ABC 147 解説

kort0n, kyopro_friends, satashun, sheyasutaka

2019 年 12 月 8 日

For International Readers: English editorial starts on page 7.

A: Blackjack

3 つの整数を受け取り、その合計が 22 以上かそうでないかによって条件分岐します。以下は C 言語での実装例です。

```
#include<stdio.h>
int main(){
    int a,b,c;
    scanf("%d%d%d",&a,&b,&c);
    if(a+b+c>=22)puts("bust");
    else puts("win");
}
```

B: Palindrome-philia

S の長さを N とします。S の長さは変えられないため、S の 0 文字目 (最初の文字) と $N-1$ 文字目 (最後の文字) が一致していなければそのうちの一方をもう一方に合わせ、1 文字目と $N-2$ 文字目が一致していなければそのうちの一方をもう一方に合わせ、...、 文字列の中心までこれを繰り返すのが最適です。for, while などの構文を使って実装するのが一般的でしょう。以下、C 言語での実装例を挙げます。

```c
#include <stdio.h>
#include <string.h>
char s[101];
int main(void) {
  scanf("%s", s);
  int res = 0;
  int n = strlen(s);
  for (int i = 0; i < n; i++) {
    if (s[i] != s[n - 1 - i]) res++;
  }
  printf("%d\n", res / 2);
}
```

C: HonestOrUnkind2

N 人の人がそれぞれ正直者であるか不親切な人であるかを決めれば, その状態が証言と矛盾しないかは, $O\left(N^2\right)$ で容易に確かめることが出来ます.
N 人の人がそれぞれ正直者か否かについては全部で 2^N 通りしかありませんから, これら全ての場合について矛盾が生じないかを調べ, 矛盾が生じないうちでの正直者の最多人数を答えとすれば良いです.
実装上は, 0 以上 2^N 未満の整数 j が「1 以上 N 以下の整数 k について, 人 k が正直者であることと, j を 2 進数表記した際に $k-1$ 桁目が 1 であることが同値」という状態を表すことと定義すると, 容易に全ての場合を試すことが出来ます.
この手法は bit 全探索と呼ばれています.
C++ による解答例:https://atcoder.jp/contests/abc147/submissions/8830089

D: Xor Sum 4

XOR をとる操作で各 bit は互いに干渉しないため、bit ごとに独立に考えることができます。したがって A_i が 0 または 1 である場合が解ければよいです。
A_i XOR A_j は、$A_i = A_j$ のとき 0、$A_i \neq A_j$ のとき 1 なので

$$\begin{aligned}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(A_i \text{ XOR } A_j) &= \#\{(i,j) \mid i<j \text{ かつ } A_i \neq A_j\} \\ &= (0 \text{ の個数}) \times (1 \text{ の個数})\end{aligned}$$

として求めることができます。各 bit ごとに 0 のものと 1 のものの個数がわかっていればよいので元の問題は $O(N \log \max A_i)$ で解けました。

E: Balanced path

色の塗り方と移動経路を決める順番は答えに影響しません。そこで、移動しながら塗り方を決めることで、bool 値のテーブルを持つことを考えます。

$$DP[i][j][k] = \text{マス } (i,j) \text{ までの経路の偏りが } k \text{ になることがあるか？}$$

マス (i,j) で偏りが k であるとき、マス $(i+1,j)$ での偏りは、マス $(i+1,j)$ の数の色の塗り方により $k+|A_{i+1,j}-B_{i+1,j}|$ か $|k-|A_{i+1,j}-B_{i+1,j}||$ となります。マス $(i,j+1)$ についても同様です。したがって、i,j が小さい方から順にテーブルを埋めていくことができます。

$M=\max\{|A_{11}-B_{11}|,...,|A_{HW}-B_{HW}|\}$ として、偏りは最大で $O((H+W)M)$ 程度まで考慮する必要があるので、全体で $O(HW(H+W)M)$ で解くことができました。なお、計算の順序を工夫することで空間計算量 $O(\min\{H,W\}(H+W)M)$ で解くことや、bitset を用いた高速化をすることもできます。

F: Sum Difference

S と T の和は一定であるので、$S-T=S-(U-S)=S*2-U$ (U は全ての和) より、 差の代わりに S の和として考えられるものの個数を求めれば良いです。

まず、$D=0$ の場合、X が N 個あります。$X=0$ のとき、和は 0 のみです。$X\neq 0$ のとき、和は $0, X, \cdots N*X$ の $N+1$ 通りです。

以下では $D\neq 0$ の場合を考えます。$D<0$ の場合は、S と T の寄与を入れ替えることを考えると、$X=-X, D=-D$ として解いても同じことがわかるので、$D>0$ とします。

S 側が $k(0\leq k\leq N)$ 個の元から成るとします。数列 $X+i*D(0\leq i<N)$ の和は $k*X+I*D$ (I は i の和) の形をしており、$0+1+\cdots+(k-1)\leq I\leq (N-k)+(N-k+1)\cdots+(N-1)$ を満たしますが、この範囲の全ての整数を取ることができます。

