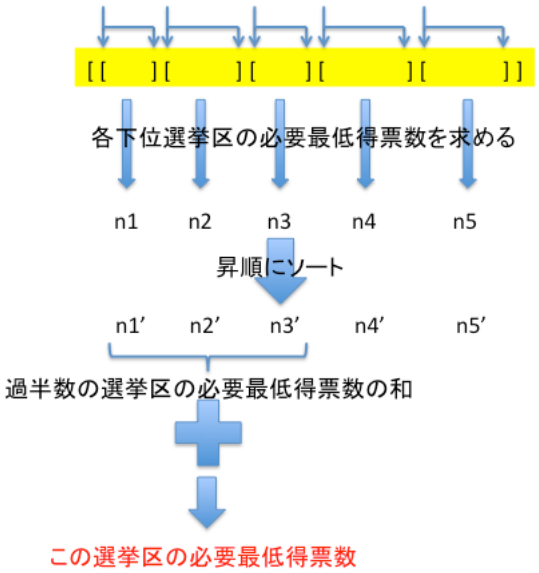
ACM ICPC2013 国内予選　問題 C 階層民主主義

- 基本アルゴリズム
  1. 最下位（第 1 段階）の選挙区ならば有権者数の過半数がこの選挙区の最低必要得票数である
  2. 最下位の選挙区でない場合は、下位の各選挙区に対して、最低必要得票数を再帰的に求める
  3. Step 2 で求めた下位の各選挙区の最低必要得票数を昇順にソートする
  4. 過半数の下位選挙区の最低必要得票数の和（昇順に加算）がこの選挙区の最低必要得票数である
- 左括弧に対応する右括弧の位置を求める方法
  - count を 0 にセット
  - 左括弧の位置から順に以下を繰り返す
    - 左括弧なら count を 1 増やす
    - 右括弧なら count を 1 減らし、count が 0 になれば、求める右括弧である



【プログラム例】

```
// ACM ICPC2013 Domestic Problem C
// http://sparth.u-aizu.ac.jp/icpc2013/d_problems.php?lang=jp#section_C
// Filename:            pc.c
// Compile:             cc pc.c
// Execution:            ./a.out < C0 > C0.result
// Check:   diff C0.result C0.ans

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX_LEN 10000                   // 選挙区割定義の最大長
#define MAX_NC     3333                 // 下位選挙区数の最大値
char buff[MAX_LEN+1];                   // 選挙区割定義の文字列

// 整数配列を qsort する為の比較関数
int compar(const void *p1, const void *p2)
{
   int *ip1 = (int *)p1;
   int *ip2 = (int *)p2;
   if(*ip1 < *ip2) return -1;
   else if(*ip1 == *ip2) return 0;
   else return 1;
}

// lp 位置の左鍵括弧に対応する右鍵括弧の位置を求める
int find_rp(int lp)
{
   int count = 1;       // 対応する右鍵括弧が見つかっていない左鍵括弧の個数
   int rp = lp;
   char c;
   while(count){
      c = buff[++rp];
      if(c == '[') count++;
      else if(c == ']') count--;
   }
   return rp;
}

// (from,to)で指定される選挙区を制するのに最低必要な得票数を求める
int mc(int from, int to)
{
   int nv[MAX_NC];              // 下位選挙区の最低必要得票数
   int nd;                      // 下位の選挙区数
   int v;
   int lp,rp;                                   // 下位選挙区の左右鍵括弧の位置
   int i;

   if(buff[from+1] == '['){         // 下位選挙区が存在
      nd = 0;
      lp = from+1;                  // 最初の下位選挙区の左鍵括弧の位置
      while(lp < to){               // lp がこの選挙区の下位選挙区の間繰り返す
```

```
        rp = find_rp(lp);                       // lp に対応する右鍵括弧の位置を求める
        nv[nd++] = mc(lp,rp);     //  下位選挙区(lp,rp)の最低必要得票数を求める
        lp = rp+1;                //  次の下位選挙区へ
      }
      //  この時点で、nd は下位選挙区の個数。
      // nv[i]は、各下位選挙区を制するのに最低必要な票数
      qsort(nv, nd, sizeof(int), compar);                   // nv をソート
      //  過半数の下位選挙区を制するのに最低必要な票数を求める
      v = 0;
      for(i=0; i<(nd+1)/2; i++)
        v += nv[i];
      return v;                 // v が求める値
    }
    else {                      //  これが最下位の選挙区
      sscanf(&buff[from+1], "%d", &v);       // v =  有権者数
      return (v+1)/2;           //  この選挙区の過半数の票数を返す
    }
}

int main()
{
    int n,i,ans;
    scanf("%d¥n", &n);                                  // n  はデータセット数
    for(i=0; i<n; i++){
      scanf("%s¥n", buff);              //  データセットの読み込み
      ans = mc(0, strlen(buff)-1);//  解を求める
      printf("%d¥n", ans);
    }
}
```