ABC 057 解説

writer: Hec

2017 年 3 月 26 日

A: Remaining Time

2 つの整数 A と B を入力として受け取ります。基本的に、コンテストの開始時刻は、$A + B$ 時となります。$A + B < 24$ の場合は当日の時刻ですが、$A + B \geq 24$ の場合は翌日の時刻となるため、$A + B$ から 24 を引く必要があります。また、$0 \leq B \leq 23$ よりコンテストの開始時刻が翌々日以降の時刻になることはないのでこれで十分です。なお、時刻を 24 で割った余り (mod 24) として扱うと処理が簡単になります。

C++のコード例 (24 で割った余りを利用)

```
int main(void){
    int A,B;
    cin >> A >> B;
    cout << (A+B)%24 << endl;
    return 0;
}
```

B: Checkpoints

各学生がどのチェックポイントに移動するかを調べます。まず、全てのチェックポイント間のマンハッタン距離を計算します。多くのプログラミング言語では、絶対値を求める処理はライブラリで提供されています。次に、マンハッタン距離が最小となるチェックポイントの番号を求めていきます。これはマンハッタン距離の最小値とそのチェックポイントの番号を管理する 2 つの変数を用いて、ループで求めることが可能です。以上のことから、時間計算量は $O(NM)$ となり、これは間に合います。

C++のコード例

```
int main(void){
    int N,M;
    cin >> N >> M;

    const int NMMAX=50;
    int A[NMMAX],B[NMMAX],C[NMMAX],D[NMMAX];

```

```
    for(int i=0;i<N;++i){
        cin >> A[i] >> B[i];
    }

    for(int j=0;j<M;++j){
        cin >> C[j] >> D[j];
    }

    for(int i=0;i<N;++i){
        int min_dist=abs(A[i]-C[0])+abs(B[i]-D[0]),checkpoint=1;

        for(int j=1;j<M;++j){
            const int cur_dist=abs(A[i]-C[j])+abs(B[i]-D[j]);
            if(min_dist>cur_dist){
                min_dist=cur_dist;
                checkpoint=j+1;
            }
        }

        cout << checkpoint << endl;
    }

    return 0;
}
```

C: Digits in Multiplication

まず、$F(A,B)$ について考えます。「A の桁数と B の桁数のうち大きい方」とは、$M=\max(A,B)$ の桁数です。整数 M の桁数は次のような方法で求められます。

- M を 10 進表記の文字列に変換したときの文字列の長さ
- M を 0 になるまで 10 で繰り返し割った時の回数

次に、正の整数 N に $N=A\times B$ を満たす 2 つの正の整数の組 (A,B) について考えます。このとき、A,B はともに N の約数です。そこで、N の約数を全列挙することを考えます。考えられる方法としては $1\leq A\leq N$ を満たす A についてループを回して、N が A で割り切れるかを試します。しかし、この方法では時間計算量が $O(N)$ となるため間に合いません。そこで、交換法則 $N=A\times B=B\times A$ が成り立つことを利用します。このことから、$A\leq B$ を満たす A を列挙すれば良いことがわかります。この場合、$1\leq A\leq\sqrt{N}$ を満たし、時間計算量は $O(\sqrt{N})$ となるため間に合います。なお、32bit 整数型によるオーバーフローに注意してください。

C++のコード例

```
using ll=long long;

```

```cpp
int cnt_digits(ll N){
    int digits=0;

    while(N>0){
        N/=10;
        digits++;
    }

    return digits;
}

int main(void){
    ll N;
    cin >> N;

    int ans=cnt_digits(N); // 1*N=N max(cnt_digits(1),cnt_digits(N))=cnt_digits(N)

    for(ll A=1LL;A*A<=N;++A){
        if(N%A!=0) continue;
        const ll B=N/A;

        const int cur=max(cnt_digits(A),cnt_digits(B));

        if(ans>cur){
            ans=cur;
        }
    }

    cout << ans << endl;
    return 0;
}
```

