ABC055 / ARC069 解説

writer : camypaper

2017 年 2 月 18 日

A : Restaurant

「何回 200 円もらえるか？」というのが本質的な問題です．$n/15$ とすることで C++ などのプログラミング言語では n を 15 で割った値の小数点以下切り捨てを求めることが可能です．$200 * n/15$ としてしまうと、正しい値を求めることができないことにも注意が必要です．

```
#include <iostream>
using namespace std;
int main(){
    int n;
    cin >> n;
    cout<<n*800-(n/15)*200<<endl;
}
```

B : Training Camp

「$N!$ を $10^9 + 7$ で割ったあまりを求めよ」という問題です．$N!$ そのものを求めようとすると、この値は非常に値が大きくなるため 64 ビット符号付き整数ではオーバーフローによって正しく値を求められなかったり，多倍長整数の乗算計算は時間がかかるため実行時間制限内に計算を終えることが難しくなります．

求める必要があるのは「$10^9 + 7$ で割ったあまり」であることに着目すると，現在のパワーを x として $x = i \times x \bmod 10^9 + 7$ として更新していけば、64 ビット符号付整数で収まる範囲で計算をすることが可能です．32 ビット符号付整数で扱っている場合 $x \times i$ が 10^9 より大きくなってしまいオーバーフローしてしまうことに注意してください．

C : Scc Puzzle

「S 字型のピースと c 字型のピースを組み合わせて Scc という組を可能な限り多く作りなさい」という問題です．ただし，c 字型のピースを 2 つ組み合わせて，S 字型のピースを 1 つ作ることが可能です．このままでは複雑なので，可能な操作を以下の 2 種類だと考えます．

1. S 字型のピース 1 つと c 字型のピース 2 つを組み合わせて Scc の組を 1 つ作る
2. c 字型のピース 4 つを組み合わせて Scc の組を 1 つ作る

なぜ，このように言い換えていいかを考えてみましょう．S 字型のピースが残っているにも関わらず，c 字型のピースを 2 つを組み合わせて S 字型のピースを作る必要がないのは直感的にも明らかでしょう．すると，S 字のピースがないときのみ c 字型のピース 2 つを使って S 字型のピースを作る，という操作が行われます．その直後の操作はやはり，S 字型のピース 1 つと c 字型のピース 2 つを組み合わせて Scc の組を作る，という操作になります．この 2 つの操作を 1 つにまとめると，「c 字型のピース 4 つを組み合わせて Scc の組を 1 つ作る」操作だとみなすことができます．

上記のような 2 種類の操作であることがわかると，可能な限り 1 番の操作を行ったのち，可能な限り 2 番の操作を行うというのが最適なことが分かります．このようにして作ることができる Scc の組の数は簡単な四則演算により $O(1)$ で答えを求めることが可能です．

D : Menagerie

「円環状に並んでいる N 匹の動物たちに両隣の動物が同じ種類かどうか訪ねた結果と矛盾しないような動物の割当てが存在するか？」という問題です．動物は羊と狼の 2 種類だけですが，羊は本当のことを言うのに対して狼は嘘をつくのが問題を複雑にしています．

この問題の重要な性質は「ある連続した 2 匹の種類が分かれば，その隣にいる動物の種類も分かる」ということです．例えば $i-1$ 番と i 番の動物の種類がそれぞれ分かっているとすると，i 番の言っていることが本当か嘘かがまず分かります．次に $i-1$ 番の種類が分かっているので，$i+1$ 番の種類がどちらか分かる，ということです．さらに，i 番と $i+1$ 番の種類が分かっているので $i+2$ 番の種類が分かり，$i+1$ 番と $i+2$ 番の種類が分かっているので $i+3$ 番の種類が分かり，というように連鎖的に動物の種類が分かります．

上の性質から 1 番と 2 番の動物の種類を仮定すると $3, 4, 5, \ldots, N$ 番の動物の種類が連鎖的に定まります．最後にこの仮定が正しかったかどうかを s と矛盾しないかどうかで調べればよいです．試す候補は高々 4 通りなので $O(N)$ でこの問題を解くことが可能です．

E : Frequency

「N 個の石の山からあるルールで石の山を取り除いて長さ $\sum a_i$ の数列 s を辞書順最小になるように構成するとき，s に $1, 2, 3, \ldots, N$ がそれぞれ何回含まれるか？」という問題です．石を 1 個取り除く山を選ぶときのルールは「(a_i, i) が最大であるような i を選ぶ」としてよいです (以下，これを単にルールと呼びます).

