4 たくさんの数字 (Many Digits)

Author : 蜂矢 倫久 (Mitsubachi)

小課題 1

葵さんが書く整数は 1 個で，その値は $A_1 + B_1$ である．したがって，この数字の桁数を求めればよい．

これは C++ ならば to_string 関数で整数を文字列に変換したのち，その文字列の長さを取得すればよい．$O(1)$ で答えが求まる．

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

int main(){
    int n;
    cin >> n;
    vector<long long> a(n),b(n);
    for(int i = 0; i < n; i++){
        cin >> a[i];
    }
    for(int i = 0; i < n; i++){
        cin >> b[i];
    }

    cout << to_string(a[0] + b[0]).size() << endl;
}
```

小課題 2

葵さんが書く N^2 個の整数は for 文を用いることですべて列挙できる．それらすべてに対して小課題 1 の手法で桁数を求めればよい．$O(N^2)$ で答えが求まる．

```
int ans = 0;
for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
        ans += to_string(a[i] + b[j]).size();
    }
}
cout << ans << endl;
```

小課題 3

$A_i = x$ となる i の個数が a 個で，$B_j = y$ となる j の個数が b 個とする．このとき，$A_i = x$，$B_j = y$ となる (i, j) の個数は ab 個である．

$1 \leqq x \leqq 2\,000$，$1 \leqq y \leqq 2\,000$ について $x + y$ の桁数に ab をかけた値を答えに足すという操作を行えばよい．あらかじめ x，y について a，b を求めておくと高速に答えが求まる．

$V = 2\,000$ として $O(V^2)$ で答えが求まる．

```
vector<long long> count_a(2010, 0), count_b(2010, 0);
for(int i = 0; i < n; i++){
    count_a[a[i]]++;
    count_b[b[i]]++;
}

long long ans = 0;
for(int i = 0; i < 2010; i++){
    for(int j = 0; j < 2010; j++){
        ans += count_a[i] * count_b[j] * to_string(i + j).size();
    }
}
cout << ans << endl;
```

小課題 4

葵さんが書く整数は 9 桁か 10 桁であり，特に 10 桁となるのは $A_i = 500\,000\,000$，$B_j = 500\,000\,000$ のときのみである．

よって，$A_i = 500\,000\,000$ となる i の個数が a 個で，$B_j = 500\,000\,000$ となる j の個数が b 個とすると答えは $9N^2 + ab$ である．$O(N)$ で答えが求まる．

```
    long long count_a = 0, count_b = 0;
    for(int i = 0; i < n; i++){
        if(a[i] == 500000000){
            count_a++;
        }
        if(b[i] == 500000000){
            count_b++;
        }
    }

    cout << (long long)(n)* n * 9 + count_a * count_b << endl;
```

小課題 5

葵さんが書く整数は 9 桁か 10 桁である．i ごとに $A_i + B_j$ が 10 桁になる j の個数を求める．

このような j は $10^9 - A_i \leqq B_j$ を満たす．あらかじめ B_j を保存する配列を sort しておくと lower_bound により $O(\log N)$ で j の個数を求めることができる．

$O(N \log N)$ で答えが求まる．

```
    sort(b.begin(),b.end());

    long long ans = (long long)(n) * n * 9;
    for(int i = 0; i < n; i++){
        ans += b.end() - lower_bound(b.begin(), b.end(), 1000000000 - a[i]);
    }
    cout << ans << endl;
```

小課題 6

$A_i + 1$ から $A_i + N$ までの桁数の合計を高速に求めることを考える．

これはあらかじめ 1 から x までの桁数の合計を $1 \leqq x \leqq 300\,000$ について求める．これは累積和を用いれば高速に求まる．$A_i + 1$ から $A_i + N$ までの桁数の合計は 1 から $A_i + N$ までの桁数の合計から 1 から A_i までの桁数の合計を引いた値であるので $O(1)$ で求まる．

$V = 150\,000$ として $O(V + N)$ で答えが求まる．

```
    vector<long long> sum(150100 + n, 0);
    for(int i = 1; i < sum.size(); i++){
        sum[i] = sum[i - 1] + to_string(i).size();
    }

    long long ans = 0;
    for(int i = 0; i < n; i++){
        ans += sum[a[i] + n] - sum[a[i]];
    }

    cout << ans << endl;
```

小課題 7

小課題 6 と同様に $A_i + 1$ から $A_i + N$ までの桁数の合計を高速に求めることを考える．

$A_i + 1$ から $A_i + N$ までのうち d 桁であるような数の個数を $1 \leqq d \leqq 10$ について求めることを考える．このような個数は l を 10^{d-1} と $A_i + 1$ のうち小さくない方，r を $10^d - 1$ と $A_i + N$ のうち大きくない方として，$r - l + 1$ と 0 のうち大きくない方である．

よって，$O(N)$ で答えが求まる．

```
    long long ans = 0;
    for(int i = 0; i < n; i++){
        long long left = 1, right = 10;
        for(int d = 1; d <= 11; d++){
            long long l = max(left, a[i] + 1), r = min(right - 1, a[i] + n);
            ans += d * max(0LL, r - l + 1);
```

```
            left *= 10;
            right *= 10;
        }
    }

    cout << ans << endl;
```

満点

$1 \leqq d \leqq 10$ について i を固定したときに $A_i + B_j$ が d 桁となるような j の個数を求めることを考える．

このような j は $10^{d-1} - A_i \leqq B_j \leqq 10^d - 1 - A_i$ を満たす．したがって，j の個数は小課題 5 のように，あらかじめ B_j を保存する配列を sort しておくと lower_bound により $O(\log N)$ で j の個数を求めることができる．

よって，$O(N \log N)$ で答えが求まる．

```
    sort(b.begin(),b.end());

    long long ans = 0;
    long long x = 1;
    for(int d = 1; d <= 10; d++){
        // d 桁の数字の個数を数える
        for(int i = 0; i < n; i++){
            // a[i] + b[j] が d 桁となるような j の数を数える
            // 10^{d - 1} <= a[i] + b[j] < 10^d
            // 10^{d - 1} - a[i] <= b[j] < 10^d - a[i]
            long long left = x - a[i], right = 10 * x - a[i];
            auto itr1 = lower_bound(b.begin(), b.end(), right);
            auto itr2 = lower_bound(b.begin(), b.end(), left);
            ans += d * (itr1 - itr2);
        }
        x *= 10;
    }

    cout << ans << endl;
```