Beginners After Typhoon Contest 01 解説

leafirby(@leafirby)

2019 年 10 月 14 日

目次

1 A: Wedding Ceremony

Writer: klno / Tester: QCFium, leafirby, null0124

N が奇数のときは N、N が偶数のときは $N+1$ が答えです。

C++ での実装例

2 B: Rating じゃんけん

Writer: null0124 / Tester: QCFium, klno, leafirby

問題文を簡潔に纏めると、

$(i) A = C$ のとき、$B = D$ なら”Draw”,$(B+1) mod 3 = D$ なら”null”, それ以外なら”tRue”を出力する。

(ii) それ以外のとき、$A > C$ なら”null”, それ以外なら”tRue”を出力する。

となります。あとは、これを実装するだけです。

C++ での実装例

3 C: う し た ぷ に き あ く ん 笑

Writer: null0124 / Tester: QCFium, leafirby

問題の通りに実装して下さい。

C++ での実装例

4 D: Grid Numbers

Writer: leafirby / Tester: QCFium, klno

長さ $H \times W$ の一次元配列に入力を受け取り、昇順ソートしてから 1 行ずつ出力すれば良いです。

C++ での実装例

5 E: Devation Score

Writer: klno / Tester: QCFium, leafirby

問題の通りに実装すればいいのですが、int 型で計算すると許容外の誤差が発生してしまいます。

従って double 型で計算したあと、int 型で出力するとよいです。

C++ での実装例

6 F: 凸凹

Writer: null0124 / Tester: QCFium, leafirby

$q-p$ が十分小さいので、全探索で解くことができます。
このとき、(y,x) を格納する長さ $q-p+1$ の pair 配列 P を用意し、順次 P に入れたあと
P をソートすると、答えとなる座標は $P[q-p]$ と $P[0]$ になります。
ただし、y 座標 が最大の座標が複数ある場合は、より p に近い方の座標を出力しなければ
ならないことに注意して下さい。
C++ での実装例

7 G: キムワイプイーター

Writer: null0124 / Tester: QCFium, leafirby

問題の通りに実装すれば良いです。
具体的には、$(N+1)\times(N+1)$ の 1 で初期化した bool 配列を用意します。
この配列を dist とすると、最初 $dist_{N,0}=0$ です。
あとは、S の通りに動いて、訪れたところを 0 にして移動が終わったら dist 配列を出力すれば良いです。
(ただし、この配列において座標は dist[上下反転した Y 軸][X 軸] に格納されていることに注意して下さい。)
C++ での実装例

8 H: Part 1

Writer: leafirby / Tester: QCFium, null0124

長くなるので解説記事を別途用意しました。
解説記事
C++ での実装例

9 I: おもち

Writer: leafirby / Tester: QCFium, klno

愚直にべき乗してしまうと O(N) かかってしまい、間に合いません。
ここで登場するのが、”繰り返し二乗法 (二分累乗法)”というアルゴリズムです (参考)。
このアルゴリズムを利用すると、O(log N) で計算することができ、この問題を解くのに十分高速です。
C++ での実装例

10 J: Devation Score 2

Writer: klno / Tester: QCFium, leafirby

この問題が実装するだけなのは言うまでもありません。(問題文通りの実装は $O(NlogN)$ です。)
本質は高速化です。無駄な宣言を減らす、入出力を高速化するなどの定数倍高速化を頑張ると通ります。
C++ での実装例

11 K: Circles

Writer: klno / Tester: QCFium, leafirby, null0124

この問題において、全ての円の中心は x 軸上に存在することが保障されているので、
当然、一番多くの円の中にある座標は x 軸上に存在します。
従って、長さ $4 \times 10^5 + 1$ の配列 P を用意して $P[i]$ に $(i, 0)$ はどれだけの円の中に入っているかを
記録していきます。ただし、そのまま記録すると i が負の数になってしまうので、先に全ての円の中心座標を
$+2 \times 10^5$ しておきます。そして、各円について中心から左側の座標 $(i, 0)$ で x 軸と交われば $P[i]+ = 1$、
右側で交われば $P[i]- = 1$ とします。このとき、$P[i]$ の累積和を取り、昇順ソートすれば、
$P[4 \times 10^5]$ が答えになります。よって、この問題は $O(NlogN)$ で解けました。
C++ での実装例

