JOIG 2021/2022 本選 3問目

Voting 解説

大佐 健人

問題概要

- N 人が順番に「賛成」/「反対」に投票する
- 直前に投票した X_i 人のうち Y_i 人以上が賛成→賛成に投票
- そうでない→反対に投票
- 賛成に投票したのは何人？

小課題 1 ($N \leq 3000$)

- 問題文通りに**シミュレーション**

- for 文で $1, 2, \ldots, N$ 番目の人を順に見ていく
- i 番目の人について、どちらに投票するか判定したい

小課題 1 ($N \leq 3000$)

- 配列 A に、それぞれの人の投票先を保存しておく
  - 賛成なら $A[i] = 1$, 反対なら $A[i] = 0$ など
- for 文で、$A[i-1], A[i-2], \ldots, A[i-Xi]$ の中の賛成の個数を数える
  - 計算量は $O(N)$
- 賛成が Y_i 人以上なら賛成に投票
  - $A[i]$ を更新

小課題 1 ($N \leq 3000$)

- 全員の投票先が分かるので、解けた
- このような処理は二重の for 文でできる

- 計算量は $O(N^2)$
  - i を 1 から N まで動かし、各 i について判定に $O(N)$ かかるため
  - この計算量だと満点は間に合わない…

- 28点

小課題 1 ($N \leq 3000$)

- 実装例（C++）

```cpp
int N;
cin >> N;
vector<int> X(N), Y(N), A(N);
for (int i = 0; i < N; i++) {
    cin >> X[i] >> Y[i];
}
int res = 0;
for (int i = 0; i < N; i++) {
    int cnt = 0;
    for (int j = i - X[i]; j < i; j++) {
        cnt += A[j];
    }
    if (cnt >= Y[i]) {
        //賛成に投票
        res++;
        A[i] = 1;
    }
}
cout << res << endl;
```

小課題 2 ($X_i = i - 1$)

- Q.この小課題の意味は？

- A. それぞれの人について、今までに投票した**全員**の中で何人賛成したかが分かれば、投票先が分かる

- 「**今までに賛成した人数**」を変数として持てばよい！
  - これを変数 c とする

小課題 2 ($X_i = i - 1$)

- 条件：$Y_i \leq c$ なら賛成、そうでないなら反対
- 賛成なら c を 1 増やす

- これを N 人について、順番に行えば解ける

- 32点

小課題 2 ($X_i = i - 1$)

- 実装例（C++）

```cpp
int N;
cin >> N;
vector<int> X(N), Y(N);
for (int i = 0; i < N; i++) {
    cin >> X[i] >> Y[i];
}
int c = 0, res = 0;
for (int i = 0; i < N; i++) {
    if (Y[i] <= c) {
        //賛成に投票
        c++;
        res++;
    }
}
cout << res << endl;
```

満点解法

- $N \leq 500000$ →小課題 1 の $O(N^2)$ は間に合わない
- それぞれの人について、投票先を高速に決めたい

- 小課題 2 のアイデアを応用できないか…？

満点解法

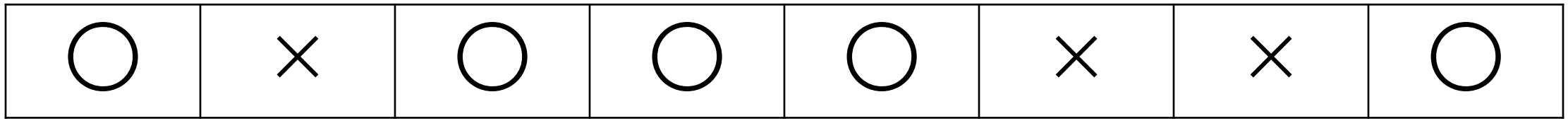
- 小課題 2 では $X_i = i - 1$ だったから、「$i - 1$ 番目までに何人賛成したか」を使えば判定できた

- 重要な事実：

(直前 X_i 人で賛成した人数)

$=$ ($i - 1$ 番目までに賛成した人数) $-$ ($i - X_i - 1$ 番目までに賛成した人数)

満点解法



直前 5 人のうち賛成した人数 = 3

||

8番目までに賛成した人数 = 5

|

3番目までに賛成した人数 = 2

（○：賛成、×：反対）

満点解法

- $S[i]$：i 番目までに賛成した人数 という配列を作っておく
- ”重要な事実” を書き直すと…

- (直前 X_i 人で賛成した人数)$= S[i-1] - S[i-X_i-1]$

- $O(1)$ で計算できた！

満点解法

- $S[i]$ 自体も、$S[i-1]$が分かっていれば $O(1)$ で計算できる
  - 賛成なら $S[i] = S[i-1] + 1$、反対なら $S[i] = S[i-1]$

- このような高速化テクニックを「**累積和**」という
  - 重要！

- 計算量は全体で $O(N)$ →間に合う
- 100点

満点解法

- 実装例（C++）　（入力部が小課題1,2と違うことに注意）

```
int N;
cin >> N;
vector<int> X(N + 1), Y(N + 1), S(N + 1);
for (int i = 1; i <= N; i++) {
    cin >> X[i] >> Y[i];
}
int res = 0;
for (int i = 1; i <= N; i++) {
    S[i] = S[i - 1];
    if (S[i - 1] - S[i - 1 - X[i]] >= Y[i]) {
        res++;
        S[i]++;
    }
}
cout << res << endl;
```