ルールに従ったとき s が辞書順最小の数列となることを示します．x を石の数が最大である山のうち最も左側にある石の山の番号とします.

$i < x$ かつ $0 < a_i < a_x$ であるような i が存在しない場合を考えます．x 番以外の山の石が取り除かれたとき，次の操作において s の末尾に追加される数は明らかに x です．x 番の石の山から石が取り除かれたとき，その他の石の山は全て $a_x - 1$ 個以下の石からなります．さらに x 番より左側には石の数が 0 個の山しか存在しないので，石を 1 個取り除いた後も石の数最大であって最も左側にある石の山は x 番です．よって，このときも次の操作において s の末尾に追加される数は x です ($a_x = 0$ となった場合は処理が終了するため，考慮する必要はありません)．結果として，x のみからなる数列が得られます．x より小さな数が s に追加されることはないため，明らかに辞書順最小です.

$i < x$ かつ $0 < a_i < a_x$ であるような i が存在している場合を考えます．y を $i < x$ において (a_i, i) が最大であるような i とします．このとき，辞書順最小の数列を得るためには x より大きな値が s に追加されてはならないようにいくつかの石を取り除いて y が可能な限り早く s に現れるようにする必要があります．これを，石の数を $\max(0, a_i - a_y)$ としたような石の山たちに対して辞書順最小の数列を作っていくと考えると，x 番より左側には 0 個の山しかないという状況に帰着させることが可能です．こうして石を取り除いたあと，y 番の山が石の数最大であって最も左側にある石の山となります．再帰的にこのような処理を行っていけばいつか全ての石がなくなります.

以上より，(a_i, i) が最大であるような山から石を取り除く，という操作を繰り返していけば s を辞書順最小の数列にすることが可能なことが示されました.

ルールを愚直にシミュレーションすると実行時間制限に間に合わないので高速化する必要があります．石の数が同じ石の山をまとめる，石をまとめて取り除く，という 2 つの高速化を行った以下のアルゴリズムにより $O(N \log N)$ で解くことが可能です.

- 石の山と石の番号 (a_i, i) で降順にソートしたものを (x_i, y_i) とする
- (ソートしたあとの並びで) 1 番目から順に処理を行う
- 今 i 番目の石の山に着目しているとして，$i(x_i - x_{i+1})$ 個の石を取り除き，s に $\min\{y_1, y_2, \ldots, y_i\}$ を $i(x_i - x_{i+1})$ 個追加する (便宜上 $x_{N+1} = 0$ とする)

これは i 番目の山を見ている時点で $1, 2, 3 \ldots, i$ 番の山に含まれる石の数が等しく，i 個の石を取り除く，という操作を $x_i - x_{i+1}$ 回行っていると考えると分かりやすいかもしれません.

F: Flags

「i 番の旗を x_i か y_i のどちらかに設置する，という操作を N 回行ったときの旗同士の距離の最小値を最大化せよ」という問題です．求める答えが d だとして，旗同士の距離が $d-1, d-2, \ldots, 0$ となるように設置することが可能であったとみなしても問題ありません．そこで二分法を用いて d を求めていくことを考えます g．d を固定したとき，この問題は以下のように表せます．

> N 本の旗があり，i 番の旗は x_i か y_i のどちらかに設置しなくてはならない．どの 2 つの旗も距離が d 以上になるように設置することは可能か判定せよ．

2 つの旗の距離が d 未満にならないかどうかにのみ着目すればよくなり，見通しがよくなりました．旗同士の距離は 2 つの旗の位置関係のみに依存しており，その他の旗は関係ないことは明らかです．すると「i 番の旗を x_i に置いたとき，j 番の旗は y_j に置いてはならない」というような制約が最大 $O(N^2)$ 個与えられるので，条件を満たすように旗を設置することが可能か？という問題になります．これを 2-SAT に帰着させて解いていきます．

まず「i 番の旗を x_i と y_i のどちらに置くか」という N 個の変数に関する 2-SAT ではなく，z を x と y を連結して昇順に並び替えた数列として「i 番の旗を座標 z_i に設置するかどうか」という $2N$ 個の変数に関する 2-SAT として考えます．このようにすると z は昇順の数列となっているため「距離 d 未満に別の旗があってはならない」という制約が扱いやすくなります．2-SAT として考えやすいよう，問題を以下のように言い換えます．

