Arrays- IV

CS10001: Programming & Data Structures

Sudeshna Sarkar

Dept. of Computer Sc. & Engg.,
Indian Institute of Technology Kharagpur

Searching an Array:
Linear and Binary Search

Searching

- Check if a given element (key) occurs in the array.

- If the array is unsorted :
  - start at the beginning of the array
  - inspect every element to see if it matches the key

Linear Search

```
/* If key appears in a[0..size-1], return its location, pos, s.t.
    a[pos] == key. If key is not found, return -1 */

int search (int a[], int size, int key)  {
        int pos = 0;
        while (pos < size && a[pos] != key)
                pos++;
        if (pos<n)
                return pos;
        return -1;
}
```

Linear Search

int x= {12,-3, 78,67,6,50,19,10} ;

Trace the following calls :

search (x, 8,6) ;

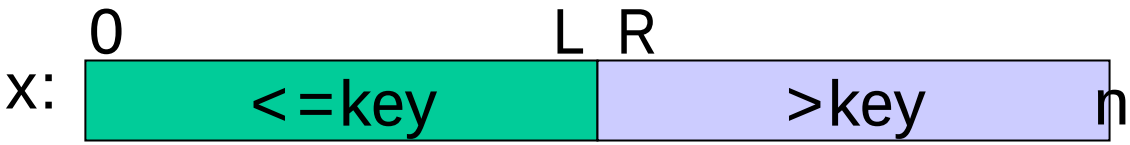
search (x,8,5) ;

Searching a sorted array

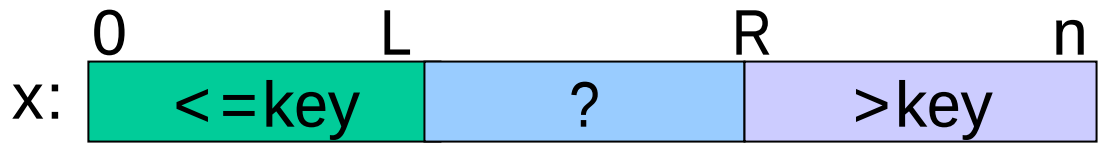
- **Binary search works if the array is sorted**
  - **Look for the target in the middle**
  - **If you don’t find it, you can ignore half of the array, and repeat the process with the other half.**

Binary Search Strategy

- **What we want : Find split betwen values larger and smaller than x :**



•Situation while searching :



•Step : Look at [(L+R)/2]. Move L or R to the middle depending on test.

Binary Search

```
/* If key appears in x[0..size-1], return its location, pos s.t.
     x[pos]==key. If not found, return -1 */
int binsearch (int x[], int size, int key)        {
     int L, R, mid;
     _____________________________;
     while (_____________________)      {




     }
     _____________________________ ;
}
```

Binary Search

```
/* If key appears in x[0..size-1], return its location, pos s.t.
    x[pos]==key. If not found, return -1 */
int binsearch (int x[], int size, int key) {
    int L, R, mid;
    _____________________________;
    while (_____________________)){
        mid = (L+R)/2;
        if (x[mid] <= key)
            L = mid;
        else R = mid;
    }
    _____________________ ;
}
```

Binary Search: loop termination

```
/* If key appears in x[0..size-1], return its location, pos s.t.
    x[pos]==key. If not found, return -1 */
int binsearch (int x[], int size, int key)  {
    int L, R, mid;
    _____________________________;
    while (  L+1 != R  ){
        mid = (L+R)/2;
        if (x[mid] <= key)
            L = mid;
        else R = mid;
    }
    _____________________ ;
}
```

Binary Search: Return result

```
/* If key appears in x[0..size-1], return its location, pos s.t.
    x[pos]==key. If not found, return -1 */
int binsearch (int x[], int size, int key)  {
    int L, R, mid;
    _____________________________;
    while ( L+1 != R)      {
        mid = (L+R)/2;
        if (x[mid] <= key)
            L = mid;
        else R = mid;
    }
    if (L>=0 && x[L]==key) return L;
    else return -1;
}
```

Binary Search: Initialization

```
/* If key appears in x[0..size-1], return its location, pos s.t.
    x[pos]==key. If not found, return -1 */
int binsearch (int x[], int size, int key)  {
    int L, R, mid;
    L=-1; R=size;
    while ( L+1 != R)      {
        mid = (L+R)/2;
        if (x[mid] <= key)
            L = mid;
        else R = mid;
    }
    if (L>=0 && x[L]==key) return L;
    else return -1;
}
```

Binary Search Examples

-17	-5	3	6	12	21	45	63	50

Trace :

binsearch (x, 9, 3);

binsearch (x, 9, 145);

binsearch (x, 9, 45);

Is it worth the trouble ?

- Suppose you had 1000 elements
- Ordinary search (if key is a member of x) would require 500 comparisons on average.
- Binary search
  - after 1st compare, left with 500 elements
  - after 2nd compare, left with 250 elements
  - After at most 10 steps, you are done.
- What if you had 1 million elements ?

