## Lesson1

> 排序
>
> - 快速排序
> - 
> ```
> int n, q[N];
>
> inline void quick_sort(int *q, int l, int r) {
>     if (l >= r) return;
>
>     int x = q[l+r>>1], i = l - 1, j = r + 1;
>     while (i < j) {
>         do i++; while (q[i] < x);
>         do j--; while (q[j] > x);
>         if (i < j) swap(q[i], q[j]);
>     }
>
>     quick_sort(q, l, j), quick_sort(q, j+1, r);
> }
> ```
>
> - 归并排序
> - 
> ```
> int n, q[N], tmp[N];
>
> inline void merge_sort(int *q, int l, int r) {
>     if (l >= r) return;
>
>     int mid = l + r >> 1;
>     merge_sort(q, l, mid), merge_sort(q, mid+1, r);
>
>     int k = 0, i = l, j = mid+1;
>     while (i <= mid && j <= r) {
>         if (q[i] <= q[j]) tmp[k++] = q[i++];
>         else tmp[k++] = q[j++];
>     }
>
>     while (i <= mid) tmp[k++] = q[i++];
>     while (j <= r) tmp[k++] = q[j++];
>
>     for (int i=l, j=0; i<=r; i++, j++) q[i] = tmp[k];
> }
> ```

> 二分
>
> - 有单调性一定可以二分，可二分不一定需要有单调性
> - 二分左区间中答案

```
while (l < r) {
    mid = l + r + 1  >> 1;
    if (check(mid))
    true: ans in [mid, r], 更新方式 l = mid
    false: ans in [l, mid-1], 更新方式 r = mid - 1;
}
```

- 为何 +1 ?

  防止死循环

  例如：l = r - 1，check为true时发生死循环

- 二分右区间中答案

```
while (l < r) {
    mid = l + r >> 1;
    if (check(mid))
    true: ans in [l, mid], 更新方式 r = mid
    false: ans in [mid+1, r], 更新方式 l = mid + 1;
}
```

- eg
- ```
  #include <iostream>

  using namespace std;

  const int N = 1e5+10;

  int n, m;
  int q[N];

  int main(void) {
      scanf("%d%d", &n, &m);

      for (int i=0; i<n; i++) scanf("%d", &q[i]);

      while (m--) {
          int x; scanf("%d", &x);

          int l = 0, r = n - 1;
          while (l < r) {
              int mid = l + r >> 1;
              if (q[mid] >= x) r = mid;
              else l = mid + 1;
          }

          if (q[l] != x) cout << "-1 -1" << endl;
          else {
              cout << l << " ";

              l = 0, r = n - 1;
              while (l < r) {
                  int mid = l + r + 1 >> 1;
  ```
  ```

```
                if (q[mid] <= x) l = mid;
                else r = mid - 1;
            }

            cout << l << endl;
        }
    }

    return 0;
}
```

- 浮点数二分

```
#求数的三次方根
#include <iostream>

using namespace std;

int main(void) {
    double x;
    scanf("%lf", &x);

    double l = -100, r = 100;

    while (r - l > 1e-8) {
        double mid = (l + r) / 2;
        if (mid * mid * mid >= x) r = mid;
        else l = mid;
    }

    printf("%.6lf", l);

    return 0;
}
```