ABC 146 解説

beet, kort0n, kyopro_friends, tempura0224

2019 年 11 月 24 日

For International Readers: English editorial starts on page 7.

A: Can't wait for holiday

与えられた文字列がどの曜日に対応するものであるか調べて分岐し、答えを出力します。

C++ での実装例は次のとおりです。

```
#include <iostream>
using namespace std;
int main(){
    string s;
    cin >> s;
    int ans;
    if(s == "SUN")ans = 7;
    else if(s == "MON")ans = 6;
    else if(s == "TUE")ans = 5;
    else if(s == "WED")ans = 4;
    else if(s == "THU")ans = 3;
    else if(s == "FRI")ans = 2;
    else if(s == "SAT")ans = 1;
    cout << ans << endl;
}
```

B: ROT N

まず元の文字がアルファベットの何番目の文字かを 0-indexed で求め、それに N を加えて 26 で割った余りを求めると、出力するべき文字が分かります。

```
#include<stdio.h>
#include<string.h>
int main(){
    int n;
    char s[10010];
    scanf("%d␣%s",&n,s);
    int len=strlen(s);
    for(int i=0;i<len;i++){
        int x=s[i]-'A';
        x=(x+n)%26;
        putchar('A'+x);
    }
    putchar('\n');
}
```

C: Buy an Integer

まず、整数 N の値段を計算するためには N の桁数を求める必要がありますが、これは

- $10^k \leq N < 10^{k+1}$ となる整数 k を見つけると、N は $(k+1)$ 桁の整数である。
- 標準の関数等を使って一旦文字列に変換して長さを求める (Python なら len(str(N)) とする、など)。

などの方法で求めることができます。
しかし、1 から 10^9 (10 億) まで順番に値段を計算して購入できるかを判定しているのでは時間制限に間に合わないので工夫が必要です。
今回の問題では値段に単調性 (大きい整数ほど値段も高い) があるので、以下のように二分探索を利用することができます。

0. 10 億を買うことができるなら 10 億が答えです。以下はそうでないと仮定します。
1. まず 5 億が買えるかどうかを調べます。
2. 買えたとします。この場合、買うことのできる最大の整数は 5 億以上で 10 億より小さいということがわかります。
3. 次に 5 億と 10 億の間である 7 億 5000 万が買えるかどうかを調べる。
4. 今度は買えなかったとします。この場合、買うことのできる最大の整数は 5 億以上で 7 億 5000 万より小さいということがわかります。
5. 次は 5 億と 7 億 5000 万の間である 6 億 2500 万が買えるかどうかを調べます。
6. ...。

このようにすることで、1 回ごとに答えの候補の範囲が半分ずつに縮まっていき、30 回程度の計算で答えを見つけることができます。

D: Coloring Edges on Tree

まず、色数の最小値 K がいくつになるかを考えます。任意の頂点 v について、 $\deg(v) \leq K$ を満たす必要があります ($\deg(v)$ は頂点 v の次数を表します)。

実は、 $K = \max_v \deg(v)$ となるような色の塗り分けが存在します。まず、頂点を一つ選び、それを根として G を根付き木にします。根から幅優先探索を行います。各頂点では、その子との間の辺の色を順番に決めていきます。このとき、その頂点を端点に持つ辺の色としてまだ使われていない色の中で最小のものを使うようにすると、その値は K を超えないことがわかります。

構築を終えた後、出力の際には連想配列などを用いることができます。以下は C++ による解答例です。`https://atcoder.jp/contests/abc146/submissions/8600549`

E: Rem of Sum is Num

数列 $\{S_n\}$ を $S_i = A_1 + A_2 + \cdots A_i$ と定めます。$S_0 = 0$ とします。
すると、連続部分列 $A_{i+1}, \ldots, A_j$ がみたすべき条件を式で表すと、

