C Programming Basic – week 7

Searching (part 2)

Lecturers :

Do Quoc Huy

Dept of Computer Science

Hanoi University of Technology

Binary Search (Tìm kiếm nhị phân)

1	3	5	6	10	11	14	25	26	40	41	78

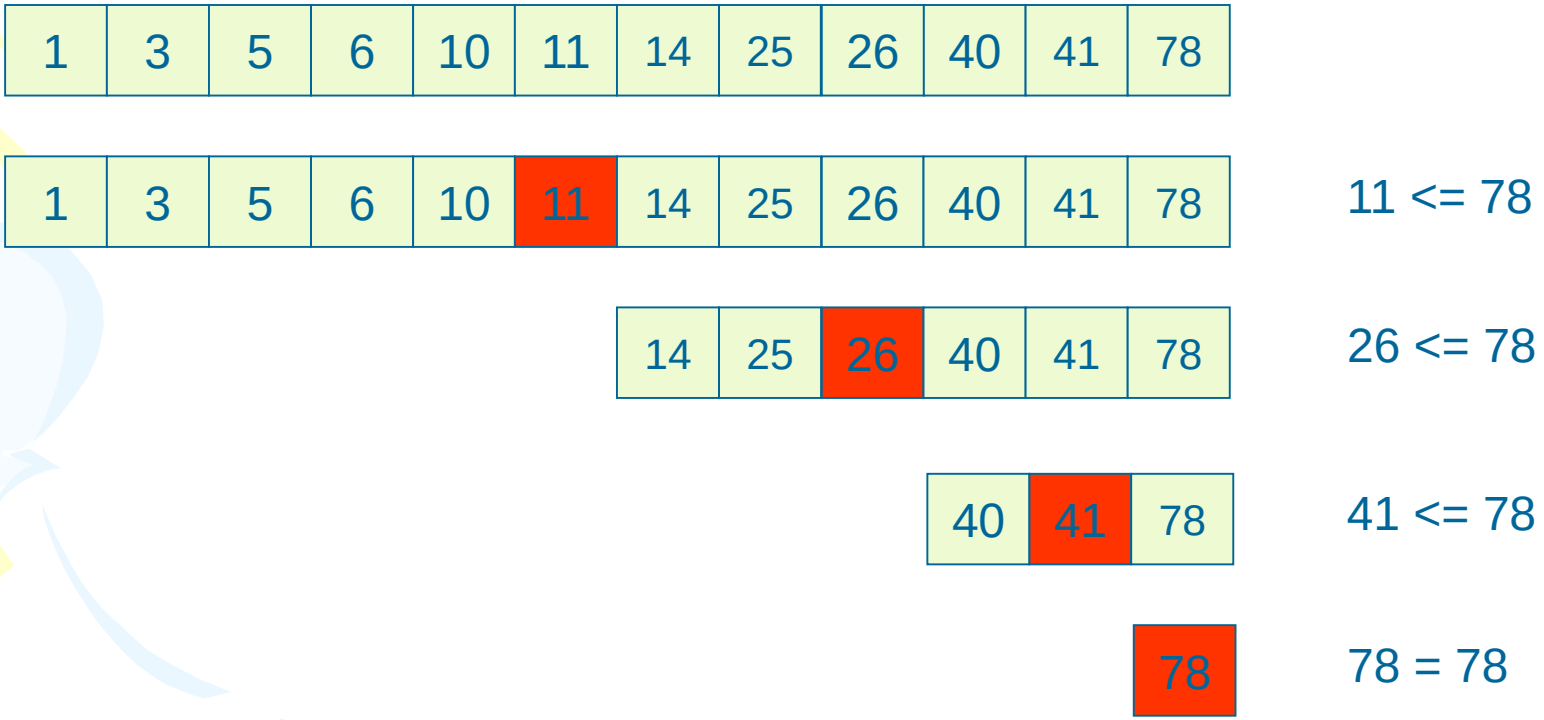
- Thuật toán tìm kiếm nhị phân sử dụng kĩ thuật chia để trị để tìm kiếm.
- Đầu tiên, phần tử tìm kiếm được so sánh với phần tử giữa của list.
- Nếu phần tử tìm kiếm bé hơn phần tử giữa, giới hạn tìm kiệm lại về nửa đầu của list.
- Nếu không, tìm kiếm nửa sau của list.

Binary Search

- Tìm kiếm nhị phân là 1 kỹ thuật mạnh đáng kinh ngạc để tìm kiếm trong 1 list đã được sắp xếp.
- Nó quen thuộc với mọi người sử dụng danh bạ điện thoại!

