Chapter2 分治法

对分治法思想的体会

分治法的核心思想有三步

- 确定问题的最小规模

```
if (STATEMENT) return;
```

- 划分子问题
- 合并子问题 (有的时候不需要合并)

二分搜索的应用

在顺序数组中查找值

时间复杂度：$O(nlogn)$

- Code

```
#include <bits/stdc++.h>

using namespace std;
#define ll long long
#define mod 1000000007
const ll maxn = 2e6 + 7;

ll BinSearch(ll a[], ll left, ll right, ll x) {
    ll mid;
    while (left <= right) {
        mid = (left + right) / 2;
        if (a[mid] == x) {
            return mid;
        } else if (a[mid] > x) {
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }
    return -1;
}

ll a[maxn];

int main() {
    ll x, n;
    cin>>n>>x;
    for (long long i = 0; i < n; ++i) {
        cin>>a[i];
    }
    printf("%lld", BinSearch(a, 0, n-1, x));

```

```
33
34      return 0;
35  }
```

STL中的二分搜索

- `upper_bound()` 返回的是被查序列中第一个**大于**查找值得指针
- `lower_bound()` 返回的是被查序列中第一个**大于等于**查找值的指针

```
1  int t = lower_bound(a+l,a+r,m) - a;
```

解释：在升序排列的`a`数组内二分查找$[l,r)$区间内的值为m的元素。返回m在数组中的下标。
特殊情况：

- 如果m在区间中没有出现过，那么返回第一个比m大的数的下标。
- 如果m比所有区间内的数都大，那么返回r。这个时候会越界，小心。
- 如果区间内有多个相同的m，返回第一个m的下标。

假定一个值来确定是否可行—猜测确定最优解

问题：求解满足某个条件$C(x)$的最小x

对于任意满足$C(x)$的x，如果所有的$x' >= x$也满足$C(x')$的话 $\Longrightarrow$ 可以二分求$Min(x)$

- 初始化
  - 区间左端点 $\rightarrow$ 不满足$C(x)$的值 `e.g.0`
  - 区间右端点 $\rightarrow$ 满足$C(x)$的值
- 每次取中点 $mid = (lb + rb)/2$
- 判断$C(mid)$是否满足并缩小范围，直到$[lb, rb)$足够小，ub就是最小值
- 求最大的方法类似.

$e.g.$ 派

> https://nanti.jisuanke.com/t/T1157

- Code

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4  #define ll long long
5  #define mod 1000000007
6  const double PI=acos(-1.0);
7  const ll maxn = 2e6 + 7;
8  double a[maxn];
9
10 double Vpai(double r) {
11     return PI * 1.0 * r * r;
12 }
13
14 int main() {
15     ll n, f;
16     double ans;
17     scanf("%lld%lld", &n, &f);
18     f++;//加上自己
19     for (long long i = 0; i < n; ++i) {
```

```
        scanf("%lf", &a[i]);
        a[i] = Vpai(a[i]);
    }
    sort(a,a+n);
    double left = 0, right = a[n-1];
    while (fabs(left - right) > 0.000001) {
        double mid = (left + right) / 2.0;
        ll cnt = 0;
        for (long long i = 0; i < n; ++i) {
            cnt += a[i] / mid;
        }
        if (cnt < f) right = mid;
        else left = mid;
    }
    printf("%.3lf",left);
    return 0;
}
```

*最大化最小值

*最大化平均值

归并排序

- Code

```
void Merge(ll a[],ll b[],ll l, ll m, ll r) {
    ll i = l, j = m + 1, k = l;
    while ((i<=m) && (j<=r))
        if (a[i] <= a[j]) b[k++] = a[i++];
        else {
            b[k++] = a[j++];
            //num += m - i + 1; 逆序对
        }
    if (i>m)
        for (ll q = j; q <= r; q++)   b[k++] = a[q];
    else
        for (ll q = i; q <= m; q++)   b[k++] = a[q];
    for (ll p = l; p <= r; p++)
        a[p] = temp[p];

}
//0, n-1
void MergeSort(ll a[], ll left, ll right) {

    if (left >= right) {
        return;
    }
    else {
        ll i = (left + right) / 2;
        MergeSort(a, left, i);
        MergeSort(a, i+1, right);
        Merge(a, temp, left, i, right);

```