String

- A string of characters is a sequence of data of type char stored in consecutive memory lcoations and terminated by the null character ‘\0’ whose Ascii value is 0/
- The null string (of length 0) is the null character only.

- String constant: “Programming in C”

P	r	o	g	r	a	m	m	i	n	g		i	n		C	\0	

Reading and printing a string

```
char name[40] ;
. . .
scanf (“%s”, name) ;
```

```
int main () {
        char a[MAXLEN], b[MAXLEN], c[MAXLEN] ;
        scanf (“%s”, a) ;
        scanf (“%[^\n]”, b) ;
        scanf (“%[^n]”, c) ;

        printf (“a = %s\n”, a) ;
        printf (“b = %s\n”, b) ;
        printf (“c = %s\n”, c) ;
        return 0;
}
```

Input:

The quick brown fox

jumped over the lazy dog

Output:

a = The

b = quick brown fox

c = jumped over the lazy dog

A program using strings

```
/* Count the number of words in a string */
#include <ctype.h>
int word_cnt  (char s[])  {
      int cnt = 0, i;
      for (i=0; s[i] != '\0';)  {
            while (isspace(s[i]))            /* skip white space */
                  ++i;
            if  (s[i] != '\0')   {           /* found a word      */
                  ++cnt;
                   while (!isspace(s[i]) && s[i]!='\0')   /* skip the word */
                         ++i;
             }
      }
      return cnt;
}
```

The program with pointers

```
/* Count the number of words in a string */
#include <ctype.h>
int word_cnt  (char *s)  {
      int cnt = 0;
      while (*s != '\0')  {
            while (isspace(*s))                  /* skip white space */
                  ++s;
            if  (*s != '\0')   {                 /* found a word      */
                  ++cnt;
                   while (!isspace(*s) && *s!='\0')   /* skip the word */
                         ++s;
             }
      }
      return cnt;
}
```

String-handling functions in the standard library

- #define string.h
- char *strcat (char *s1, const char *s2);
- Takes 2 strings as arguments, concatenates them, and puts the result in s1. Returns s1. Programmer must ensure that s1 points to enough space to hold the result.

```
char *strcat(char *s1, const char *s2)
{
    char *p = s1;
    while (*p)   /* go to the end */
        ++p;
    while (*p++ = *s2++)  /* copy */
        ;
    return s1;
}
```

Dissection of the strcat() function

char *p = s1;

p is being initialized, not *p. The pointer p is initialized to the pointer value s1. Thus p and s1 point to the same memory location.

while (*p) ++p; /* ≡ while (*p != '\0') ++p; */

As long as the value pointed to by p is non-zero, p is incremented, causing it to point at the next character in the string. When p points to \0, the expression *p has the value 0, and control exits the while statement.

while (*p++ = *s2++) ;

At the beginning, p points to the null character at the end of string s1. The characters in s2 get copied one after another.

- **int strcmp (const char *s1, const char *s2);**

Two strings are passed as arguments. An integer is returned that is less than, equal to, or greater than 0, depending on whether s1 is lexicographically less than, equal to, or greater than s2.

```
int strcmp(char *s1, char *s2) {
    for (;*s1!='\0'&&*s2!='\0'; s1++,s2++)
    {
        if (*s1>*s2) return 1;
        if (*s2>*s1) return -1;
     }
     if (*s1 != '\0') return 1;
     if (*s2 != '\0') return -1;
     return 0;
}
```

char *strcpy (char *s1, const char *s2);

The characters is the string s2 are copied into s1 until \0 is moved. Whatever exists in s1 is overwritten. It is assumed that s1 has enough space to hold the result. The pointer s1 is returned.

size_t strlen (const char *s);

A count of the number of characters before \0 is returned.

```
size_t strlen (const char *s)  {
        size_t n;
        for (n=0; *s!='\0'; ++s)
                ++n;
        return n;
}
```

```
char * strcpy (char *s1, char *s2)  {
        char *p = s1;
        while (*p++ = *s2++) ;
        return s1;
}
```

Examples of string handling functions

```
char s1[] = “beautiful big sky country”,
      s2[] = “how now brown cow”;
```

Expression	Value
strlen (s1)	
strlen (s2+8)	
strcmp(s1,s2)	

Statements	What gets printed
printf(“%s”,s1+10);	
strcpy(s1+10,s2+8);	
strcat(s1, “s!”);	
printf(“%s”, s1);	

Examples of string handling functions

```
char s1[] = “beautiful big sky country”,
      s2[] = “how now brown cow”;
```

Expression	Value
strlen (s1)	25
strlen (s2+8)	9
strcmp(s1,s2)	negative integer

Statements	What gets printed
printf(“%s”,s1+10);	big sky country
strcpy(s1+10,s2+8);	
strcat(s1, “s!”);	
printf(“%s”, s1);	beautiful brown cows!