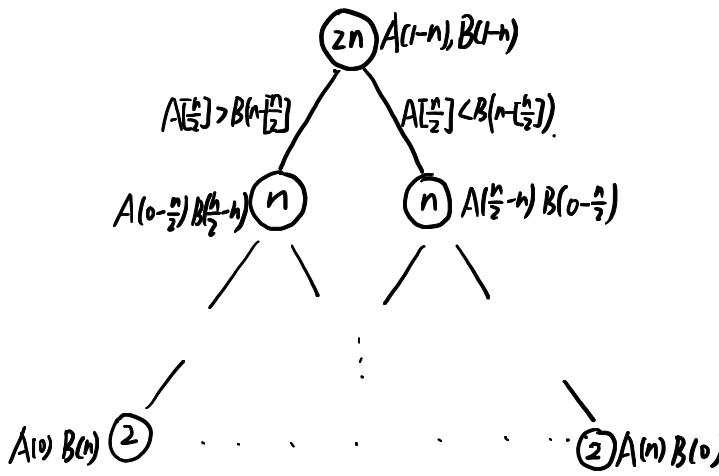


算法设计与分析 Assignment 1 孔静 2014K8009929022

1.

- (a) 询问 A 数据库 $\lceil \frac{n}{2} \rceil$ 大数据，询问 B 数据库 $n - \lceil \frac{n}{2} \rceil$ 大数据
两者比较，若 $A > B$, 去掉 $A \lceil \frac{n}{2} \rceil$ 之后与 $B n - \lceil \frac{n}{2} \rceil$ 之前的数据。
重复之前步骤，经过 $2 \cdot \log_2 n$ 次即可知 AB 数据库的中位数。

(b)



(c) 不妨设之为 AB 数据库。

比较 A, B 数据库的中位数 ($\lceil \frac{n}{2} \rceil$ 个, $n - \lceil \frac{n}{2} \rceil$ 个)。

若 $A > B$, 那么对于 $A \lceil \frac{n}{2} \rceil$ 来说, 在它之前有 A 数据库 $\lceil \frac{n}{2} \rceil - 1$ 个数, 以及 B 数据库至少 $n - \lceil \frac{n}{2} \rceil$ 个, 说明 $A \lceil \frac{n}{2} \rceil$ 在整体中至少是第 n 位数。
若 $A < B$, 同理 $B \lceil \frac{n}{2} \rceil$ 在整体中至少是第 n 位数。

若 $A = B$, 显然 $A \lceil \frac{n}{2} \rceil = B \lceil \frac{n}{2} \rceil$ 就是中位数。

当 A=B 时, 去掉大数之上 (从 1 小数之下) 的数, 两边数量相等, 且不影响中位数 (中间数仍然是中位数)
所以可循环至整体只剩下 2 个数时判断即可。

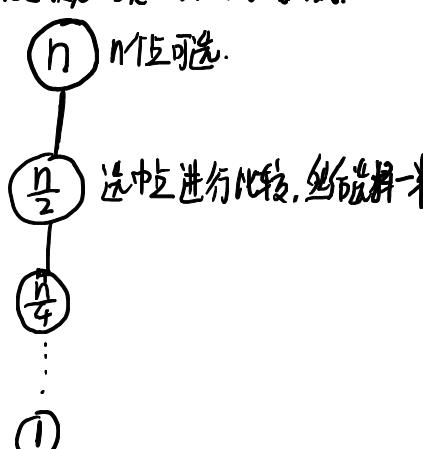
综上可求得中位数。

$$(d) T(n) = T\left(\frac{n}{2}\right) + 2 \therefore O(\log n)$$

2. (1)

- (a) 先找到边缘点如 X 最小那点 (若有 2 个取一个)
然后以这个点为原点建立极坐标系, 显然共点在 0~180 度范围内。
将其点以角排序, 一个点找过去, 能找到不同分界点..
并满足一条经过找人与鱼的线一侧人鱼相等。

(b)



```

int findmid(Aleft, Aright, Bleft, Bright) {
    int Amid = Aleft + (Aright - Aleft)/2;
    int Bmid = Bright - (Bright - Bleft)/2;
    int a = ask(Amid); //询问 A 数据库 Amid 的值。
    int b = ask(Bmid);
    if (Amid == Aleft) //A 数据库只剩 1/2 份。
        if (a > b)
            return b;
        else
            return a;
    if (a == b)
        return a;
    else if (a > b)
        return findmid(Aleft, Amid, Bmid, Bright);
    else
        return findmid(Amid, Aright, Bleft, Bmid);
}

int main {
    printf("%d", findmid(1, n, 1, n));
    //输出 A, B 数据库第 n 小的数。
}

