查找、排序与递归

C Reference

线性查找与二分查找

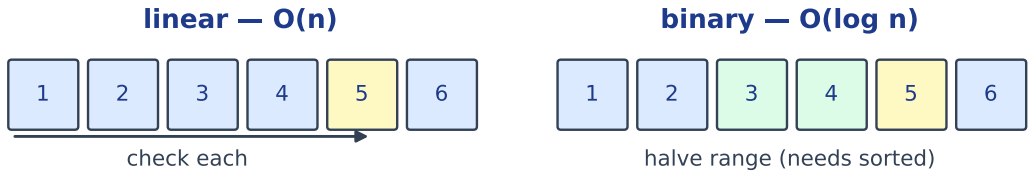
线性查找 (linear search) 依次检查每个元素 — 它对任何数组都适用。二分查找 (binary search) 快得多, 但需要一个已排序 (sorted) 的数组: 它检查中间值, 每一步丢掉一半。两者都返回索引, 找不到则返回 `-1`。

```c
#include <stdio.h>

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

int binary(int a[], int n, int target) {
    int lo = 0, hi = n - 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;
}

int main(void) {
    int a[] = {2, 5, 8, 12, 16, 23};
    int n = sizeof(a) / sizeof(a[0]);
    printf("%d %d %d\n", linear(a, n, 12), binary(a, n, 12), binary(a,
↪   n, 9));
    return 0;   // 3 3 -1
}
```



Exam tip: binary search needs a sorted array; linear works on any order

Linear scans all; binary halves a sorted range

排序

冒泡排序 (bubble sort) 反复比较相邻元素, 并交换 (swap) 任何顺序不对的一对。每一趟之后, 最大的值就 “ 冒泡 ” 到末尾。

```c
#include <stdio.h>

int main(void) {
    int a[] = {5, 2, 9, 1, 7};
    int n = sizeof(a) / sizeof(a[0]);
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - 1 - i; j++) {
            if (a[j] > a[j + 1]) {
                int t = a[j]; a[j] = a[j + 1]; a[j + 1] = t;
            }
        }
    }
    for (int i = 0; i < n; i++) printf("%d ", a[i]);
    printf("\n");   // 1 2 5 7 9
    return 0;
}
```

递归

递归 (recursion) 是一个调用自己的函数。它需要一个基准情形 (base case) 来停止，以及一个朝它靠近的递归调用。没有基准情形它就永不结束。

```c
#include <stdio.h>

int factorial(int n) {
    if (n <= 1) return 1;           // 基准情形
    return n * factorial(n - 1);    // 递归调用
}

int main(void) {
    printf("%d\n", factorial(5));  // 120
    return 0;
}
```

常见错误

- 二分查找需要**已排序**的数组; 递归需要基准情形。
- 每次递归都要更靠近基准情形, 否则栈会溢出。
- 冒泡和插入排序是 $O(n^2)$— 学习可以, 规模大了就慢。