ABC_236

D-Dance

直接搜索会出现多组重复数据导致超时
如果每次从当前**未选中**人中的**最小**的编号
作为一对人中第一个,并依次向下枚举
可以避免最后整体选择重复
对选出的整体数据进行验证即可.

```
const int max_n = 2*8+2;

int aff[max_n][max_n],ans;
bool flag[20];
int n;
std::vector<std::pair<int,int>> parten;

void dfs(int deep){
    if(deep == n+1){
        int temp = 0;
        for(auto it : parten)
            temp ^= aff[it.first][it.second];
        ans = std::max(ans,temp);
        return;
    }
    int l;
    for(int i = 1;i < 2*n;i++){
        if(!flag[i]){
            l = i;
            break;
        }
    }
    flag[l] = 1;
    for(int i = 1;i <= 2*n;i++){
        if(!flag[i]){
            parten.push_back({l,i});
            flag[i] = 1;
            dfs(deep+1);
            parten.pop_back();
            flag[i] = 0;
        }
    }
    flag[l] = 0;
}

int main(){
    scanf(" %d",&n);
    for(int i = 1;i < 2*n;i++)
        for(int j = i+1;j <= 2*n;j++)
            scanf(" %d",&aff[i][j]);
```

```
    dfs(1);
    printf("%d\n",ans);
    return 0;
}
```

E-Average and Median

可以使用二分搜索来得到答案
关键在答案的检查上

- 平均数的检查:
  因为必须至少选择第**i**和第**i+1**张中一张牌
  使用dp_e[i] 表示到i张牌时最大的平均数
  用add[i]表示第**i**张牌加入时的贡献
  所以有:

```
dp_e[i] = max(dp_e[i-1]+add[i],dp_e[i-2]+add[i]);
```

对dp_e[n],dp_e[n-1]检查

- 中位数的检查:
  it为待检查数
  当 **b[i]** = (**array[i]** >= **it**) ? 1 : -1;
  用额外标记**flag**去掉**i**或**i+1**中的小于**it**的数,表示不选
  对b[ ]的总和进行检查

```
const int max_n = 1e5+10;
const double eps = 1e-6;

int n,array[max_n];
double dp_e[max_n];

bool check_mid(int it){
    int sum_ = 0,flag = 1;
    for(int i = 1;i <= n;i++){
        if(flag && array[i] < it){
            flag = 0;
            continue;
        }
        else{
            flag = 1;
            sum_ += (array[i] >= it ? 1 : -1);
        }
    }
    return sum_ > 0;
}

bool check_ever(double it){
```

```
    dp_e[1] = 1.0 * array[1] - it;
    for(int i = 2;i <= n;i++){
        double temp = 1.0 * array[i]-it;
        dp_e[i] = std::max(dp_e[i-1]+temp,dp_e[i-2]+temp);//
    }
    return (dp_e[n]+eps > 0 || dp_e[n-1]+eps > 0);
}

void bin_mid(){
    int l = 1, r = 1e9;
    while(l <= r){
        int mid = (l+r)>> 1;
        if(check_mid(mid))
            l = mid+1;
        else
            r = mid-1;
    }
    printf("%d\n",l-1);
}

void bin_ever(){
    double l = 0 , r = 1e9;
    while(r-l > eps){
        double mid = (l+r)/2;
        if(check_ever(mid))
            l = mid;
        else
            r = mid;
    }
    printf("%.9f\n",l);
}

int main(){
    scanf(" %d",&n);
    for(int i = 1;i <= n;i++){
        scanf(" %d",&array[i]);
    }
    bin_ever();
    bin_mid();
    return 0;
}
```