Searching, sorting & recursion

C Reference

Linear & binary search

Linear search 线性查找 checks each element in turn —it works on any array. Binary search 二分查找 is far faster but needs a sorted 已排序 array: it checks the middle and discards half each step. Both return the index, or `-1` if missing.

```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
}
```

Sorting

Bubble sort 冒泡排序 repeatedly compares neighbours and swaps 交换 any pair that is out of order. After each pass the largest value ”bubbles” to the end.

```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;
}
```

Recursion

Recursion 递归 is a function that calls itself. It needs a base case 基准情形 to stop, and a recursive call that steps toward it. Without a base case it never ends.

```
#include <stdio.h>

int factorial(int n) {
    if (n <= 1) return 1;          // base case
    return n * factorial(n - 1);   // recursive call
}

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