```

```

int min (n) {
    int t=0,k;
    for (k=t; k<n; k++)
        if (a[k]<=a[t]) t=k;
}

```

```

int find_the_line (int left, int right) {
    int mid = (left+right)/2;
    if (c[mid][0] == c[mid][1]) return mid;
    else if (c[mid][0] < c[mid][1])
        return find_the_line (left, mid);
    else
        return find_the_line (mid, right);
}

int main (void) {
    min (n); //找到人的位置。
    sort (n); //根据角度排序, 将所有位置按从小到大排入 C[0][i]。
    find_the_line (0, m); //询问 j=0 人 j+1 人
}

```

min (n); //找到人的位置。
sort (n); //根据角度排序, 将所有位置按从小到大排入 C[0][i]。
find_the_line (0, m); //询问 j=0 人 j+1 人

(c) 如(a)所述，必能找到边缘点，如 x 坐标最小，若只有一个人（ x 是离以下人最近的）， $x=x_{\min}$ 右侧所有点，有对称，也能找到切线使其余所有点都在一侧，那么那些点的 x 值在 $0 \sim 180$ 以内，否则需要舍弃（而非直线）。

归排序后，最近2个点（ θ 最大及最小），若有一个是鬼，显然已找到这样的直线，即一侧人=鬼。

若均为人，那么鬼中角度最小的鬼（与人连线）右侧人数 > 鬼数 / 角度最大的鬼右侧鬼数 > 人数。

所以中间必能找到一点，人数相等。

(d) 排序 $O(n \log n)$, 其余 $O(n)$ \therefore 共 $O(n \log n)$

(2) 重复1过程，即可找到 $T'(n) = T(n) + T(n-1) \dots = O(n^2 \log n)$.

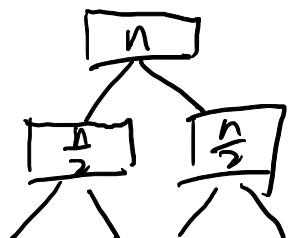
3

(a) 不断两分，至每组1个数，左>3右，则第一个逆序，然后排序成一组。
不断合并，并计算左右合并的新逆序。

```
#include <stdio.h>
long a[100000], b[100000];
int inversions(int left, int right){
    int num = 0, k;
    if (left >= right)
        return 0;
    int mid = (left + right) / 2;
    num = num + inversions(left, mid) +
inversions(mid + 1, right);
    int i = left, j = mid + 1;
    for (k = left; k <= right; k++)
        b[k] = a[k];
    while (j <= right){
        for (k = i; k <= mid; k++)
            if (b[k] > 3 * b[j]){
                num = num +
mid - k + 1;
                j++;
                i = k;
                break;
            }
        if (k > mid && b[mid] <= 3 * b[j])
            break;
    }
    i = left; j = mid + 1;
    for (k = left; k <= right; k++){
        if ((b[i] < b[j] && i <= mid) || j >
right){
            a[k] = b[i];
            i++;
        } else{
            a[k] = b[j];
            j++;
        }
    }
    return num;
}
```

```
int main(void){
    int i;
    scanf("%d", &i);
    int j;
    for (j = 0; j < i; j++)
        scanf("%d", &a[j]);
    printf("%d", inversions(0, i - 1));
    return 0;
}
```

(b)



(c) 逆序数 = 左组内部 + 右组内部 + 左与右产生的。

虽然没错 ORZ

(d) $T(n) = 2T(\frac{n}{2}) \therefore O(n \log n)$:

4. (a) 从上往下找. 根节点与左右儿子比较大小

①根节点最小, 即局部最小点.

②左/右儿子最小, 选择左/右子树重复上述比较.

int min(Tree){

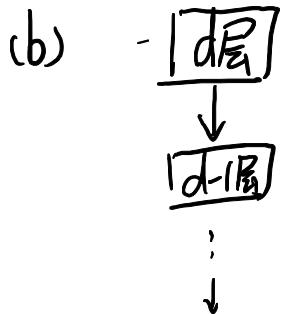
if (Tree.root < Tree.root.left && Tree.root < Tree.root.right)

return Tree.root

else if (Tree.root.left < Tree.root.right)

return (Tree.left)

else return (Tree.right); }

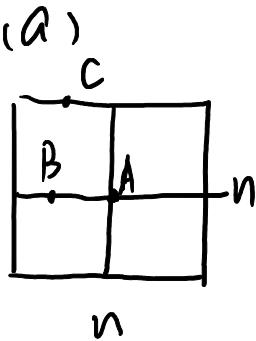


(c) 若根节点最小, 显然满足局部最小性.