$$(S_j - S_i)\%K = j - i$$

と書けることがわかります。上の式をさらに変形すると、

$$(S_j - j)\%K = (S_i - i)\%K \text{ かつ } j - i < K$$

と同値です。したがって、各 j $(1 \leq j \leq N)$ について、$(S_j - j)\%K = (S_i - i)\%K$ をみたすような $j - K < i < j$ の個数を数えるといいです。
これをそのまま 1 つずつ調べていては間に合いませんが、j に対して調べるべき区間 $(j - K, j)$ と $j + 1$ に対して調べるべき区間 $(j - K + 1, j + 1)$ は左右が 1 つずつ違うだけなので、j 番目について調べたあと $j + 1$ 番目について調べるには、左の要素を 1 つ捨てて、右の要素を 1 つ加えるだけでよいです。
捨てる、加える などの操作は連想配列 (C++ の map や Python の dict など) を用いて $S_i - i$ たちの個数を管理することで、高速に実現することができます。
計算量は全体で $O(N \log K)$ 程度です。

F: Sugoroku

'1' から成る連続した M マスの区間が存在するときは, 答えは明らかに -1 です.
また, そのような区間が存在しないときは, 高橋君がゴールに到達することは可能です.
以下では, そのような区間が存在しない場合のみ考えます.
この双六において, 移動出来るマスの関係をグラフで表すことを考えます. このとき, この双六は $N+1$ 頂点で, $S_i = 0$ となるような頂点 i から, 頂点 $i+1, i+2, ..., \min(i+M, N+1)$ に有向辺が張られたグラフと見なすことが出来ます.
このグラフ上で, 「各頂点からゴールまで最短何手でゴール出来るか」を求めます. これは, 逆辺を張ったグラフ上で頂点 $N+1$ からの最短路問題を解くことに等しいです.
グラフ上の辺が多い為, 有名な最短路問題アルゴリズム (ダイクストラ法等) では実行時間制限に間に合いませんが, グラフの特殊な形状に着目すると, 区間最小値を求めるデータ構造を使えば良いことが分かります.
或いは, このグラフの特殊な形状により, ($S_i = 1$ となる頂点を無視すれば) 前述の値が頂点番号に対して単調減少になることを利用すると, 「頂点 $N+1$ までの最短路の長さが x であるような頂点のうち, 最も頂点番号が小さいもの」を各 x について記録すれば, 頂点番号を降順に走査するだけで簡単に前述の値を求めることが出来ます.
以上の値を求めることが出来れば, あとは頂点 1 から順に, 頂点 $N+1$ までの最短路の長さが減少するような頂点のうち頂点番号が最も小さいものへ進むことを繰り返せば良いです.
最短路の長さが減少しないような頂点へ移動するとゴールまでの手数が最小とはなりません. また, 頂点番号が最小ではない頂点へ移動してもゴールまでの手数は変わりませんが, 出目を並べた数列の辞書式順序が大きくなります.
C++ による解答例:https://atcoder.jp/contests/abc146/submissions/8592598

A: Can't wait for holiday

Find what day of the week does the given string represent, then split into cases and output the answer. The following is a sample code in C++.

```cpp
#include <iostream>
using namespace std;
int main(){
    string s;
    cin >> s;
    int ans;
    if(s == "SUN")ans = 7;
    else if(s == "MON")ans = 6;
    else if(s == "TUE")ans = 5;
    else if(s == "WED")ans = 4;
    else if(s == "THU")ans = 3;
    else if(s == "FRI")ans = 2;
    else if(s == "SAT")ans = 1;
    cout << ans << endl;
}
```

B: ROT N

First find the 0-indexed index of original character in the alphabet table, then add N and get the remainder divided by 26, then you will find which character to output.

```
#include<stdio.h>
#include<string.h>
int main(){
    int n;
    char s[10010];
    scanf("%d␣%s",&n,s);
    int len=strlen(s);
    for(int i=0;i<len;i++){
        int x=s[i]-'A';
        x=(x+n)%26;
        putchar('A'+x);
    }
    putchar('\n');
}
```

C: Buy an Integer

First, in order to calculate the price of integer N you have to find the number of digits of N. This can be found in such ways like:

- Find a integer k such that $10^k \leq N < 10^{k+1}$, then N has $(k+1)$ digits.
- Convert the integer into a string by using standard function etc., and find its length (use len(str(N)) in Python for example).

