M-SOLUTIONS プロコンオープン 解説

writer: yokozuna_57

2019 年 6 月 1 日

For International Readers: English editorial starts on page 7.

A: Sum of Interior Angles

正 N 角形の内角の和は $180(N-2)^\circ$ です。以下に C++での実装例を示します。

```
#include <iostream>
using namespace std;

int main(){
        int N;
        cin >> N;
        cout << (N-2)*180 << endl;
}
```

B: Sumo

高橋君が残りの試合すべてに勝ったときに 8 勝以上できるならば、高橋君は次の大会にも参加できる可能性があります。逆に、高橋君が次の大会にも参加できる可能性があるならば、高橋君が残りの試合すべてに勝ったときに 8 勝以上できます。したがって、残りの試合すべてに勝ったときに 8 勝以上できるかを確認すればよいですが、これは k 試合目までに 8 敗以上していないことと同値です。以上より、文字列 S に現れる'x' の数を数えて、7 以下なら"YES"を、8 以上なら"NO"を出力すればよいです。以下に C++での実装例を示します。

```
#include <iostream>
#include <string>
using namespace std;

int main(){
        string s;
        cin >> s;
        int cnt = 0;
        for(int i = 0 ; i < s.size() ; i ++){
                if(s[i] == 'x')cnt ++;
        }
        if(cnt <= 7)puts("YES");
        else puts("NO");
}
```

C: Best-of-(2n-1)

まずは引き分けがない場合を考えます。$X(M)$ でゲームがちょうど M 回行われる確率を表すとします。$M \leq N-1, 2N \leq M$ のときは $X(M) = 0$ です。そうでないときゲームがちょうど M 回行われるのは、$M-1$ 回目までで高橋君がちょうど $N-1$ 勝し、M 回目のゲームでも高橋君が勝つ場合と $M-1$ 回目までで青木君がちょうど $N-1$ 勝し、M 回目のゲームでも青木君が勝つ場合です。したがって、

$$X(M) = \binom{M-1}{N-1}\left(\left(\frac{A}{100}\right)^N\left(\frac{B}{100}\right)^{M-N} + \left(\frac{A}{100}\right)^{M-N}\left(\frac{B}{100}\right)^N\right)$$

となります。

引き分けについて考えます。引き分けでないゲームが M 回行われるまでに (引き分けを含めて) ゲームが行われる回数の期待値を $Y(M)$ とすると、$Y(M) = M \cdot \dfrac{100}{100-C}$ となります。これは直感的には明らかですが、次のようにして証明することもできます。

> $Y(0) = 0$ である。M を正の整数とする。1 回目が引き分けであれば残りの試合の期待値は $Y(M)$ であり、1 回目が引き分けでなければ残りの試合の期待値は $Y(M-1)$ である。したがって、任意の正の整数 M について、
>
> $$\begin{aligned} Y(M) &= 1 + \frac{C}{100}Y(M) + \frac{100-C}{100}Y(M-1) \\ &\Leftrightarrow \frac{100-C}{100}Y(M) = \frac{100-C}{100}Y(M-1) + 1 \\ &\Leftrightarrow Y(M) = Y(M-1) + \frac{100}{100-C} \end{aligned}$$
>
> となるから、$Y(M) = M \cdot \dfrac{100}{100-C}$ である。

以上より、答えは

$$\begin{aligned} \sum_{M=N}^{2N-1} X(M)Y(M) &= \sum_{M=N}^{2N-1} \binom{M-1}{N-1}\left(\left(\frac{A}{100}\right)^N\left(\frac{B}{100}\right)^{M-N} + \left(\frac{A}{100}\right)^{M-N}\left(\frac{B}{100}\right)^N\right) M \cdot \frac{100}{100-C} \\ &= \sum_{M=N}^{2N-1} \binom{M-1}{N-1} \frac{(A^N B^{M-N} + A^{M-N} B^N)M}{100^{N-1}(100-C)} \end{aligned}$$

