YNU CPC 2021 VC 008 A + ex
解説

電子情報システム EP B3 ky

2021 年 6 月 18 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. 教室
   https://img.atcoder.jp/gigacode-2019/editorial.pdf

2. ... (Triple Dots)
   https://img.atcoder.jp/abc168/editorial.pdf

3. Traveling AtCoDeer Problem
   https://img.atcoder.jp/abc064/editorial.pdf

4. 総和
   https://img.atcoder.jp/data/abc/037/editorial.pdf

5. AtColor
   https://www.slideshare.net/chokudai/abc014

6. Exactly N points
   https://img.atcoder.jp/data/other/cf16-final/editorial.pdf

7. チョコレート
   https://www.slideshare.net/chokudai/arc025

8. JOIOJI
   https://www.ioi-jp.org/camp/2014/2014-sp-tasks/2014-sp-d3-joioji-review.pdf

1 教室

1.1 問題 url

https://atcoder.jp/contests/gigacode-2019/tasks/gigacode_2019_a

1.2 解法

$B \times B$ の面積の教室が A 個ありますから, 合計の面積は $B \times B \times A$ になります。これを cout を用いて出力すれば良いです。時間計算量 O(1):定数です。

1.3 実装例

```
#include <bits/stdc++.h>
using namespace std;

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

2 ... (Triple Dots)

2.1 問題 url

https://atcoder.jp/contests/abc168/tasks/abc168_b

2.2 解法

まず, if 文を用いて文字列 S の長さが K 以下であるかどうか判定します。S が格納される string 型の長さは S.size() で取り出すことができます。判定後は K 以下である場合はそのまま cout で出力。K より大きい場合は部分文字列を取り出す関数 S.substr() 等を使うことができます。a 番目の index から b 文字取り出したいとき, 引数には S.(a, b) のようにします。つまり, 今回の場合は 0 番目から K 個の文字列なので, S.(0, K) です。他にも例えば空の文字列に for 文を用いて S[0] から S[K-1] までを足し合わせていく等して S の先頭 K 文字を取り出すこともできます。最後に文字列"..."を足し合わせて出力します。時間計算量は O($|S|$) です。($|S|$ は文字列の大きさ)

2.3 実装例

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  int K;
  string S;
  cin >> K >> S;
  if (S.size() <= K) {
    cout << S << endl;
  } else {
    cout << S.substr(0, K) + "..." << endl;
  }
}
```

S.substr(0, K) の部分は次のようにすることもできます。

```
string ans;
for (int i = 0; i < K; i++) ans += S[i];
cout << ans + "..." << endl;
```

3 Traveling AtCoDeer Problem

3.1 問題 url

https://atcoder.jp/contests/abc064/tasks/abc064_b

3.2 解法

1 次元座標上に家が並んでいるので, 座標が最小の家から初めて最大の家まで 1 直線に向かってそこで終了すれば, その間に全ての家があるので最短の移動距離になります。すなわち, 答えは a の最大値-a の最小値になります。max_element, min_element という関数を用いるもしくは max, min という変数を置いて for 文で更新していくことで最大値, 最小値が得られます。時間計算量は O(N) です。

3.3 実装例

max_element, min_element を用いる場合

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  int N;
  cin >> N;
  vector<int> a(N);
  for (int i = 0; i < N; i++) cin >> a[i];
  cout << max_element(begin(a), end(a)).operator*() - min_element(begin(a), end(a)).operator*() <<
      endl;
}
```

愚直に更新する場合

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  int N;
  cin >> N;
  vector<int> a(N);
  for (int i = 0; i < N; i++) cin >> a[i];
  int max = a[0];
  int min = a[0];
  for (int i = 0; i < N; i++) {
    if (a[i] > max) max = a[i];
    if (a[i] < min) min = a[i];
  }
  cout << max - min << endl;
}
```

4 総和

4.1 問題 url

https://atcoder.jp/contests/abc037/tasks/abc037_c

4.2 解法

1. 全探索をする場合 (時間制約に間に合いません)
   全探索をするとおおよそ K 回の足し算を N-K+1 回行うことになるので, 合計の計算回数は K(N-K+1) になります。よって計算量は $O(KN)$ です。例えば $N = 10^5, K = 5 \times 10^5$ のときには計算回数は $K(N-K+1) \sim 2.5 \times 10^9$ となり, 時間制約に間に合いません。時間制約に間に合わせるためにはおおよそ 10^7 から 10^8 以下の計算回数にする必要があります。

2. 累積和を使う場合
   講習会で説明した累積和を使うことで, 毎回の長さ K の部分列の総和を 1 回の引き算で得ることができます。よって合計の計算量は O(N) になり, 十分高速に動作します。(ただし, 初めに累積和の配列を作るとき (前処理) に N 回の計算が必要になります。) また, オーバーフローを防ぐために long long 型を使うことに注意します。

4.3 実装例

講習会とは少し違う実装になっている点に注意して下さい。
a の配列そのものを累積和として定数倍高速化を図っています。

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  int N, K;
  cin >> N >> K;
  vector<long long> a(N + 1);
  for (int i = 0; i < N; i++) cin >> a[i + 1];
  for (int i = 0; i < N; i++) a[i + 1] += a[i];
  long long ans = 0;
  for (int i = 0; i < N - K + 1; i++) ans += a[i + K] - a[i];
  cout << ans << endl;
}
```

講習会通りの実装は次の通りです。
https://atcoder.jp/contests/abc037/submissions/23099134

5 AtColor

5.1 問題 url

https://atcoder.jp/contests/abc014/tasks/abc014_3

5.2 解法

先程の累積和を使った問題と同様に, 全探索では間に合いません。よって Imos(いもす) 法を使います。時間計算量は O(n) です。

5.3 実装例

最終行の max_element は最大値を返す関数です。for 文と if 文で最大値を見つけることもできます。

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  int n, a, b;
  cin >> n;
  vector<int> cum_sum(1000000 + 2);
  for (int i = 0; i < n; i++) {
    cin >> a >> b;
    cum_sum[a]++;
    cum_sum[b + 1]--;
  }
  for (int i = 0; i < 1000000 + 1; i++) cum_sum[i + 1] += cum_sum[i];
  cout << max_element(begin(cum_sum), end(cum_sum)).operator*() << endl;
}
```

