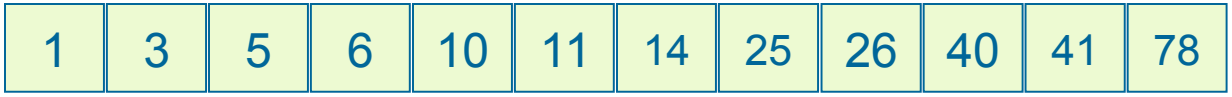
C Programming Basic – week 7

Content

Binary search

Binary Search



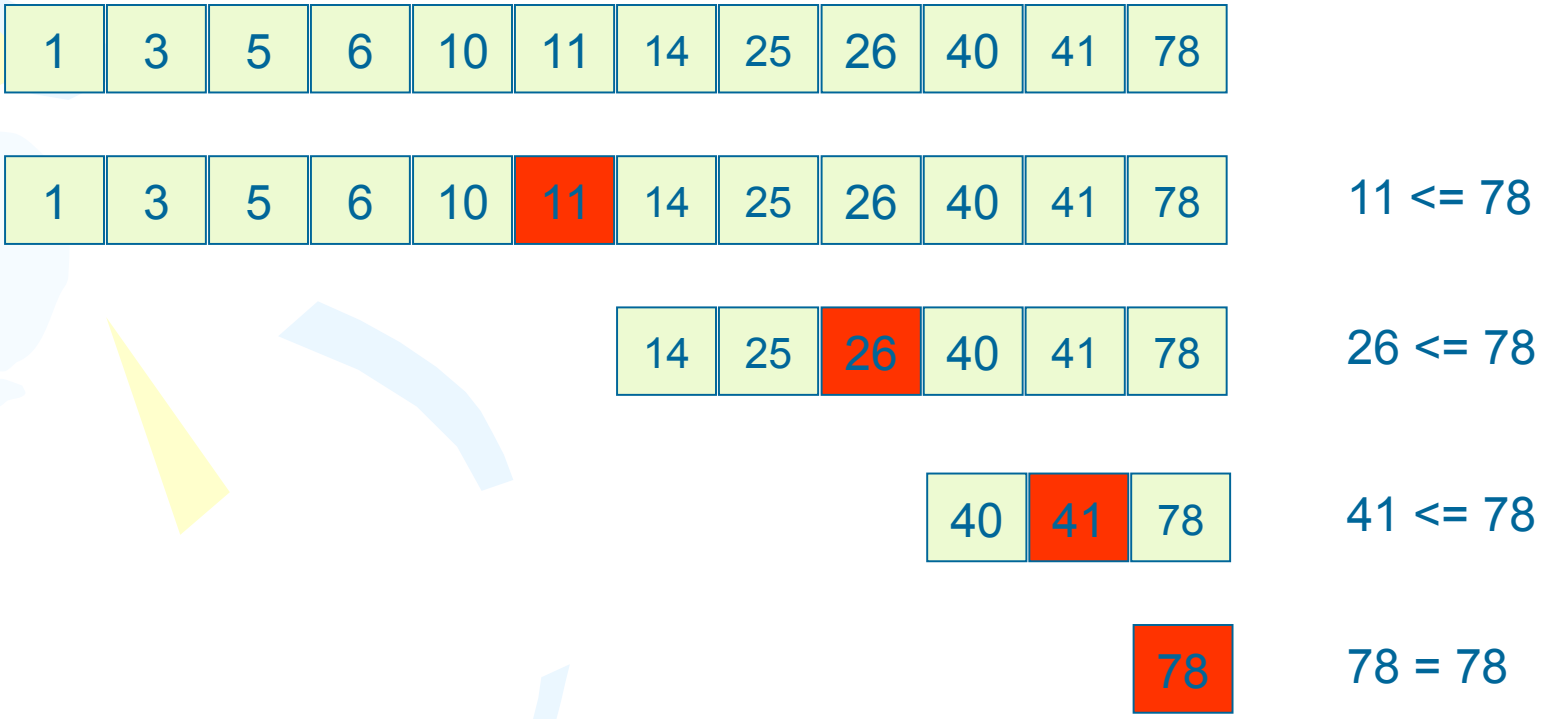
- Divide-and-conquer strategy
- First, the search item is compared with the middle element of the list.
- If the search item is less than the middle element of the list, restrict the search to the first half of the list.
- Otherwise, search the second half of the list.

Binary Search

- Prerequisit : List is ordered
- Familiar in searching dictionary or yellow pages

Example

- Searching for a key=*78*



Need four operations to find out the element.

How many operations in case of sequential search?

Algorithm

```
int binSearch(int List[], int Target, int Size) {
    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;
}
```

Example

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

int 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 7.1

- Implement a recursive version of a binary search function.

Big O Notation

- Definition: Suppose that $f(n)$ and $g(n)$ are nonnegative functions, $f(n)$ is $O(g(n))$ if there exists constants $C > 0$ and $N > 0$ such that for all $n > N$, $f(n) \leq Cg(n)$.

- *f(n)* grows at a rate no faster than *g(n);* thus *g(n)* is an upper bound on *f(n).*

- Big-O expresses an upper bound on the growth rate of a function, for sufficiently large values of n.

Complexity of search algorithm

- Measure the number of comparison operations
- Compare results with the problem's size (size of input data)
- Sequential Search: $O(n)$
- Binary Search: $O(\log_2 n)$

Exercise 7.2

- Define an array of integers, load from 1 to 100 in order to the array.
- Read a number from the standard input, perform the binary search in the array. Output "Not Found" if the array does not contain the number.
- Output the array index at each searching step. Finally, display the number of comparisons when the target number is found.

Hint

- With each comparison:
  - increase a global variable counter

Execise 7.3

- Compare running time of recursive and non-recursive versions of binary search

Dictionary Order

- The comparison of two strings is based on dictionnary order.
- Dictionary order:
  - 'a' < 'd', 'B' < 'M'
  - "acerbook" < "addition"
  - "Chu Trong Hien" > "Bui Minh Hai"
- Use *strcmp*() function.

Exercise 7.4

- Mobile phone address book.
- Declare a structure which can store at least name, telephone number and e-mail address.
- Declare an array of structures that can handle 100 addresses.
- Read 10 addresses from an input file (sorted by name in alphabetic order) to the array
- Ask user to enter a name, print out the index of the first item that matches this name; print out “Not found” otherwise