つまり、各 k について $\bmod D$ で同じ整数から成る連続した区間を成しており、これらの和集合が求める答えとなります。これは $k*X \bmod D$ ごとに区間をソートしておくと求めることができます。

以上の時間計算量は $O(NlogN)$ であり、十分高速です。(答えのオーダーは $O(N^3)$ です)

A: Blackjack

Receive 3 integers and switch cases depending on whether their sum is equal to or more than 22 or not. The following is an implementation example in C Language.

```
#include<stdio.h>
int main(){
    int a,b,c;
    scanf("%d%d%d",&a,&b,&c);
    if(a+b+c>=22)puts("bust");
    else puts("win");
}
```

B: Palindrome-philia

Let N be the length of S. Since you cannot change the length of S, it is optimal to set the 0-th (first) letter to be the $N-1$-th (last) letter if it's not same, and to set the 1-st letter to be the $N-2$-th letter if it's not same, ..., until they meets at the middle. The most general way may be using for or while statements. The following is an implementation example in C Language.

```c
#include <stdio.h>
#include <string.h>
char s[101];
int main(void) {
  scanf("%s", s);
  int res = 0;
  int n = strlen(s);
  for (int i = 0; i < n; i++) {
    if (s[i] != s[n - 1 - i]) res++;
  }
  printf("%d\n", res / 2);
}
```

C: HonestOrUnkind2

If you fix whether each of N people is honest or unkind, then you can check easily if it contradicts to any testimonies or not in a total of $O\left(N^2\right)$ time.
There are only 2^N combinations of whether each of N people is honest or unkind in total, so you can check for all the pattern if there are no contradictions, and let the answer be the maximum number of honest person.
When implementation, if you regard an integer j in a range of 0 (inclusive) to 2^N (exclusive) as representing "for each integer k in a range of 1 to N (inclusive), person k is honest if and only if $k-1$-th digit of j when binary-notated is 1," then you can perform brute forcing easily.
This technique is called bitmask brute forcing.
Sample Code in C++:https://atcoder.jp/contests/abc147/submissions/8830089

D: Xor Sum 4

In an operation of taking XOR, each bit is independent from other bits, so you can think each bit independently. Therefore, it is sufficient to solve for A_i is 0 or 1.
A_i XOR A_j is 0 if $A_i = A_j$, and 1 if $A_i \neq A_j$, so it holds that

$$\begin{aligned}\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (A_i \text{ XOR } A_j) &= \#\{(i,j) \mid i < j \text{ and } A_i \neq A_j\} \\ &= (\text{number of 0s}) \times (\text{number of 1s}).\end{aligned}$$

Now you have to find, for each bit, the number of elements such that the bit is 0 and 1, so the original problem can be solved in a total of $O(N \log \max A_i)$ time.

E: Balanced path

The answer is independent from the order of deciding colors to paint and the path. So let's consider holding a bool table by deciding colors while moving.

$$DP[i][j][k] = \text{Can the unbalancedness of a path to square } (i, j) \text{ be } k?$$

If the unbalancedness at square (i, j) is k, then unbalancedness at square $(i + 1, j)$ is either $k + |A_{i+1,j} - B_{i+1,j}|$ or $|k - |A_{i+1,j} - B_{i+1,j}||$, depending on how to paint the numbers of square (i, j). The same is true for square $(i, j + 1)$. Therefore, you can fill the table in the increasing order of i, j.

Let $M = \max\{|A_{11} - B_{11}|, ..., |A_{HW} - B_{HW}|\}$, then you have to consider a maximum unbalancedness of $O((H + W)M)$, so it can be solved in a total of $O(HW(H + W)M)$ time. If you choose good order of calculation, you can solve in a total of $O(\min\{H, W\}(H + W)M)$ space. You can also accelerate the solution using bitset.

F: Sum Difference

Since the sum of S and T is constant and $S - T = S - (U - S) = S * 2 - U$ (where U is the sum of all the elements), it is sufficient to count the number of possible sums of S instead of differences.

First, if $D = 0$, there are N X's. If $X = 0$, then the only sum is 0. If $X \neq 0$, then there are $N + 1$ possible sums, $0, X, \cdots N * X$.

Hereinafter we assume that $D \neq 0$. If $D < 0$, then the contribution of S and T can be swapped, so we can solve this problem under conditions of $X = -X, D = -D$, so we assume that $D > 0$.

Assume that S consists of $k (0 \leq k \leq N)$ elements. The sum of a sequence $X + i * D (0 \leq i < N)$ can be represented as $k * X + I * D$ (where I is sum of i's), and it holds that $0 + 1 + \cdots + (k - 1) \leq I \leq (N - k) + (N - k + 1) \cdots + (N - 1)$. Actually I can be any integers in this range.

Therefore, for each k it forms a continuous sequence, each element of which has same remainder mod D, and the answer is the union of them. You can find this by sorting the segments for each $k * X \bmod D$.

The total time complexity is $O(NlogN)$, and it's fast enough. (The answer will have an $O(N^3)$ size.)