入力したときに目的地までの距離 A_i を前もって計算しておきます。

i 番目に入力する数を自由に変更できるとき、その数の入力が終わった時点で $\forall x \in [0, B_i]$ について Alice が目的地 x の距離にいるよう数を設定することができます。逆に、P_i より目的地に遠い場所に移動させることはできません。

よって、i 番目以降の数を入力しても Alice が目的地に辿りつけない最小の距離 B_i が求まれば、i 番目を変更する計画が可能かは $A_i \geq B_{i+1}$ かどうかと等しくなります。

B_i は実は後ろから順に求めていくことができます。まず、$B_{N+1}=1$ です。

- $B_{i+1} \leq \lfloor d_i/2 \rfloor$ のとき $B_i = B_{i+1}$
  $\because$ B_{i+1} の距離から d_i を入力しても $B_{i+1} \leq \lfloor d_i/2 \rfloor$ より動きません。よって、B_{i+1} から i 番目以降の数を入力しても目的地にたどり着きません。これより近い場合、同様に i 回目の数の入力では動きませんが、$i+1$ 番目の数を入力する時点で B_{i+1} より近い場所にいるので、目的地にたどり着きます。
- それ以外のとき $B_i = B_{i+1} + d_i$
  $\because$ 同様に、距離 B_i から i 番目以降の入力を行っても目的地にたどり着きません。距離 B_i より近い位置から i 番目の入力を行ったとき、B_{i+1} より目的地に近い位置に動くので、仮定から目的地にたどり着きます。(以下の図参照)



F : Dam

i 日目までに流入した水を、流入した日が後になるにつれ水温が昇順するよう管理することを考えます。こうすることで、水を入れるためにダムから排水するときは、今ある水のうち最も水温の低い (=最も早く流入した) ものを過去に排出したことにすることで対応できます。

水が流入したとき、その前で流入した水より温度が低いときは、上記の性質が壊れてしまいますが、代わりにその 2 種類の水が混ざって同時に流入してきたものとして扱うことができます。なぜなら、その前で流入した水を捨てるより、冷たい水が混ざってから捨てたほうが残った水の温度が高くなるからです。

以上の操作は、両端キューを使うことで水の排出は先頭からの削除、水の流入は末尾の削除と追加により毎回の操作を O(1) で行うことができます。i 日目には、水の排出と流入を終えた後、全体の熱量 (体積×温度) の総和を L で割った値が答えです。1 日の間に要素を複数削除することはありますが、追加は 1 日につき 1 回なのでならしで O(N) となります。

ABC 059 / ARC 072 Editorial

writer : hogloid

2017 年 4 月 22 日

A : Three-letter acronym

C++ Code

```
int main(){
  string a, b, c;
  cin >> a >> b >> c;
  char dif='A' - 'a';
  printf("%c%c%c\n", a[0] + dif, b[0] + dif, c[0] + dif);
        return 0;
}
```

B : Comparison

If the two numbers have different lengths, the longer number is greater. Otherwise, if the two numbers have the same lengths, you can compare them lexicographically.

C : Sequence

Try two possibilities independently: even-length prefixes are positive or odd-length prefixes are positive.

You determine the numbers greedily from left to right. When you check the i-th term, if the sum of the first i terms already has the desired sign, you don't need to change the i-th term. (Instead of changing it, you can do the same thing by performing the same operation for the $i + 1$-th term).

Otherwise, you should change it such that the sum of the first i terms becomes 1 (in case the desired sign is positive) or -1 (in case the desired sign is negative).

This way, you can compute the optimal cost.

D : Alice&Brown

If $|X-Y| \leq 1$, Brown wins. Otherwise Alice wins.

- $Incase |X-Y| \leq 1$, no matter how you move, after your turn $|X-Y| > 1$ will be held, and your opponent wins.
- In case $X+Y > 1, |X-Y| > 1$, if you take appropriate number of stones, you can satisfy $|X-Y| \leq 1$ after your turn and you win.

E : Alice in linear land

Suppose that the witch changes the q-th number.

Let A_i be the distance to the goal after you finish the first i operations (when the which doesn't do anything). Then, before the q-th operation, you will be at A_{q-1}. After the q-th operation, you can be at arbitrary integer place between 0 and A_{q-1}, inclusive (depending on the change made by the witch).

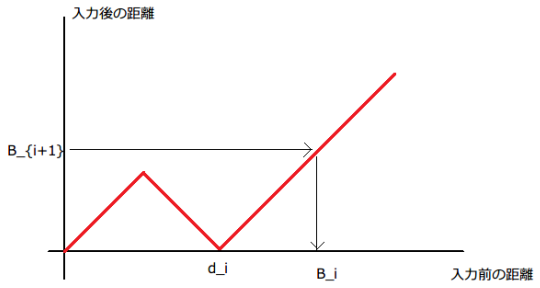
Let B_i be the smallest integer such that, if you are at B_i after the first i operations, you won't reach the goal. Clearly, the answer to the query is 'YES' if and only if $B_q \leq A_{q-1}$.

You can compute B_i in the decreasing order of i as follows:

- Obviously, $B_N = 1$.
- If $B_{i+1} \leq \lfloor d_i/2 \rfloor$, $B_i = B_{i+1}$

  $\because$ If you start from B_{i+1} and perform the operation d_i, you won't be moved by this operation and you won't reach the goal. If you start from closer positions, you will reach the goal by the definition of B_{i+1}.
- Otherwise, $B_i = B_{i+1} + d_i$

  $\because$ Check the following picture. Here, the horizontal axis shows the distance before the operation d_i, and the vertical axis shows the distance after the operation d_i.



F : Dam

The state of the dam can be represented by two parameters: [the amount of water] and [the product of the amount of the water and the temperature]. This way, when we mix two cups of water with parameters (x_1, y_1) and (x_2, y_2), we get water with the parameter $(x_1 + x_2, y_1 + y_2)$.

At each time, we should keep the set of possible states of the water in the dam. If you plot this set on an x-y plane, it will always be a concave polyline. For example, after the first day, this is a line segment that connects $(0, 0)$ and $(v_1, t_1 v_1)$.

How can we store/update this polyline? The polyline can be seen as a sequence of vectors. We keep it using a deque.

- When we add water with the state (x, y), we simply push it to the front of the deque.
- When we discharge water, replace the polyline with its convex hull.

And for each query, we answer the y-coordinate of the polyline at $x = V$. The amortized complexity is $O(N)$.