若左/右儿子最小, 选择左/右子树重复过程且已有父结大于该结点的条件.
综上能找到局部最小点.

(d) 显然至多n次即 $\log_2(n+1)$ - 1 即 $O(\log n)$ 的复杂度

5



$n \times n$ 比较图上 6n - 9 个点, 找出最小点. 三种情况.

① 最小点在交点上, 即所求局部最小点. 结束

② 在 B 这种位置. 比较 B 上下两点及 B, 若 B 最小, 也为所求点,
若上方最小, 则选上方 $\frac{n}{2} \times \frac{n}{2}$ 重叠, 下方同理

③ 在 C 这种位置, 则选 C 所在 $\frac{n}{2} \times \frac{n}{2}$ 区域重叠.

struct coordinate findmin(int x, int y, int n){

 struct coordinate min = compare(int x, int y, int n); 比较过程略

 if((min.x == x || min.x == x + n / 2 || min.x == x + n - 1) && (min.y == y || min.y == y + n / 2 || min.y == y + n - 1)) return min;

 else if(min.x == x) return findmin(x, y + (min.y - y) / (n / 2), (n+1) / 2);

 else if(min.y == y) return findmin(x + (min.x - x) / (n / 2), y, (n+1) / 2);

 else if(min.x == x + n - 1) return findmin(x + n - 1, y + (min.y - y) / (n / 2), (n+1) / 2);

 else if(min.y == y + n - 1) return findmin(x + (min.x - x) / (n / 2), y + n - 1, (n+1) / 2);

 else if(min.x == x + n / 2){

 if (point(min.x, min.y) < point(min.x + 1, min.y) && point(min.x, min.y) < point(min.x - 1, min.y)) return min;

 else if (point(min.x - 1, min.y) < point(min.x + 1, min.y)) return findmin(x, y + (min.y - y) / (n / 2) * (n / 2), (n+1) / 2);

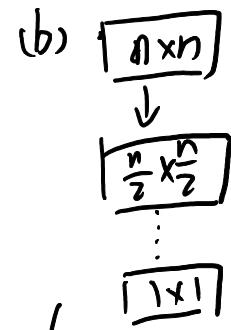
 else return findmin(x + n / 2, y + (min.y - y) / (n / 2) * (n / 2), (n+1) / 2);}

 else if(min.y == y + n / 2){

 if (point(min.x, min.y) < point(min.x, min.y + 1) && point(min.x, min.y) < point(min.x, min.y - 1)) return min;

 else if (point(min.x, min.y - 1) < point(min.x, min.y + 1)) return findmin(x + (min.x - x) / (n / 2) * (n / 2), y, (n+1) / 2);

 else return findmin(x + (min.x - x) / (n / 2) * (n / 2), y + n / 2, (n+1) / 2);}



6

(a) 递归. $f(n) = \sum_{i=0}^{n-3} f(n-1-i) \cdot f(2+i)$ ($n \geq 4$) 如 $f(2)=f(3)=1$

int function (n){

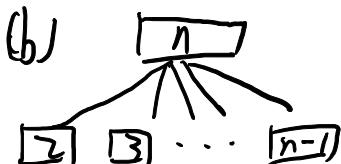
 int i, sum = 1;

 if (n >= 4)

 for (i=0; i < n-2; i++)

 sum = sum + function(n-1-i) * f(2+i);

 return n; }



(c) 每边参与且仅参与一个三角形，所以找一条边再寻找一个顶点
即可将其唯一分成2个国家，且不重复，

$$\text{所以 } f(n) = \sum_{i=0}^{n-3} f(n-1-i) \cdot f(2+i) \quad n > 4$$

(d) $T(n) = 2(T(2) + T(3) + \dots + T(n-1)) \Rightarrow T(n) = n^2 \therefore O(n^2)$

(c) 正确情况上一题 寻找过程是递减过程，必有终止，
且每次 return min 的前提已确认是局部最小

(d) $T(n) = T(\frac{n}{2}) + O(n)$ 所以 $T(n) = O(n)$.