ABC 147 解説

kort0n, kyopro_friends, satashun, sheyasutaka

2019 年 12 月 8 日

For International Readers: English editorial will be published in a few days.

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)