6 Exactly N points

6.1 問題 url

https://atcoder.jp/contests/cf16-final/tasks/codefestival_2016_final_b

6.2 解法

最小となる”解く問題の配点のうちの最大値”は $1+2+...+i=\frac{i(i+1)}{2} \geq N$ を満たす最小の i です。これは次のように示すことができます。以下, i を 1 から $1+2+...+i$ が N 以上となるまで試していきます。時間計算量は $O(\sqrt{N})$ です

(i) $1+2+...+i < N$ のとき
1 から i までのすべての数を足しても N に達しないので明らかに和が N となるような {1, 2, ... , i} の部分集合は存在しない。

(ii) $1+2+...+i \geq N$ となったとき
$S=1+2+...+i$, また, $N=S-a$(a は現在の S から抜かす数の和) とすると

$$S-i < N \leq S$$

$$\Rightarrow 0 \leq a \leq i-1$$

よって 0 から i-1 のうちのどれかの数を 1 つだけ抜かすことで, 答えとなる部分集合が求まります。

6.3 実装例

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  int N;
  cin >> N;
  for (int i = 1; i < 10000000; i++) {
    if (i * (i + 1) / 2 >= N) {
      for (int j = 1; j <= i; j++) {
        if (i * (i + 1) / 2 - j == N) continue;
        cout << j << endl;
      }
      break;
    }
  }
}
```

7　チョコレート

7.1　問題 url

https://atcoder.jp/contests/arc025/tasks/arc025_2

7.2　解法

切り出し方は HW 個のマスから 2 つ選んでその長方形を取り出せば良いので, 全部で ${}_{HW}C_2 = O(H^2W^2)$ 通りです。個々の切り出し方ごとに和をとる $= O(HW)$ と間に合わないですが, 2 次元累積和を使うと, $O(1)$ である長方形に含まれる総和が分かります。今回はブラックとホワイトがあるので, 個別に累積和用の 2 次元配列を作れば良いです。以上より時間計算量は $O(H^2W^2)$ となり制約に間に合います。また, チョコの色によって正負を反転させて, 長方形中の総和が 0 になるか判定することで 2 次元配列を 1 つで済ませることもできます。

7.3　実装例

私の微妙な実装例
https://atcoder.jp/contests/arc025/submissions/23545846
Rogi 君の実装例 (入力と累積和構築を同時に, 正負の反転も)
https://atcoder.jp/contests/arc025/submissions/23545674

8 JOIOJI

8.1 問題 url

https://atcoder.jp/contests/joisc2014/tasks/joisc2014_h

8.2 解法

S[i]=文字列の i 文字目までに含まれる [J の数, I の数, O の数](要するに累積和)(S[i] は列ベクトルです) とします。すると, 区間 [L, R] に含まれる J, O, I の個数が等しいためには S[R] - S[L] = [定数, 定数, 定数](全て等しい定数で R - L = 3 * 定数) となる必要があります。例えば S[L] = [1, 2, 3] のとき, S[R] = [2, 3, 4] は S[R] - S[L] = [1, 1, 1] より先の条件を満たし, [L, R] に'J', 'O', 'I' が 1 つずつ含まれます。よって A[i] を S[i] のすべての要素から S[i] の最も小さい要素を引いたものに置き換えると A[i] が等しい i の集合の最大値-最小値が解の候補になります。例えば, 先の例では A[R] = [2, 3, 4] - [2, 2, 2] = [0, 1, 2] でこれは A[L] と一致します。具体的な実装案としては A[i] を map<A[i], i の集合 > のような形で持つことで計算量 O(NlogN) です。

8.3 実装例

```
int main() {
  int N;
  string S;
  cin >> N >> S;
  vector<vector<int>> JOI(N + 1, vector<int>(3));
  for (int i = 0; i < N; i++) {
    JOI[i + 1] = JOI[i];
    if (S[i] == 'J') JOI[i + 1][0]++;
    if (S[i] == 'O') JOI[i + 1][1]++;
    if (S[i] == 'I') JOI[i + 1][2]++;
  }
  map<vector<int>, vector<int>> calc;
  for (int i = 0; i < N + 1; i++) {
    int MIN = min({JOI[i][0], JOI[i][1], JOI[i][2]});
    for (int j = 0; j < 3; j++) JOI[i][j] -= MIN;
    calc[JOI[i]].emplace_back(i);
  }
  int ans = 0;
  for (auto &i: calc) {
    int val = max_element(begin(i.second), end(i.second)).operator*() - min_element(begin(i.second),
        end(i.second)).operator*();
    if (val > ans) ans = val;
  }
  cout << ans << endl;
}
```

9 魚の生息範囲 (Fish)

9.1 問題 url

https://atcoder.jp/contests/joi2013yo/tasks/joi2013yo_e

9.2 解法

3 次元の座標圧縮です。x 座標, y 座標, z 座標を圧縮して重複を表す直方体はおおよそ N^3 個になるので, あとは愚直に計算できます。

9.3 実装例

すみません, かなり雑です。
https://atcoder.jp/contests/joi2013yo/submissions/22172054

9.4 公式解説

https://www.ioi-jp.org/joi/2012/2013-yo/2013-yo-t5/review/2013-yo-t5-review.html