Minh họa

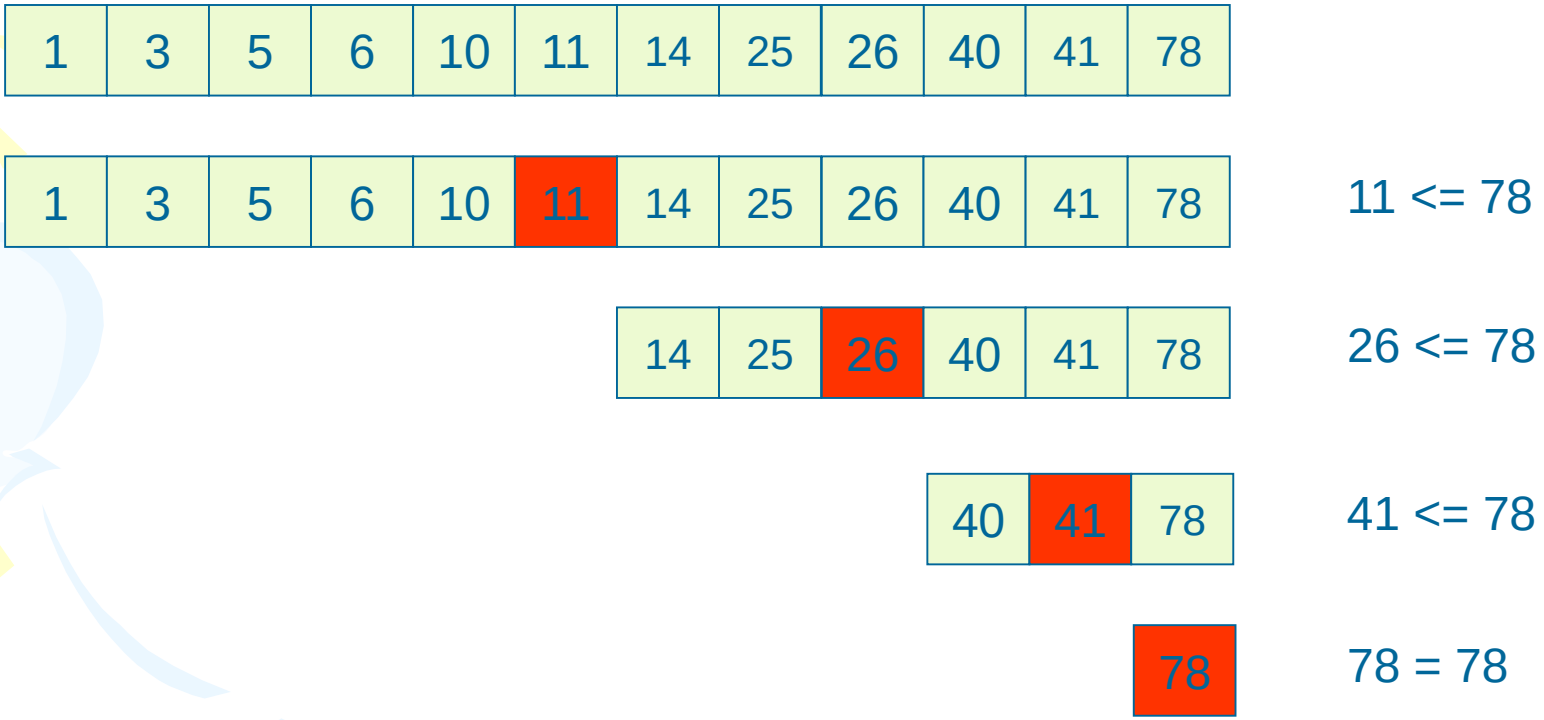
- Tìm kiếm với key=*78*



Đầu tiên so sánh 78 với phần tử giữa của list L[5] là 11.
78 > 11. Vì vậy ta giới hạn lại tìm kiếm L[6……11] như hình minh hoạ.

Minh họa

- Tìm kiếm với key=*78*



4 phép toán cần thiết để tìm ra phần tử phù hợp.
- Thử tính xem phải dùng bao nhiêu phép toán nếu sử dụng tìm kiếm tuần tự?

Binary Search Code

```
int binSearch(int List[], int Target, int Size) {
    // target là số cần tìm, size là kích thước list
    int   Mid,
          Lo = 0,
          Hi = Size – 1;
    while ( Lo <= Hi ) {
          Mid = (Lo + Hi) / 2;
          if ( List[Mid] == Target )
                return Mid;
          else if ( Target < List[Mid] )
                Hi = Mid – 1;
          else
                Lo = Mid + 1;
    }
    return -1;
}
```

Test Program

```
#include <stdio.h>
#define NotFound (-1)
typedef int ElementType;

int  BinarySearch(ElementType A[ ], ElementType X, int N )  {
      int Low, Mid, High;
     Low = 0; High = N - 1;
     while( Low <= High ) {
            Mid = ( Low + High ) / 2;
            if( A[ Mid ] < X )
                        Low = Mid + 1;
            elseif( A[ Mid ] > X )
                        High = Mid - 1;
            else
                        return Mid;  /* Found */
     }
     return NotFound;     /* NotFound is defined as -1 */
}
main( )
{
   static int A[ ] = { 1, 3, 5, 7, 9, 13, 15 };
   int SizeofA = sizeof( A ) / sizeof( A[ 0 ] );
   int i;
   for( i = 0; i < 20; i++ )
      printf( "BinarySearch of %d returns %d\n",
            i, BinarySearch( A, i, SizeofA ) );
   return 0;
}
```

Exercise 1: Đệ quy tìm kiếm nhị phân

- Khai triển 1 phiên bản đệ quy cho hàm tìm kiếm nhị phân.