12 L: Zero-Xor-Sum Ranges

Writer: null0124 / Tester: QCFium, klno, leafirby

x と y の排他的論理和を $x \oplus y$ と書くことにします。

重要な性質

任意の非負整数 N について、$N \oplus N = 0$ である。

P を長さ $n+1$ の数列とし、$P_0 = 0, P_i = P_{i-1} + A_i$ とします。
ある連続する部分列の総 xor 和は、P の 2 要素の xor 和です。
また、重要な性質から xor 和が 0 になる連続する部分列の数は、P の中で値の等しい 2 つの要素を
選ぶ方法の数に等しいです。これは、P の要素をソートするなどして、同じ値の個数を数えることで
計算出来ます。ソートに $O(NlogN)$ かかるため、この問題は $O(NlogN)$ で解けました。
C++ での実装例

13 M: Four Points Tour

Writer: leafirby / Tester: QCFium, null0124

null くんが K 秒目に地点 A にいる場合,$K+1$ 秒目に地点 A にいることはできません。
また K 秒目に地点 B, C, D にいる場合、$K+1$ 秒目に地点 A にいることができます。
つまり、K 秒目に地点 A にいる通り数は $K-1$ 秒目に地点 A にいない通り数に等しいです。
また、1 秒ごとに、樹形図が 3 本に分かれるので、この問題を解く漸化式は、

$$DP[i] = 3^{i-1} \times DP[i-1]$$

になります。(ただし、$DP[0] = 1$) 従って、この問題は $O(N)$ で解くことができました。
また、別解として $O(logN)$ 解法も存在します。興味のある方は考えてみて下さい。
C++ での実装例
C++ での実装例 (別解)

14 N: Same heights

Writer: klno / Tester: QCFium, leafirby

結論から述べると、$A = x_i$の中央値 のとき、コストが最小になります。詳しい説明は以下の通りです。
x が昇順になってるとするとき、x_i と x_{i+1} の間で A を 1 右に動かすと、
A より左の柱のコストの合計は i 増えて A より右の柱のコストの合計は $(N-i)$ 減るので,
合計のコストの変化は $(N-2i)$ である。
従って $N-2i > 0$ である間 A を右に動かすと、$N-2i$ の符号は中央値のところで反転するので
$A = x_i$の中央値 のとき、コストが最小になる。
C++ での実装例

15　O: RPN

Writer: klno / Tester: QCFium, leafirby, null0124

問題の通りに実装すれば良いです。

具体的には、A_i を順次読み取っていき、数字は stack に入れ、符号が来た場合は、stack から 2 要素取り出して計算し、再び stack に入れていきます。

このとき、上記の操作後に残った stack の要素が答えです。従って、この問題は $O(N)$ で解けました。

C++ での実装例

16　P: null くんの鉄道旅行

Writer: null0124 / Tester: QCFium, leafirby

この問題は、最短ハミルトンパス (全ての頂点を一度ずつ通るパスの内、最短のもの) の長さを求める問題に帰着できます。これは dfs や、next permutaion を用いて全探索することで,$O(N!)$ で求めることができます。

N は高々 8 なので、問題を解くのに十分高速です。

また、ここでは詳しく説明しませんが bitDP を用いて $O(N^2\ 2^N)$ で解くこともできます。

C++ での実装例

17　Q: 黒塗りの LCM

Writer: leafirby / Tester: QCFium, klno

K を素因数分解したものを $l_1^{r_1} \times l_2^{r_2} .... \times l_k^{r_q}$ とします。そして、これを分配したものが $P_1, P_2, ...P_N$ です。
ここで、各 P_i の素因数分解を次のように表します。

$$P_i = l_1^{r_{1_i}} \times l_2^{r_{2_i}} .... \times l_q^{r_{q_i}} (l, r \text{ は非負整数})$$

このとき、

$$LCM(P_1, P_2, ...P_N) = l_1^{max(r_{1_1}, r_{1_2}, ...r_{1_N})} \times l_2^{max(r_{2_1}, r_{2_2}, ...r_{2_N})} ... \times l_q^{max(r_{q_1}, r_{q_2}, ...r_{q_N})}$$