D: Maximum Average Sets

まず、与えられる品物の価値を降順にソートします。これを $v_0 \geq v_1 \geq \cdots \geq v_{N-1}$ とします。ここで、品物を選ぶ個数を K に固定して考えます。このとき、選んだ品物の価値の平均の最大値にする集合は、$\{v_0, v_1, \ldots, v_{K-1}\}$ です。また、K 個選んだときの品物の価値の平均の最大値と $K+1$ 個個選んだときの品物の価値の平均の最大値の差を比較します。

$$\frac{1}{K}\sum_{i=0}^{K-1} v_i - \frac{1}{K+1}\sum_{i=0}^{K} v_i = (K+1)\sum_{i=0}^{K-1} v_i - K\sum_{i=0}^{K} v_i = \sum_{i=0}^{K-1}(v_i - v_K) \geq 0$$

このことより、K 個選んだときの品物の価値の平均の最大値は $K+1$ 個個選んだときの品物の価値の平均の最大値以上となることが分かります。したがって、A 個選んだとき品物の価値の平均の最大値が 1 行目の解答とな

ります。

次に、選んだ品物の平均が最大となるような品物の選び方の数を数え上げます。このとき、上位 A 個の品物の価値 $\{v_0, v_1, \ldots, v_{a-1}\}$ により数え上げの方法を場合分けします。

- $v_0 \neq v_{A-1}$ の場合
  この場合は品物の個数を増やしていくと、選んだ品物の平均が減少していきます。したがって、$\{v_0, v_1, \ldots, v_{A-1}\}$ となる品物の選び方を数え上げていきます。この選び方は、$v_0 = v_i (0 \leq i \leq N-1)$ となる個数を X、$v_0 = v_i (0 \leq i \leq A-1)$ となる個数を Y とすると、答えは ${}_X\mathrm{C}_Y$ 通りです。

- $v_0 = v_{A-1}$ の場合
  この場合は品物の個数を増やした場合でも、選んだ品物の平均が最大となる場合があります。それは、選んだ全ての品物の価値が v_0 となる場合であり、A 個から B 個までの範囲内で品物を選んだ集合を数え上げていきます。したがって、$v_0 = v_i (0 \leq i \leq N-1)$ となる個数を X とおくと、答えは $\sum_{i=A}^{B} {}_X\mathrm{C}_i$ 通りとなります。

組み合わせ ${}_n\mathrm{C}_m (0 \leq n, m \leq N)$ はパスカルの三角形を利用することで、時間計算量 $O(N^2)$ で求めることが可能です。

以上のことから、時間計算量はソートで $O(N \log N)$、答えの平均の計算で $O(N)$、パスカルの三角形による組み合わせの前計算で $O(N^2)$、答えの数え上げに $O(N)$ となるため、合計で $O(N^2)$ となるため間に合います。

なお、階乗と階乗の逆元の前計算による時間計算量 $O(N \log N)$ となる解法が存在します。(詳細は略)

C++のコード例 (パスカルの三角形を利用した組み合わせの前計算)

```cpp
using ll=long long;
using R=double;

// Combination Table
ll C[51][51]; // C[n][k] -> nCk

void comb_table(int N){
    for(int i=0;i<=N;++i){
        for(int j=0;j<=i;++j){
            if(j==0 or j==i){
                C[i][j]=1LL;
            }else{
                C[i][j]=(C[i-1][j-1]+C[i-1][j]);
            }
        }
    }
}

int main(void){
    int N,A,B;
    cin >>  N >> A >> B;

```

```
    const int NMAX=50;
    ll v[NMAX];

    for(int i=0;i<N;++i){
        cin >> v[i];
    }

    comb_table(N);
    sort(v,v+N);
    reverse(v,v+N);

    R max_average=0.0;
    for(int i=0;i<A;++i){
        max_average+=v[i];
    }
    max_average/=A;

    int a_th_val_num=0,a_th_val_pos=0;
    for(int i=0;i<N;++i){
        if(v[i]==v[A-1]){
            a_th_val_num++;
            if(i<A){
                a_th_val_pos++;
            }
        }
    }

    ll cnt=0LL;
    if(a_th_val_pos==A){
        for(a_th_val_pos=A;a_th_val_pos<=B;++a_th_val_pos){
            cnt+=C[a_th_val_num][a_th_val_pos];
        }
    }else{
        cnt+=C[a_th_val_num][a_th_val_pos];
    }

    cout.precision(20);
    cout << fixed << max_average << endl;
    cout << cnt << endl;

    return 0;
}
```