Ký hiệu chữ O lớn

- Định nghĩa: Giả sử rằng *f(n)* và *g(n)* là các hàm không âm của n. Ta nói *f(n)* là O(g(n)) nếu tồn tại các hằng số *C>0* và *N>0* để với mọi *n>N* thì *f(n)* ≤ *Cg(n)*.

- Điều đó nói lên rằng hàm f(n) tăng với 1 tốc độ tỉ lệ không lớn hơn hàm g(n),tức g(n) là cận trên của f(n).

- Kí hiệu O thể hiện giá trị cận trên của tỉ lệ tăng của 1 hàm với giá trị lớn thích đáng của n.

Phân tích thời gian tính của thuật toán

- Ước lượng số lượng phép so sánh.
- So sánh kết quả với kích thước của dữ liệu vào.
- Tìm kiếm tuần tự : O(n)
- Tìm kiếm nhị phân: O(log2(n))

Exercise 2

- Định nghĩa 1 mảng số nguyên và nhập từ 1 đến 100 theo thứ tự vào mảng.
- Đọc 1 số từ 1 đầu vào chuẩn. Tìm kiếm nhị phân trên mảng.In ra “not found” nếu mảng không chứa số đó.
- Khi thực hiện tìm kiếm nhị phân, đưa ra chỉ số mảng được so sánh với đầu ra chuẩn, và hiển thị số các phép so sánh đến khi dữ liệu được tìm ra.

Gợi ý

- Với mỗi phéo so sánh:

- Tăng biến đếm toàn cục.

Execise 3

- Sử dụng hàm đệ quy cho thuật toán tìm kiếm nhị phân.
- In ra số lần gọi hàm Binary Search đến khi mà dữ liệu được tìm thấy.
- So sánh nó với bản không đệ quy.

Thứ tự từ điển và tìm kiếm nhị phân

- Khi tìm kiếm 1 xâu giá trị thì sự so sánh giữa 2 giá trị là dựa trên thứ tự từ điển.
- Ta có:
  - 'a' < 'd', 'B' < 'M'
  - "acerbook" < "addition"
  - "Chu Trong Hien" > "Bui Minh Hai"
- Chỉ cần sử dụng hàm strcmp()

Exercise 4

- Tạo 1 danh bạ điện thoại.
- Khai báo cấu trúc gồm ít nhất các trường: tên, sđthoại địa chỉ mail.Khai báo 1 mảng cấu trúc có thể chứa 100 phần tử.
- Đọc mảng dữ liệu khoảng 10 phần tử từ 1 file đầu vào,ghi ra tên trùng với tên được mô tả và chỉ số mảng của ai là nhỏ nhất trong file đầu ra.
- Sử dụng tìm kiếm nhị phân cho bài tập này.

Exercise 5

- Trở về bài tập SortedList (list đã được sắp xếp) trong week 4 (quản lý Student) (Linked list) với cấu trúc của phần tử như sau
-
```
typedef struct Student_t {
    char    id[ID_LENGTH];
    char    name[NAME_LENGTH];
    int     grade;

    struct Student_t *next;
} Student;
```

- Khai triển hàm Binary Search trên list này dựa trên:

- Name
- Điểm (grade)

Của student

Kiểm thử các list

- So sánh các list để kiểm tra rằng chúng đồng nhất với nhau không.
- Ví dụ: employee với employer.
- Độ phức tạp:
  - Thứ tự ngẫu nhiên: O(mn)
  - List được sắp xếp: O(tsort(n)+tsort(m)+m+n)
  - Với tsort(n) là thời gian sắp xếp list có n phần tử.

Kiểm thử các list (tiếp)

- Cho 2 list mà các phần tử của chúng cùng kiểu.Tìm:

- (a) Tất cả các bản ghi tìm thấy ở list 1 mà không tìm thấy ở list 2.
- (b) Tất cả các bản ghi tìm thấy ở list 2 mà không tìm thấy ở list 1.
- (c) Tất cả các bản ghi tìm thấy ở list 1 và list 2 với cùng key (khoá tìm kiếm) nhưng khác ở 1 trường nào đó khác.