なので、$LCM(P_1, P_2, ...P_N)$ を最小化する為には、同じ素因数を出来るだけ分散させればいいです。
これは、$D : GridNumbers$ と同じ操作です。例えば、$N = 5, K = 3840$ の場合、

$$3840 = 2^8 \times 3 \times 5$$

2 2 2 2 2

2 2 2 3 5

---

4 4 4 6 10

となり、最小の最小公倍数は 60 だとわかります。
以上の操作から、

$min(LCM(P_1, P_2....P_N)) = l_1^{\lceil \frac{r_1}{N} \rceil} \times l_2^{\lceil \frac{r_2}{N} \rceil} .... \times l_q^{\lceil \frac{r_q}{N} \rceil}$ (ただし、$\lceil X \rceil$ は実数 X に対して X 以上の最小の整数とする。)

と表せます。素因数分解に $O(\sqrt{K})$、累乗に繰り返し二乗法を使って $O(q\ log\ max(r1, r2, ...r_q))$ かかるので、
この問題は $O(\sqrt{K})$ で解けました。
C++ での実装例
別解
実際に、D 問題の操作を行った上で順次最小公倍数を取っていく方法でもできます。
この場合は、$O(NlogK)$ です。これは、問題を解くのに十分高速です。
C++ での実装例 (別解)

18 EX: No Bingo!

Writer: leafirby / Tester: QCFium

※この問題の解決に、色々な人に協力して頂きました。ありがとうございました。

※解説の原案は square1001 さんによるものです。

解説

ビンゴにならない条件は以下の通りです。

(i) どの行, 列にも開いてない所がある。

(ii) どちらの対角にも開いてない所がある。

よって、この問題は以下の問題に帰着できます。

順列

$(0, 1, 2...N-1)$ からなる順列 P は全部で $N!$ 通りある。
このうち、$P_i = i$ となる場所と $P_i = N - i + 1$ となる場所の両方が存在するものは何通りあるか。

このままでは考えにくいので、以下の様に分割します。

$A-$ 全事象

$B-$ $P_i = i$ となる場所が存在しない順列の個数

$C-$ $P_i = N - i + 1$ となる場所が存在しない順列の個数

$D-$ $B \cap C$ を満たす順列の個数

このとき、求める答えは $A - (B + C) + D$ です。

まず、$A = N!$ です。

次に、B は「完全順列 (撹乱順列)」なので $B = A\sum_{k=2}^{N} \frac{(-1)^k}{k!}$ です。

C は B と対象なので、$C = B$ です。

最後に、D を包除原理を用いて求めることを考えます。

いくつかの i について「すくなくとも $P_i = i$ または $P_i = N - i + 1$ であると決める」とします。

この"決めた i の個数"を k とすると、残りは $(N-k)!$ 通りありますが、包除原理なので D の値に $(N-k)! \times (-1)^k$ 影響します。

x に対して合計の"影響値"は $1, N$ 列目で $1, N$ 行目の 4 マス,$2, N-1$ 列目で $2, N-1$ 行目の 4 マス $3, N-2$ 列目で $3, N-2$ 行目の 4 マス・・・と

"4 マスごとに" 決めていけばよいです。当然 N が奇数の場合は、最後 1 マスだけになります。

多項式 $P(x) = a_0 \times x^0 + a_1 \times x^1 + ... + a_N \times x^N$ を考えます。

このとき、$D = (a_i \times (N-i)! \times (-1)^i)$ の合計、というように求められるようにします。多項式 $P(x)$ は 4 マス決まるときに $(2x^2 + 4x + 1)$ 掛けられ、1 マス決まるときは $(x + 1)$ 掛けられます。

要するに、$N = 2q$ と表される場合 $P(x) = (2x^2 + 4x + 1)^q$

$N = 2q + 1$ のとき $P(x) = (2x^2 + 4x + 1)^q \times (x + 1)$ です。

全部の計算は、普通にやると $O(N^2)$ ですが、FFT + 繰り返し二乗法で $O(NlogN)$ で求めることができます。

A,B,C の計算に $O(N)$ かかるので、$O(NlogN)$ でこの問題を解くことができました。

別解: D の多項式は OEIS に掲載されています。これを活用することで、$O(N)$ でこの問題を解くことができます。