となります。表式に現れる二項係数と累乗はすべて $O(N)$ で前計算できます。

D: Maximum Sum of Minimum

あらかじめソートすることで、$c_1 \geq c_2 \geq \cdots c_N$ としてよいです。k 本の T の辺を選んで、それらの辺とその端点からなる T の部分グラフを U とします。U は森になるので、$k+1$ 個以上の頂点をもちます。したがって、k 本の辺に書き込まれる整数のうち最小のものは c_{k+1} 以下になります。この考察により、各辺に書き込まれる整数を $x_1 \geq x_2 \geq \cdots x_{N-1}$ とすると、$x_i \leq c_{i+1}$ となることがわかります。したがって、スコアは常に $c_2 + c_3 + \cdots + c_N$ 以下です。

逆に、次のように正の整数を書き込むことでスコアが $c_2 + c_3 + \cdots + c_N$ になります。

- T の辺を 1 つ選んで (e_1 とします) 両端に c_1, c_2 をそれぞれ書き込みます。
- $k = 2, 3, \ldots, N-1$ に対して次の操作を繰り返します。$e_1, e_2, \ldots, e_k$ とその端点からなる T の部分グラフが連結になるように T の辺 e_k を選ぶことができます。このとき、e_k の端点のうち一方にはまだ整数が書き込まれていないのでその端点に c_{k+1} を書き込みます。

スコアを求める際には、辺 e_i にはそれぞれ c_{i+1} が書き込まれます。それぞれの辺を選ぶ際にまだ使っていないすべての辺を調べても $O(N^2)$ なので、このアルゴリズムをそのまま実装すればよいです。

E: Product of Arithmetic Progression

すべてのクエリが $d = 1$ をみたすときは簡単に解けます。

この場合、積 $x(x+1)...(x+n-1)$ を求めたいです。これは、 x と $x+n-1$ のあいだに 1000003 の倍数がある場合は 0, そうでない場合は $(x+n-1)!/(x-1)!$ なので、階乗とその逆元を前計算しておくことにより各クエリに $O(1)$ で答えることができます。

一般の場合は、まず $d = 0$ の場合を例外処理しておきます (x^n). そうでないとき、すべての項を d で割ると差が 1 の等差数列を得ることができます。

$x/d, x/d+1, ..., x/d+(n-1)$

答えは、これらの積 (これは上に書いた方法で求められます) を d^n 倍したものです。

F: Random Tournament

$1 \leq i < j \leq N$ に対して、

$$dpl[i][j] := \text{人 } i, \text{人 } i+1, \ldots, \text{人 } j \text{ のみを考えたとき、人 } i \text{ が優勝する可能性がある。}$$
$$dpr[j][i] := \text{人 } i, \text{人 } i+1, \ldots, \text{人 } j \text{ のみを考えたとき、人 } j \text{ が優勝する可能性がある。}$$

とします。このとき、人 i が優勝する可能性があることは $dpl[i][N] = dpr[i][1] = \text{true}$ と同値です。より一般には、人 i, 人 $i+1$, …, 人 j のみを考えたとき、人 k $(i \leq k \leq j)$ が優勝する可能性があることは $dpl[k][j] = dpr[k][i] = \text{true}$ と同値です。

$dpl[i][j]$ の遷移について考えます。まず、人 i が人 k に勝ち、$dpl[k][j] = dpr[k][i+1] = \text{true}$ となるような k $(i \leq k \leq j)$ が存在したとき、人 k は人 $i+1$, 人 $i+2$, …, 人 j のみを考えたときに優勝する可能性があるため、$dpl[i][j] = \text{true}$ です。逆に、$dpl[i][j] = \text{true}$ となるとき、このような k が存在することを示します。人 i, 人 $i+1$, …, 人 j のみを考え、人 i が優勝するような順番に従って試合を行ったとします。さらに、人 i が試合で勝った相手を人 k_1, 人 k_2, …, 人 k_t $(k_1 < k_2 < \cdots < k_t)$ とします。このとき、人 i の試合をすべてとばして大会を行った後、人 k_1 と人 k_2 の試合、その勝者と人 k_3 の試合、…、その勝者と人 k_t の試合を行い、最後の試合の勝者と人 i が試合をすれば人 i は 1 試合のみで優勝できます。さらに、最後の試合の勝者の番号を k とすれば、この k は先ほどの条件をみたします。

以上より、$dpl[i][j]$ の遷移は

$$dpl[i][j] = \exists k \in [i+1, j], A_{i,k} = 1 \wedge dpl[k][j] = \text{true} \wedge dpr[k][i+1] = \text{true}$$

