YNU CPC 2021 VC 013 A + ex
解説

KY2001

2021 年 7 月 23 日

以下公式解説のリンクです。わからない部分があればこちらを参照してみて下さい。

1. Snack
   https://img.atcoder.jp/abc148/editorial.pdf

2. Next Prime
   https://img.atcoder.jp/abc149/editorial.pdf

3. Skip
   https://img.atcoder.jp/abc109/editorial.pdf

4. Digits in Multiplication
   https://img.atcoder.jp/abc057/editorial.pdf

5. Knight
   https://img.atcoder.jp/abc145/editorial.pdf

6. Tough Journey
   https://img.atcoder.jp/code-festival-2018-final/editorial.pdf

7. ネタだけ食べたい寿司
   https://img.atcoder.jp/dwacon2017-prelims/editorial.pdf

1 Snack

1.1 解法

答えは A と B の”最小な””公倍数”です。C++ では標準ライブラリの lcm 関数を使うことができます。

1.2 実装例

```
#include <bits/stdc++.h>
#define int long long
#define rep(i, n) for (int i = 0, i##_len = (n); i < i##_len; i++)
using namespace std;

signed main() {
  int A, B;
  cin >> A >> B;
  cout << lcm(A, B) << endl;
}
```

2 Next Prime

2.1 解法

講習会資料の通り, ある整数 n が素数か判定する時間計算量は $O(\sqrt{n})$ です。また, 計算すれば分かりますが, 素数同士の幅は最大でも 72 なので, X 以上の数を貪欲に素数かどうか判定していけば良いです。

2.2 実装例

```
#include <bits/stdc++.h>
#define int long long
#define rep(i, n) for (int i = 0, i##_len = (n); i < i##_len; i++)
using namespace std;

bool is_prime(int n) {
  if ((n % 2 == 0 and n != 2) or n == 1) return false;
  for (int k = 3; k < (int)sqrt((double(n))) + 2; k += 2)
    if (n % k == 0) return false;
  return true;
}

signed main() {
  int X;
  cin >> X;
  for (int i = X; i < 1000000; i++) {
    if (is_prime(i)) {
      cout << i << endl;
      break;
    }
  }
}
```

3 Skip

3.1 解法

題意より, D は移動に関する最小単位となることが分かります。すなわち全ての都市を訪れるためには, 全ての x について, X からの相対距離 $x - X$ が D の倍数になる必要があります。つまり, 答えは全ての x について, X からの相対距離 $|x - X|$ を割り切るような最大の自然数です。つまり, 最大公約数 $gcd(|x_1 - X|, |x_2 - X|, ..., |x_N - X|)$ を求めれば良いことになります。また, 3 つ以上の数の最大公約数の取り方は単純に前から順番に $gcd(a_2, gcd(a_1, a_0))$ のような形でとればよいことが証明できます。なお, C++14 以降では標準ライブラリの gcd 関数を使うことができます。

3.2 実装例

```
#include <bits/stdc++.h>
#define int long long
#define rep(i, n) for (int i = 0, i##_len = (n); i < i##_len; i++)
using namespace std;

signed main() {
  int N, X, x, ans;
  cin >> N >> X;
  cin >> x;
  ans = abs(X − x);
  rep(i, N − 1) {
    cin >> x;
    ans = gcd(ans, abs(X − x));
  }
  cout << ans << endl;
}
```

4 Digits in Multiplication

4.1 解法

$N = A \times B$ を満たすような全ての $A \leq B$ について $F(A, B)$ の最小値を求めれば良いので, $A \leq B$ としても良いことに注意すると, A を 2 から $\sqrt{N}$ 以下の最大の自然数まで全探索すれば良いです (約数列挙の考え方)。また, A, B の桁数を求めるという操作は to_string 関数を用いて文字列型に直すと楽です。

4.2 実装例

```
#include <bits/stdc++.h>
#define int long long
#define rep(i, n) for (int i = 0, i##_len = (n); i < i##_len; i++)
using namespace std;

signed main() {
  int N, ans = 1e9;
  cin >> N;
  for (int i = 1; i <= (int)sqrt(N); i++) {
    if (N % i == 0) {
      ans = min(ans, (int)max(to_string(i).size(), to_string(N / i).size()));
    }
  }
  cout << ans << endl;
}
```

