エイシング プログラミング コンテスト 2020

writer: camypaper

2020 年 7 月 11 日

For International Readers: English editorial starts on page 8.

A: Number of Multiple

L 以上 R 以下の整数全てについて d の倍数かどうかの判定を行うのが最も素朴な解法です。これは $O(R-L)$ で実行可能で、今回の制約においては十分高速に動作します。

その他の解法として $f(N)$ を 1 以上 N 以下のうち d であるようなものの個数とすると、答えが $f(R)-f(L-1)$ と表せることを利用する方法があります。$f(N)=\left\lfloor \frac{N}{d} \right\rfloor$ であることから単純な四則演算のみを用いて求めることができ $O(1)$ で実行が可能です。

解答例

```
l, r, d = map(int, input().split())
print(r//d - (l-1)//d)
```

B: An Odd Problem

結局、奇数番目のマスのうち奇数であるようなものの個数を求められればよいです。for ループなどの繰り返しを用いて素直に実装すればよいです。

解答例

```
n = int(input())
a = list(map(int, input().split()))

ans = 0
for i in range(0,n,2):
    if a[i] % 2 == 1:
        ans += 1
print(ans)
```

C: XYZ Triplets

まず n の値が与えられたときに条件を満たす (x, y, z) の個数を数学的に判定するのではなく、全探索してしまうことを考えてみましょう。
(x, y, z) の値の取りうる範囲を考えてみると $1 \leq x, y, z \leq \lceil \sqrt{n} \rceil$ 程度の範囲で全探索を行えばよいです。

$f(n)$ を求めるのが $O(n^{\frac{3}{2}})$ でできることがわかりました。ここで、$f(n)$ を計算する過程で $f(n-1)$ の候補が全て現れることが重要です。
$f(1), f(2), \ldots, f(N)$ についてそれぞれ計算するのではなく、$f(1), f(2), \ldots, f(N-1), f(N)$ をまとめて計算することで全体として $O(N^{\frac{3}{2}})$ となります。これは十分高速に動作します。

解答例

```
n = int(input())
ans = [0 for _ in range(10050)]
for i in range(1,105):
    for j in range(1,105):
        for k in range(1,105):
            v = i*i+j*j+k*k+i*j+j*k+k*i;
            if v<10050:
                ans[v]+=1

for i in range(n):
    print(ans[i+1])
```

D: Anything Goes to Zero

まずはじめに操作回数の上界を見積もってみましょう。一回の操作につき N が $\log N$ へと変化すると大雑把に見積もると、操作回数は $O(\mathrm{slog}N)$ となることがわかります。ここから、この問題の制約下で操作の回数は最悪でも $5, 6$ 回程度となることがわかります。

$X_1, X_2, \ldots, X_N$ に対して愚直に操作をシミュレーションすることを考えます。ここで、問題になるのは初回の操作に限り、$O(N)$ の計算コストがかかることです。二回目以降は桁数が初回に比べて十分小さいため、愚直に実施しても問題ありません。よって、初回の操作を高速に実施する方法を考えればよいです。

$\mathrm{popcount}(X_i)$ の値は $\mathrm{popcount}(X) \pm 1$ のいずれかです。
よって、$X \pmod{\mathrm{popcount}(X)+1}, X \pmod{\mathrm{popcount}(X)-1}$ の値を予め計算しておき、$2^i \pmod{\mathrm{popcount}(X)+1}, 2^i \pmod{\mathrm{popcount}(X)-1}$ を足し引きすることで、初回の操作の結果を高速に計算することができます。
ただし、操作回数が 0 になる場合が存在することに注意してください。

32bit 整数に対する $\mathrm{popcount}(N)$ が $O(1)$ で動作することを仮定すると、全体として $O(N\mathrm{slog}N)$ となって十分高速です。

E: Camel Train

まず $L_i \geq R_i$ なるラクダを赤く、$L_i < R_i$ なるラクダを青く塗ることにします。
最適解において、青いラクダの右隣に赤いラクダがいることがないような配置が必ず存在します。これはそのような 2 頭のラクダがいたならば 2 頭の位置を入れ替えてもうれしさの総和が減少しないことから示せます。
ここから、赤、青それぞれについて独立な問題として考えればよいことがわかります。
以降は全てのラクダが $L_i \geq R_i$ であることを仮定します。

少なくとも $\sum_{i=1}^{N} \min(L_i, R_i)$ のうれしさが得られることは明らかです。よって、L_i, R_i から $\min(L_i, R_i)$ は予め答えに足しておいて、どれだけのうれしさを追加で得られるかを考えましょう。
すると、ラクダ i が先頭から K_i 番目以内にいたなら $L_i - R_i$ だけうれしさが得られる。そうでなければ何も起こらない、という制約下でうれしさの総和を最大化すればよいことがわかります。

例えば、以下のような貪欲法により、最適解を得ることができます(他にも様々な方法があります)。

- ラクダたちの集合 S を用意する。S ははじめ空集合。
- $j = 1, 2, 3, \ldots, N$ の順に着目する
    - $K_i = j$ なる全てのラクダを S に追加する。
    - その後、$|S| > j$ である限り、ラクダを取り除く
        - このとき、$L_i - R_i$ が小さいラクダから順に取り除く

上記の貪欲法は優先度付きキューを用いて $O(N \log N)$ で実行可能で十分高速です。

F: Two Snuke

説明しやすさのためテスター解を紹介します。

$\Delta s = s_2 - s_1, \Delta n = n_2 - n_1, \Delta u = u_2 - u_1, \Delta k = k_2 - k_1, \Delta e = e_2 - e_1$ とします。
また、$s = 2s_1, n = 2n_1, u = 2u_1, k = 2k_1, e = 2e_1$ とします。

これらを用いると、問題文中の条件は以下のように言い換えられます。

- $0 \leq s_1, n_1, u_1, k_1, e_1$
- $1 \leq \Delta s, \Delta n, \Delta u, \Delta k, \Delta e$
- $0 \leq (s + n + u + k + e) + (\Delta s + \Delta n + \Delta u + \Delta k + \Delta e) \leq N$

N 個の一列に並んだボールに対し、10 個の仕切りを入れて、$s, n, u, k, e, \Delta s, \Delta n, \Delta u, \Delta k, \Delta e$ とあまりに分割することを考えます。ただし、s, n, u, k, e が偶数になるよう分割する必要があります。分割を行った後、$\Delta s, \Delta n, \Delta u, \Delta k, \Delta e$ からそれぞれ 1 つ選ぶ方法の個数が答えと $1:1$ 対応しています。
これは $N-5$ 個の一列に並んだボールに対し、15 個の仕切りを(最初の 5 つのセグメントに含まれるボールの個数が偶数になるように)入れる方法と $1:1$ 対応しています。
$\mathrm{dp}(i, j, p) = i$ 個のボールまで見て、現在仕切りを j 個入れ、現状のセグメントに含まれるボールの個数のパリティが p であるような仕切りの入れ方の個数、としたDPでこの問題を解くことができます。
これは $O(N)$ のため間に合いませんが、このDPを行列累乗で高速化することで、$O(\log N)$ となって十分高速に実行できます。

なお、この問題は $O(1)$ で解くことができますが、詳細は省略します。

AIsing Programming Contest 2020

writer: camypaper

Saturday, July 11, 2020

Under Construction.