となります。$dpr[j][i]$ も同様に

$$dpr[j][i] = \exists k \in [i, j-1], A_{k,i} = 0 \wedge dpl[k][j-1] = \text{true} \wedge dpr[k][i] = \text{true}$$

となります。$j - i$ の小さい方から順に求めることで、$O(N^3)$ の時間計算量で求められます。

最後に、このような計算は bitset を用いて並列化することができます。

M-SOLUTIONS Programming Contest Editorial

writer: yokozuna_57

June 1, 2019

A: Sum of Interior Angles

The sum of the interior angles of a regular polygon with N sides is $180(N-2)^\circ$. A sample implementation in C++ follows:

```
#include <iostream>
using namespace std;

int main(){
        int N;
        cin >> N;
        cout << (N-2)*180 << endl;
}
```

B: Sumo

If Takahashi can have 8 or more wins when he wins all the remaining matches, there is a possibility that he can participate in the next tournament. Conversely, if there is a possibility that Takahashi can participate in the next tournament, he can have 8 or more wins when he wins all the remaining matches. Thus, what we need to do is to check if he can have 8 or more wins when he wins all the remaining matches, which is equivalent to having less than 8 losses in the first k matches. Therefore, we can solve the problem by counting the occurrences of `x` in the string S, print `YES` if there are 7 or less `x` and print `NO` if there are 8 or more. A sample implementation in C++ follows:

```
#include <iostream>
#include <string>
using namespace std;

int main(){
        string s;
        cin >> s;
        int cnt = 0;
        for(int i = 0 ; i < s.size() ; i ++){
                if(s[i] == 'x')cnt ++;
        }
        if(cnt <= 7)puts("YES");
        else puts("NO");
}
```

C: Best-of-(2n-1)

Let us first consider the case with no draws. Let $X(M)$ denote the probability that the game is played exactly M times. If $M \leq N-1$ or $2N \leq M$, $X(M) = 0$. Otherwise, the game will be played exactly M times if Takahashi has exactly $N-1$ wins in the first $M-1$ games and he also wins the M-th game or Aoki has exactly $N-1$ wins in the first $M-1$ games and he also wins the M-th game. Thus,

$$X(M) = \binom{M-1}{N-1}\left(\left(\frac{A}{100}\right)^N\left(\frac{B}{100}\right)^{M-N} + \left(\frac{A}{100}\right)^{M-N}\left(\frac{B}{100}\right)^N\right)$$

Now, let us consider the case involving draws. Let $Y(M)$ be the expected number of games played until there are M non-draw games, including draw games. Then, $Y(M) = M \cdot \dfrac{100}{100-C}$. This is intuitive, and we can prove it as follows.

$Y(0) = 0$ holds. Let Mbe a positive integer. If the first game is drawn, the expected number of games played from now on is $Y(M)$; If the first game is not drawn, the expected number of games played from now on is $Y(M-1)$. Thus, for any positive integer M,

$$\begin{aligned} Y(M) &= 1 + \frac{C}{100}Y(M) + \frac{100-C}{100}Y(M-1) \\ &\Leftrightarrow \frac{100-C}{100}Y(M) = \frac{100-C}{100}Y(M-1) + 1 \\ &\Leftrightarrow Y(M) = Y(M-1) + \frac{100}{100-C} \end{aligned}$$

It follows that $Y(M) = M \cdot \dfrac{100}{100-C}$.

Therefore, the answer is

$$\begin{aligned} \sum_{M=N}^{2N-1} X(M)Y(M) &= \sum_{M=N}^{2N-1} \binom{M-1}{N-1}\left(\left(\frac{A}{100}\right)^N\left(\frac{B}{100}\right)^{M-N} + \left(\frac{A}{100}\right)^{M-N}\left(\frac{B}{100}\right)^N\right) M \cdot \frac{100}{100-C} \\ &= \sum_{M=N}^{2N-1} \binom{M-1}{N-1} \frac{(A^N B^{M-N} + A^{M-N} B^N)M}{100^{N-1}(100-C)} \end{aligned}$$

We can precompute the binomial coefficients and powers in the formula in $\mathrm{O}(N)$ time.