> 真偽値をとる論理変数 $v_1, v_2, \ldots, v_{2N}$ と以下に示されるようないくつかの論理式が与えられたとき，全ての論理式が真となるような真偽値の割り当てが存在するか判定せよ．
>
> - $\neg(v_i \wedge v_j), \neg(\neg v_i \wedge \neg v_j)$ (ただし i, j は旗としての対応関係がある)
> - $\neg(v_i \wedge v_l), \neg(v_i \wedge v_{l+1}), \ldots, \neg(v_i \wedge v_{i-1})$ (l は $|z_i - z_l| < d$ となる最小の l)
> - $\neg(v_i \wedge v_{i+1}), \neg(v_i \wedge v_{i+2}), \ldots, \neg(v_i \wedge v_r)$ (r は $|z_i - z_r| < d$ となる最大の r)

このままでは最大で $O(N^2)$ 個程度の節からなる 2-SAT となってしまいます．ここで，v_i が真のとき，偽となる必要がある論理変数たちが区間をなすことに着目します．論理変数をさらに 2 倍程度の個数になるよう増やし，旗であった論理変数と葉が対応するように論理変数をセグメント木状に配置し，以下のルールで節を作ることで変数の数を $O(N)$，節の数を $O(N \log N)$ に抑えることが可能です．

1. セグメント木の葉であるような i と旗の対応関係がある j について $\neg(v_i \wedge v_j), \neg(\neg v_i \wedge \neg v_j)$
2. セグメント木上で親の頂点の番号を p として $(v_i \to v_p)$
3. $[l, i), [i+1, r+1)$ をセグメント木上で $O(\log N)$ 個の頂点 $u_1, u_2, \ldots, u_k$ に分割したとして $\neg(v_i \wedge u_1), \neg(v_i \wedge u_2), \ldots, \neg(v_i \wedge u_k)$

図 1 にルール 2, 3 の具体例として v_{26} に着目した場面を示します．v_{26} に真を割り当てるとき，ルール 2 から先祖である赤色の頂点たちもまた真が割り当てられる必要があります．この「ある頂点に真が割り当てられたならば，その先祖たちは全て真」という性質を「ある頂点に偽が割り当てられたならば，その子孫たち全てもまた偽が割り当てられている」という性質だと考えると，ルール 3 のように図中の濃い青色の頂点 $v_{19}, v_5, v_{12}, v_{27}, v_{28}$ に対して節を追加することで「v_{26} に真が割り当てられたとき，$v_{19}, v_{20}, \ldots, v_{25}, v_{27}, v_{28}$ は偽が割り当てられる必要がある」という条件を $O(\log N)$ 個の節で表すことが可能となることが分かります．

変数の数 $O(N)$，節の数 $O(N \log N)$ の 2-SAT は 1 回あたり $O(N \log N)$ で調べることが可能なので，二分法と合わせて $O(N \log N \log \max z_i)$ で解くことが可能となります．

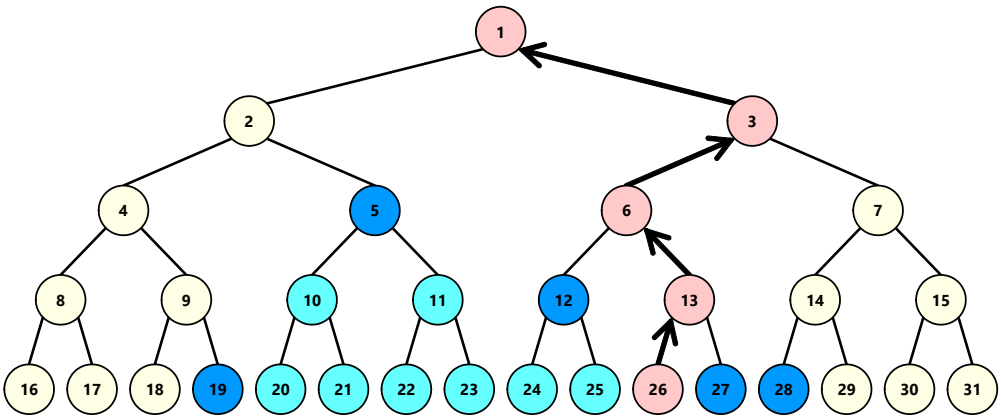

図 1　v_{26} に着目したときの節の追加ルール