ACM ICPC2016 国内予選　問題 B　当選者を探せ

http://icpc.iisf.or.jp/past-icpc/domestic2016/problems/all_ja.html#section_B

- 開票の途中で、最高得票と次点の得票の差が未開票数を上回れば当選者を決定できる。
- １票ずつ開票して各候補者の得票を集計していく。
  - 毎回、全候補者の得票数を見て最高得票数と次点の得票数を求めて当選者を決定できるかどうかを判定する。O(候補者数×投票総数)となるが、候補者数≦26, 投票総数≦100 なので問題無い。（pb1.c）
  - 開票した票を得た候補者の得票数 x をベースにして、その時点での最高得票数(first)と次点の得票数(second)を更新する。具体的には、
    - x > first なら first = x とし、
    - x ≦ first で x > second なら second = x

    とすれば良い。O(投票総数)となる。(pb2.c)

【pb1.c】

```
/*****************************************************
 * ACM ICPC2016 国内予選 問題 B 当選者を探せ
 * http://icpc.iisf.or.jp/past-icpc/domestic2016/problems/all_ja.html#section_B
 * Filename:      pb1.c
 * Compile:       cc –Wall pb1.c
 * Check:         ./a.out < B0 > B0.out; diff B0.out B0.ans
 * Algorithm:     top 2 名の候補者の得票数の差が未開票数以上になれば当選者が確定
 *                1 票開票する度に top 2 名の候補者の得票の差を求め未開票数と比較
 *                O(候補者数×投票数) 候補者数<=26, 投票数<=100 なので OK
 *****************************************************/

#include <stdio.h>
#define NC 26                       // 候補者数の最大値
#define MAXN 100                    // 投票数の最大値
#define SIZE (MAXN*2+1)             // 1 行の長さの最大値 + 1
int main()
{
    int votes[NC];                  // 各候補者の得票数
    int n;                          // 投票数
    int c;                          // 候補者番号
    char line[SIZE];                // 投票先の候補者名リスト
    while(1){
        int i,j;
        int first;                  // 最大の得票数
        int firstID;                // 最大の得票数を獲得した候補者番号
        int second;                 // 2 番目に多い得票数
        fgets(line, MAXN*2+1, stdin);   // 1 行入力
        sscanf(line, "%d", &n);         // 投票数を取得
        if(n==0) break;                 // n==0 なら終了
        fgets(line, MAXN*2+1, stdin);   // 投票先の候補者名リストの入力
        for(i=0; i<NC; i++)
            votes[i] = 0;               // 各候補者の得票数を 0 に初期化
        for(i=0; i<n; i++){             // 1 票ずつ開票
            c = line[2*i] - 'A';        // 候補者番号に変換
            votes[c]++;                 // その候補者の得票数を 1 増やす
            first = second = 0;         // 最大と 2 番目の得票数を 0 に初期化
            // 最大の得票数とその候補者番号を求める
            for(j=0; j<NC; j++){
                if(votes[j] > first){
                    first = votes[j];
                    firstID = j;
                }
            }
            // 2 番目の得票数を求める
            for(j=0; j<NC; j++){
                if(j == firstID) continue;      // 得票数が最大の候補者は除外
                if(votes[j] > second){
                    second = votes[j];
                }
            }
            // 1 番目と 2 番目の得票数の差が未開票数より大きければ当選者確定
            if(first - second > n - i - 1) break;
        }
```

```
            if(i<n){            // 当選者確定
                printf("%C %d¥n", firstID + 'A', i+1);
            }
            else                // 引き分け
                printf("TIE¥n");
        }
        return 0;
}
```

【pb2.c】

```
/*****************************************************
 * ACM ICPC2016 国内予選 問題 B 当選者を探せ
 * http://icpc.iisf.or.jp/past-icpc/domestic2016/problems/all_ja.html#section_B
 * Filename:      pb2.c
 * Compile:       cc –Wall pb2.c
 * Check:         ./a.out < B0 > B0.out; diff B0.out B0.ans
 * Algorithm:     top 2 名の候補者の得票数の差が未開票数以上になれば当選者が確定
 *                 1 票開票する度に，票が入った候補者の得票数 x をベースに
 *                top 2 名の候補者の得票数を更新しその差を求め未開票数と比較
 *                (if (x > first) first = x; else if (x > second) second = x;
 *                O(投票数) 候補者数<=26, 投票数<=100 なので OK
 *****************************************************/

#include <stdio.h>
#define NC 26                       // 候補者数の最大値
#define MAXN 100                    // 投票数の最大値
#define SIZE (MAXN*2+1)             // 1 行の長さの最大値＋ 1
int main()
{
    int votes[NC];                  // 各候補者の得票数
    int n;                          // 投票数
    int c;                          // 候補者番号
    char line[SIZE];                // 投票先の候補者名リスト
    while(1){
        int i;
        int first;                  // 最大の得票数
        int firstID;                // 最大の得票数の候補者番号
        int second;                 // 2 番目に多い得票数
        fgets(line, MAXN*2+1, stdin);   // 1 行入力
        sscanf(line, "%d", &n);         // 投票数を取得
        if(n==0) break;                 // n==0 なら終了
        fgets(line, MAXN*2+1, stdin);   // 投票先の候補者名リストの入力
        for(i=0; i<NC; i++)
            votes[i] = 0;               // 各候補者の得票数を 0 に初期化
        first = second = 0;             // top と 2 番手の得票数を 0 に初期化
        firstID = 0;                    // 最大の得票数の候補者番号を 0 に初期化
        for(i=0; i<n; i++){             // 1 票ずつ開票
            c = line[2*i] - 'A';        // 候補者番号に変換
            votes[c]++;                 // その候補者の得票数を 1 増やす
            if(votes[c]>first){         // top の得票数を超えたら
                first = votes[c];       // top の得票数と
                firstID = c;            // その候補者番号を更新
            }
            else if(votes[c] > second)  // そうではなく、
                                        // 2 番手の得票数を超えたら
                second = votes[c];      // 2 番手の得票数を更新
            // 1 番目と 2 番目の得票数の差が未開票数より大きければ当選者確定
            if(first - second > n - i - 1) break;       // 当選者確定
        }
        if(i<n){            // 当選者確定（for loop を break で抜けた）
            printf("%C %d¥n", firstID + 'A', i+1);
        }
        else                // 引き分け
```

```
                printf("TIE¥n");
        }
        return 0;
}
```