However, if you calculate prices of integers from 1 to 10^9 (one billion), it won't finish in time, so you have to think out a way. In this problem the problem has monotonicity (the larger integer has the higher price), so you can apply binary search as follows:

0. If he can buy one billion then one billion is the answer. Assume that he can't.
1. First, check if he can buy 500 million.
2. Assume that he can. Then the maximum integer he can buy is greater than or equal to 500 million and less than one billion.
3. Then check if he can buy 750 million, which is between 500 million and one billion.
4. Assume that, this time, he can't. Then the maximum integer he can buy is greater than or equal to 500 million and less than 750 million.
5. Then check if he can buy 750 million, which is between 500 million and 750 million.
6. ...

This way, the span of answer will be halved in every operation, and the answer can be found in about 30 calculations.

D: Coloring Edges on Tree

First, let's think about the minimum number of colors K. For every vertex v, $\deg(v) \leq K$ should meet (where $\deg(v)$ denotes the degree of vertex v).

In fact, there exists coloring such that $K = \max_v \deg(v)$. First, choose a vertex and let it be the root, thus G will be a rooted tree. Perform a breadth first search from the root. For each vertex, determine the colors of edges between its children one by one. For each edge, use the color with the minimum index among those which are not used as colors of edges whose one of endpoints is the current vertex. Then the each index of color does not exceed K.

After construction ended, you can use associative arrays when outputting. The following is a sample code in C++. `https://atcoder.jp/contests/abc146/submissions/8600549`

E: Rem of Sum is Num

Let's define a sequence $\{S_n\}$ such that $S_i = A_1 + A_2 + \cdots A_i$. Let $S_0 = 0$.
Then, the condition that a contiguous subsequence $A_{i+1}, \ldots, A_j$ should meet can be represented like

$$(S_j - S_i)\%K = j - i.$$

This equation can be transformed to the following equivalent conditions:

$$(S_j - j)\%K = (S_i - i)\%K \text{ and } j - i < K$$

Therefore, for each j $(1 \leq j \leq N)$ you should count the number of $j - K < i < j$ such that $(S_j - j)\%K = (S_i - i)\%K$.
If you search them this naively one by one it won't finish in time, but for j the segment needed to be searched is $(j - K, j)$, and for $j + 1$, it is $(j - K + 1, j + 1)$, and these differs only by one elements at the leftmost and rightmost, so in order to search for $j + 1$-th after searching for j-th element, you only have to discard the leftmost element and add the rightmost element.
Operations of discarding or adding can be performed quickly by managing the number of $S_i - i$'s by using associative arrays (such as map in C++ or dict in Python).
The total time complexity is about $O(N \log K)$.

F: Sugoroku

If there exists a segment of cells consisting from '1' of length M, then the answer is obviously -1.

On the other hand, if there does not exist such a segment, then it is possible for Takahashi to reach the goal.

Hereafter we will assume that there does not exist such a segment.

Let's think of representing the available moves in this Sugoroku with a graph. Then we can regard this Sugoroku as a graph of $N+1$ vertices, which has a directed edge from every vertex i such that S_i to vertices $i+1, i+2, ..., \min(i+M, N+1)$.

In this graph, let's find "how many moves are needed to reach the goal from each vertices." This is equivalent to solving shortest path problem on the graph whose vertices are all reversed.

Since there are so many edges, famous shortest-path-finding algorithms won't finish in time, but due to the special pattern of the graph, it appears that you can use a data structure of finding minimum value in the segment.

Or, due to the special pattern of the graph, the values mentioned above is in strictly decreasing order by the indices of the vertices (except for such vertices that $S_i = 1$), so you can find the value mentioned above easily by iterating through the vertices in the decreasing order of their indices, while recording "the vertex having the minimum index among such vertices that the length of the shortest path to the vertex $N+1$ is x" for each x.

Now that you obtained the values mentioned above, all you have to do is move to the vertex with the smallest index among such vertices that length of shortest path to vertex $N+1$ decreases, one after another from vertex 1.

If you move to the vertex such that length of shortest path does not decrease, then the number of moves will not be minimized. If you move to the vertex such that its index is not smallest possible, the number of moves needed does not change, but the sequence of numbers coming up in the roulette will be lexicographically older.

Sample code in C++:https://atcoder.jp/contests/abc146/submissions/8592598