ABC 117 解説

writer: drafear

2019 年 2 月 3 日

A: Entrance Examination

答えを t とすると、問題文より $X \times t = T$ が成り立つので、これを解くと $t = \frac{T}{X}$ が得られます。これを C++ で実装すると次のようになります。小数点以下 3 桁以上出力しなければならないことに注意してください。

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
  int T, X; cin >> T >> X;
  double ans = double(T) / X;
  cout << fixed << setprecision(10) << ans << endl;
}
```

B: Polygon

最大の L_i が他の L_i の和未満であれば ‘Yes‘、そうでなければ ‘No‘ を出力すれば良いことが問題文の定理からわかります。したがって、「最大の L_i」と「他の L_i の和」を求められれば判定することができます。これらを求めるアプローチはいくつかあります。

方法 1

L を昇順にソートします。すると、始めの $N-1$ 要素の和が「他の L_i の和」、最後の要素が「最大の L_i」となります。

方法 2

L_j が最大となる j を求めます。これは、仮に $j := 1$ として、$L_2 > L_j$ ならば $j := 2$ と更新し、$L_3 > L_j$ ならば $j := 3$ と更新し、・・・と繰り返すことで求められます。すると、$j \neq k$ なる k について L_k を足し合わせたものが「他の L_i の和」、L_j が「最大の L_i」となります。

方法 3

まず、最大値を求めます。これは、方法 2 やそれと似た方法でできます。次に $L_1, L_2, ..., L_N$ の合計を求め、S とします。すると、$S - L_j$ が「他の L_i の和」、L_j が「最大の L_i」となります。

これを C++ で実装すると、次のようになります。

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
  int n; cin >> n;
  int sum = 0, maxL = 0;
  for (int i = 0; i < n; ++i) {
    int L; cin >> L;
    sum += L;
    maxL = max(maxL, L);
  }
  if (maxL < sum - maxL) {
    cout << "Yes" << endl;
  }
  else {
    cout << "No" << endl;
  }
}
```

C: Streamline

コマの数が目的地の数以上の場合、すなわち $N \geq M$ の場合、初めに各目的地に 1 つ以上のコマを配置できるので 0 回の移動で目的を達成できます。以降、$N < M$ として考えます。さらに、一般性を失わずに $X_1 < X_2 < ... < X_M$ とソートされていると考えて良いです (実装では、入力された座標をソートします)。

ある 1 つのコマに着目したとき、移動を通して訪れる座標は区間になります。逆に、ある区間の整数座標全てをある 1 つのコマで訪れる場合、区間の一端に最初に配置し、もう一端に向かって移動し続けるのが最適です。また、複数のコマが同じ座標を訪れるのは無駄です。したがって、各コマが担当する区間を並べると、ちょうど $N-1$ 個の開区間 (X_i, X_{i+1}) が訪れられていない形になります。$L_i = X_{i+1} - X_i$ とすると、そのときの合計移動回数は $(X_M - X_1) -$ (訪れられていない開区間の L_i の和) となるため、(訪れられていない開区間の L_i の和) を最大化すれば良いです。これは、L_i を大きい方から $N-1$ 個取るのが最大です。計算量はソートがボトルネックとなり $O(MlogM)$ です。

D: XXOR

$0 \leq X \leq K$ なる X 全てについて $f(X)$ を計算し、最大値を出力する方法では、$O(NK)$ となり計算に 10 年以上かかりそうです。そこで、より高速に求める必要があります。

$X < K+1$ なので $K+1 = (K_{39}K_{38}...K_0)_2, X = (X_{39}X_{38}...X_0)_2$ と 2 進数で表したとき[*1]に、

$$\begin{gathered} X_{39} = K_{39} \\ \vdots \\ X_{i+1} = K_{i+1} \\ X_i < K_i \end{gathered}$$

を満たす i が存在します。このとき、$X_i = 0, K_i = 1$ です。

逆に、ある i について

$$\begin{gathered} X_{39} = K_{39} \\ \vdots \\ X_{i+1} = K_{i+1} \\ X_i < K_i \end{gathered}$$

であるような X は、$X_{i-1}, X_{i-2}, ..., X_0$ の値に関わらず $X < K$ が成り立ちます。

そこで、このような $i (0 \leq i < 40)$ を全探索することを考えます。各 i に対する $f(X)$ の最大値を a_i とすれば、解は $\max\{a_0, a_1, ..., a_39\}$ になります。

各 i に対する $f(X)$ の最大値を考えます。$X_{39}, X_{38}, ..., X_i$ は条件より

$$\begin{gathered} X_{39} = K_{39} \\ \vdots \\ X_{i+1} = K_{i+1} \\ X_i = 0 \end{gathered}$$

です。上でも述べたとおり $X_{i-1}, X_{i-2}, ..., X_0$ を決める際には $X < K$ を無視しても構いません。f はビットごとに考えることができるため、$X_{i-1}, X_{i-2}, ..., X_0$ を独立に決めることができます。具体的には、$X_k (0 \leq k \leq i-1)$ について、$A_1, A_2, ..., A_N$ の中で下から k ビット目が立っているものの数を c_k とすると、$X_k = 0$ とすればこのビットが f に貢献する値は $2^k c_k$ となり、$X_k = 1$ とすれば $2^k(n - c_k)$ となります。したがって、$c_k > n - c_k$ の場合は $X_k = 0$ とし、そうでない場合は $X_k = 1$ とするのが最適です。これを i の小さい順または大きい順に探索することで、X を高々 2 ビットずつ変化させて探索でき、$O(NlogK)$ で求めることができます。

別解として、X の上位ビットから順に決める桁 DP による $O(NlogK)$ の解法もあります。

[*1] 制約 $X, K \leq 10^{12} \leq 2^{40}$ より X, K は 40 ビットで表せます。