D: Maximum Sum of Minimum

We can assume that $c_1 \geq c_2 \geq \cdots c_N$ by sorting them in advance. Let us choose k of the edges in T, and let U be the subgraph of T consisting of those edges and their endpoints. Since U is a forest, it has $k+1$ or more vertices. Thus, the minimum among the integers written on the k edges is c_{k+1} or smaller. It follows from this observation that, if we let $x_1 \geq x_2 \geq \cdots x_{N-1}$ be the integers written on the edges in T, $x_i \leq c_{i+1}$ holds. Thus, the score never exceeds $c_2 + c_3 + \cdots + c_N$.

Conversely, we can make the score $c_2 + c_3 + \cdots + c_N$ by writing the integers as follows:

- Choose an edge e_1 in T and write c_1 and c_2 on its endpoints.
- Do the following operation for $k = 2, 3, \ldots, N-1$: We can choose an edge e_k in T so that the subgraph of T consisting of $e_1, e_2, \ldots, e_k$ and their endpoints. One of the endpoints of such an edge e_k is still empty, and we write c_{k+1} on that endpoint.

Then, c_{i+1} will be written on the edge e_i when the score is evaluated. Since it takes only $\mathrm{O}(N^2)$ time to check all the edges each time we choose an edge, so we can naively implement this algorithm.

E: Product of Arithmetic Progression

Notice that if all queries satisfy $d = 1$, we can easily solve the problem.

In this case, we want to compute the product $x(x+1)...(x+n-1)$. If there is a multiple of 1000003 between x and $x+n-1$, inclusive, the answer is zero. Otherwise, the answer is $(x+n-1)!/(x-1)!$, and by precomputing factorials (and their inverses) we can answer each query in $O(1)$.

How to solve the problem in general cases? In case $d = 0$, the answer is x^n. Otherwise, notice that if we divide each term by d, we get an arithmetic progression with difference 1:

$x/d, x/d+1, ..., x/d+(n-1)$

Thus, the answer is the product of these n terms (which can be computed in the way described above) times d^n.

F: Random Tournament

For $1 \leq i < j \leq N$, let

$dpl[i][j]$:= whether Person i may become the champion when only Person i, Person $i+1$, ..., Person j are considered

$dpr[j][i]$:= whether Person j may become the champion when only Person i, Person $i+1$, ..., Person j are considered

Then, Person i may become the champion if and only if $dpl[i][N] = dpr[i][1] = \text{true}$. More generally, when only Person i, Person $i+1$, ..., Person j are considered, Person k ($i \leq k \leq j$) may become the champion if and only if $dpl[k][j] = dpr[k][i] = \text{true}$.

Let us consider the transition of $dpl[i][j]$. First, if there exists k ($i \leq k \leq j$) such that Person i defeats Person k and $dpl[k][j] = dpr[k][i+1] = \text{true}$, Person k may become the champion when only Person $i+1$, Person $i+2$, ..., Person j are considered, so $dpl[i][j] = \text{true}$. Conversely, we will show that such k exists if $dpl[i][j] = \text{true}$. Let us only consider Person i, Person$i+1$, ..., Person j, and assume that they played matches so that Person i becomes the champion. Additionally, let Person k_1, Person k_2, ..., Person k_t ($k_1 < k_2 < \cdots < k_t$) be the persons defeated by Person i. Then, if we first play all the matches not involving Person i, then play a match between Person k_1 and Person k_2, then between the previous winner and Person k_3, ..., then between the previous winner and Person k_t, then between the previous winner and Person i, Person i can become the champion by just playing one match. Additionally, if we let Person k be the winner of the second last match, this k satisfies the condition above.

Therefore, the transition of $dpl[i][j]$ is:

$$dpl[i][j] = \exists k \in [i+1, j], A_{i,k} = 1 \wedge dpl[k][j] = \text{true} \wedge dpr[k][i+1] = \text{true}$$

Similarly, the transition of $dpr[j][i]$ is:

$$dpr[j][i] = \exists k \in [i, j-1], A_{k,i} = 0 \wedge dpl[k][j-1] = \text{true} \wedge dpr[k][i] = \text{true}$$

By finding the values in ascending order of $j - i$, we can compute the whole tables with time complexity $O(N^3)$.

Finally, we can do such a computation in parallel with bitsets.