查找与排序

Java Reference

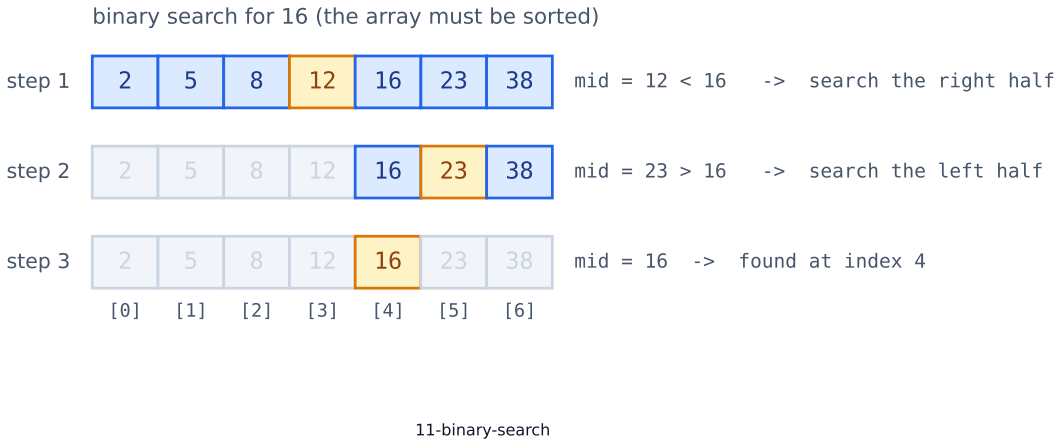
线性查找与二分查找

线性查找 (linear search) 逐个检查每个元素 — 对任何数组都适用。二分查找 (binary search) 快得多, 但需要一个**已排序** (sorted) 的数组: 它查看中间值, 然后每一步丢掉一半。两者都返回索引, 找不到则返回 `-1`。

```
public class Main {
    public static void main(String[] args) {
        int[] a = {2, 5, 8, 12, 16, 23};      // 已排序，所以二分查找可用
        System.out.println(linear(a, 12));    // 3
        System.out.println(binary(a, 12));    // 3
        System.out.println(binary(a, 9));     // -1（未找到）
    }

    static int linear(int[] a, int target) {
        for (int i = 0; i < a.length; i++)
            if (a[i] == target) return i;
        return -1;
    }

    static int binary(int[] a, int target) {
        int lo = 0, hi = a.length - 1;
        while (lo <= hi) {
            int mid = (lo + hi) / 2;
            if (a[mid] == target) return mid;
            else if (a[mid] < target) lo = mid + 1;
            else hi = mid - 1;
        }
        return -1;
    }
}
```



11-binary-search

二分查找每一步把范围减半，已排序数组的查找是 $O(\log n)$

选择排序与插入排序

选择排序 (selection sort) 反复找出剩余值中最小的那个，并把它交换到前面。插入排序 (insertion sort) 取出每个值，把它滑回到已排序部分中属于它的位置。

```java
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        int[] a = {5, 2, 9, 1, 7};
        for (int i = 0; i < a.length - 1; i++) {
            int min = i;
            for (int j = i + 1; j < a.length; j++)
                if (a[j] < a[min]) min = j;
            int t = a[min]; a[min] = a[i]; a[i] = t;   // 交换到位
        }
        System.out.println(Arrays.toString(a));   // [1, 2, 5, 7, 9]
    }
}
```

```java
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        int[] a = {5, 2, 9, 1, 7};
        for (int i = 1; i < a.length; i++) {
            int key = a[i], j = i - 1;
            while (j >= 0 && a[j] > key) {    // 把较大的值向右移
                a[j + 1] = a[j];
                j--;
            }
            a[j + 1] = key;                   // 把 key 放进空位
        }
        System.out.println(Arrays.toString(a));   // [1, 2, 5, 7, 9]
    }
}
```

各算法的速度对比:

算法	时间
线性查找	O(n)
二分查找	O(log n), 仅限已排序数组
选择排序 / 插入排序	$O(n^2)$
归并排序 (见主题 10)	O(n log n)

常见错误

- 二分查找只对**已排序**的数组有效。
- 线性查找是 O(n); 二分查找是 O(log n), 但要先排好序。
- 选择排序和插入排序都是 $O(n^2)$— 好懂, 但大数据上很慢。