5 Knight

5.1 解法

駒を $(i+1, j+2), (i+2, j+2)$ に移動した操作の回数をそれぞれ a, b 回とすれば, その時の駒の座標は $(a+2b, 2a+b)$ となります。これが (X, Y) と一致すれば良いので $(a+2b, 2a+b) = (X, Y)$ という連立方程式を解くと $(a, b) = (\frac{2X-Y}{3}, \frac{-X+2Y}{3})$ となり, 移動回数が一意に定まります。ただし, a, b が整数の値をとらない, もしくは a, b が負の値をとる場合は答えは 0 通りとなることに注意して下さい。あとは 2 つの操作の組合せの数え上げを行なう必要がありますが, これは総移動回数 $a+b$ 回のうち a 回を選んで, そのときのみ $(i+1, j+2)$ に移動すると考えると, 答えは $(a+b)Ck$ となります。なお, 二項係数 nCk の実装には前処理 O(n), 1 度の計算 O(1) の方法と, O(k) で求める方法があります。

5.2 実装例

https://atcoder.jp/contests/abc145/submissions/24427318

6 Tough Journey

6.1 解法

ペットボトルは K 本しか無いので, 街 i で飲む水は街 $i-K+1$ から街 i の間で補給する必要があります。よって, 街 i で飲む水のコストが街 $i-K+1$ から街 i の間の A の最小値とすれば, 必要な資金は最小となります。最小値を求めるのはセグ木で $O(NlogN)$ ですが, deque を用いた, スライド最小値という尺取法のような手法で $O(N)$ にすることもできます。(蟻本 p300 参照)

6.2 実装例

https://atcoder.jp/contests/code-festival-2018-final-open/submissions/24437240

```
#include <bits/stdc++.h>
#define int long long
#define rep(i, n) for (int i = 0, i##_len = (n); i < i##_len; i++)
using namespace std;

signed main() {
  int N, K;
  cin >> N >> K;
  vector<int> A(N);
  rep(i, N) cin >> A[i];
  deque<int> slide_min;
  int L = 0;
  int ans = 0;
  for (int R = 0; R < N; R++) {
    while (slide_min.size()) {
      if (A[*rbegin(slide_min)] >= A[R]) {
        slide_min.pop_back();
      } else {
        break;
      }
    }
    slide_min.push_back(R);
    if (R - L + 1 > K) {
      if (slide_min[0] == L) slide_min.pop_front();
      L++;
    }
    ans += A[slide_min[0]];
  }
  cout << ans << endl;
}
```

7　ネタだけ食べたい寿司

7.1　解法

ネタを選ぶ回数が合計 M 回となる位置を i とします。このとき, すべての皿の上にネタが乗っているので $i+1$ 番目移行は何も食べることができません。よって, このとき幸福度の最大値は i 以下の Y_j の合計 + 普通に食べるのではなく, ネタのみ食べる恩恵 $X_j - Y_j$ を上から降順に $M-1$ 個足したものとなります。i はループで回して, ソートされた $X_j - Y_j$ は優先度付きキュー等で管理していけば, $O(NlogM)$ で解くことができます。

7.2　実装例

```
#include <bits/stdc++.h>
#define int long long
#define rep(i, n) for (int i = 0, i##_len = (n); i < i##_len; i++)
using namespace std;

signed main() {
  int N, M;
  cin >> N >> M;
  vector<int> X(N), Y(N);
  rep(i, N) cin >> X[i] >> Y[i];
  priority_queue<vector<int>, vector<vector<int>>, greater<>> pos;
  int ans = 0;
  int sum_gap = 0;
  int sum_Y = 0;
  rep(i, N) {
    pos.push({X[i] - Y[i], X[i]});
    sum_gap += X[i] - Y[i];
    sum_Y += Y[i];
    ans = max(ans, sum_gap + sum_Y);
    if (pos.size() == M) {
      sum_gap -= pos.top()[0];
      pos.pop();
    }
  }
  cout << ans << endl;
}
```