```
29      }
30  }
```

快速排序

原理略，下就编程细节做一些分析

- 关于`partition`函数

```
1  ll partition(ll a[], ll left, ll right) {
2      ll x = left;
3      ll i = left + 1;
4      ll j = right;
5      while (true) {
6          while (i <= j && a[i] <= a[x]) i++;
7          while (a[j] > a[x]) j--;
8          if (i > j) break; //if(i>=j) break;????
9          swap(a[i], a[j]);
10     }
11     swap(a[left], a[j]);
12     return j;
13 }
14
```

有以下几个编程易错点

1. `Line 3`中`i = left + 1`容易错写为`i = left`这样会导致没有选取到旗标元素导致死循环
2. `Line 6`中

```
1  while (i <= j && a[i] <= a[x]) i++;
```

易错写为

```
1  while (i <= j && a[i] < a[x]) i++;
```

或

```
1  while (i < j && a[i] < a[x]) i++;
```

或

```
1  while (a[i] <= a[x]) i++;
```

都可能导致无法正确排序或者死循环

3. `Line 7`中

```
1  while (a[j] > a[x]) j--;
```

易错写为

```
while (a[j] >= a[x]) j--;
```

4. `Line 8`中

```
 if (i > j) break;
```

易错写为

```
 if (i >= j) break;
```

5. `Line 11`中

```
swap(a[left], a[j]);
```

易错写为

```
swap(a[i], a[j]);
```

- `qSort`函数

```
void qSort(ll a[], ll left, ll right) {
    if (left >= right) return;
    ll mid = partition(a, left, right);
    qSort(a, left, mid - 1);
    qSort(a, mid + 1, right);
}
```

- `find`函数

```
ll find(ll a[], ll left, ll right, ll k) {
    ll pos = partition(a, left, right);
    if (pos == k - 1) return a[k - 1];
    if (pos > k - 1) find(a, left, pos - 1, k);
    else find(a, pos + 1, right, k);
}
```

- Code

```
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
const int maxn = 6000000;

ll partition(ll a[], ll left, ll right) {
    ll x = left;
    ll i = left + 1;
    ll j = right;
    while (true) {
        while (i <= j && a[i] <= a[x]) i++;
        while (i <= j && a[j] > a[x]) j--;
        if (i > j) break; //
```

```
        swap(a[i], a[j]);
    }
    swap(a[left], a[j]);
    return j;
}

void qSort(ll a[], ll left, ll right) {
    if (left >= right) return;
    ll mid = partition(a, left, right);
    qSort(a, left, mid - 1);
    qSort(a, mid + 1, right);
}

ll find(ll a[], ll left, ll right, ll k) {
    ll pos = partition(a, left, right);
    if (pos == k - 1) return a[k - 1];
    if (pos > k - 1) find(a, left, pos - 1, k);
    else find(a, pos + 1, right, k);
}

ll a[maxn];

int main() {
    ll n, m, t;
    scanf("%lld", &t);
    while (t--) {
        scanf("%lld%lld", &n, &m);
        for (ll i = 0; i < n; i++) scanf("%lld", &a[i]);
        printf("%lld\n", find(a, 0, n - 1, m));
    }

    return 0;
}
```

PS：对n个数快速排序

```
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
const int maxn = 6000000;

ll partition(ll a[], ll left, ll right) {
    ll x = left;
    ll i = left + 1;
    ll j = right;
    while (true) {
        while (i <= j && a[i] <= a[x]) i++;
        while (a[j] > a[x]) j--;
        if (i > j) break;
        swap(a[i], a[j]);
    }
    swap(a[left], a[j]);
    return j;
}

void qSort(ll a[], ll left, ll right) {
```

```cpp
    if (left >= right) return;
    ll mid = partition(a, left, right);
    qSort(a, left, mid - 1);
    qSort(a, mid + 1, right);
}

ll a[maxn];
int main() {
    ll n;
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n;
    for (long long i = 0; i < n; ++i) {
        cin >> a[i];
    }
    qSort(a, 0, n - 1);
    for (long long i = 0; i < n; ++i) {
        if (i == n - 1) {
            cout << a[i];
            break;
        }
        cout << a[i] << " ";
    }
    return 0;
}
```