Bubble Sort & Binary Search Algorithm Report

Implement Bubble Sort & Binary Search in C

University of Ulsan IT Convergence

20152262 Hong Geun Ji

The following report covers Bubble Sort and Binary Search using C.

As you might know, Bubble Sort is easy way to sort data. Since the way how this algorithm sorts data resembles the shape of bubble rising to the surface, it is called Bubble Sort. Bubble sort will continuously swap the largest element in the unsorted position to the right until it is in its right position. So this algorithms usually takes Big-O of (n*(n-1))/2.

Binary Search is efficient than Linear Search. Mostly, Linear Search takes Big-O of n. This is because Linear Search will look all the data in turn. However, Binary Search will only just take Big-O of $\log_2 n$ (logarithm of n to the base 2).

In this implement, Bubble Sort is implemented by iterative method. But Binary Search is implemented by recursive method. Both algorithms can be implemented opposite, but using right method can help us to understand code more easily.

Take a look at my pseudo code and C version down below.

pseudo code (Bubble Sort)

```
////////////////////////////////////////////////////////////////////////

bubbleSort(list, length of list)
  for each element i, 0 to length of list - 1
        for every element, 0 to length of list - i - 1
            if (list[j] > list[j+1]) swap

////////////////////////////////////////////////////////////////////////
```

pseudo code (Binary Search)

```
////////////////////////////////////////////////////////////////////////

binSearch(list, search_num, left, right)
  middle = (left + right) / 2
  if (left <= right)
        if (list[middle] < search_num) return binSearch(list, search_num, middle+1, right)
        else if (list[middle] < search_num) return binSearch(list, search_num, middle+1,
            right)
        else (list[middle] < search_num) return binSearch(list, search_num, left,
            middle-1)
  else return -1

////////////////////////////////////////////////////////////////////////
```

C code

```
//////////////////////////////////////////////////////////////////

//
//  Sort-Search.c
//  Buble sort & Binary Search.
//
//  Created by Ji Hong Geun on 25/11/18.
//  Copyright © 2018 Ji Hong Geun. All rights reserved.
//

#include <stdio.h>
#include <stdlib.h>
#define MAX_INT_LEN 25
#define COMPARE_2_CASES(x,y) ((x < y) ? 0 : 1)
#define COMPARE_3_CASES(x,y) ((x<y) ? -1: ((x==y) ? 0 : 1))
#define SWAP(x,y,t) ((t) = (x), (x) = (y), (y) = (t))
#define MALLOC(p, t, s)  if( !(p = (t*)malloc(s))  ){\
                                    fprintf(stderr, "Insufficient memory!");\
                                    exit(EXIT_FAILURE);\
                                    }
#define VALID_LEN(l) ((l < MAX_INT_LEN && l > 0) ? 1 : 0)

int initList(int **lst);                        // malloc and assign number.
void bubbleSort(int *lst, int len);
void searchNum(int *lst, int lst_size);
int binSearch(int *lst, int search_num, int left, int right);

int main()
{
    int *lst;
    int i;
    int lst_size = initList(&lst);      // Init the list and takes length of the list.

    bubbleSort(lst, lst_size);

    printf("\nThis is the sorted list by Bubble Sort.\n");
    for(i=0; i<lst_size; i++) printf("%d ", lst[i]);
    printf("\n\n");

    searchNum(lst, lst_size);

    return 0;
}
```

```
int initList(int **lst)
{
   int i = 0;
   int tmp, len;

   do
   {
      printf("Type the length of list : ");
      scanf("%d", &len);
   } while(!VALID_LEN(len));

   MALLOC(*lst, int, sizeof(int)*len);

   while(i < len)
   {
      printf("Type the lst[%d]'s number : ", i);
      scanf("%d", (*lst)+i);
      i++;
   }

   printf("\nThis is the unsorted list.\n");
   for(i=0; i<len; i++) printf("%d ", *((*lst)+i));
   printf("\n");

   return len;
}

void bubbleSort(int *lst, int len)
{
   int i, j, tmp;

   for(i=len-1; i>0; i--)
      for(j=0; j<i; j++)
         if( COMPARE_2_CASES(lst[j], lst[j+1]) ) SWAP(lst[j], lst[j+1], tmp);

}

void searchNum(int *lst, int lst_size)
{
   int find = 0;

   while(find >= 0)
   {
      printf("If you want to stop, type the negative integer.\n");
      printf("What do you want to find? : ");
      scanf("%d", &find);
```

```
        printf("The location of %d : %d\n\n", find, binSearch(lst, find, 0, lst_size));
    }

}

int binSearch(int *lst, int search_num, int left, int right)
{
    int middle = (left + right) / 2;

    if (left <= right)
        switch(COMPARE_3_CASES(lst[middle], search_num))
        {
            case -1 : return binSearch(lst, search_num, middle+1, right);
            case 0 :  return middle;      // Basis level of this recursion.
            case 1 :  return binSearch(lst, search_num, left, middle-1);
        }

    return -1;

}
```

//

There are four functions for this assignment. initList( ) function takes double pointer which points the pointer of number list. This is because the list that this program will use needs to be work on overall, not just for in initList( ) function. If initList( ) function takes only just normal pointer, it will do the task (allocate memory to the list pointer), but will not work as we want to.

The bubbleSort( ) function is implemented by iterative method. There is no returning value in this function, just swapping the original value of the list. binSearch( ) function returns the position of the target number's location. This task will be done recursively.

The searchNum( ) function will receive target number from user and calls binSearch( ). Since one of my coding rule is to let function do only one task, I made this function. This will help binSearch( ) function can only focus on searching target number.

Result

```
ji BubbleSortBinSearch
$ ls -l
total 488
-rwxr-xr-x@ 1 ji  staff  190031 Nov 25 10:59 Description.pages
-rw-r--r--@ 1 ji  staff    2789 Nov 25 10:54 Sort-Search.c
ji BubbleSortBinSearch
$ gcc -o ./Sort-Search Sort-Search.c
ji BubbleSortBinSearch
$ ls -l
total 512
-rwxr-xr-x@ 1 ji  staff  190031 Nov 25 10:59 Description.pages
-rwxr-xr-x  1 ji  staff    8852 Nov 25 11:00 Sort-Search
-rw-r--r--@ 1 ji  staff    2789 Nov 25 10:54 Sort-Search.c
```

-> 1. Compile the 'Sort-Search.c' using gcc compiler. (The exec file will be in current directory.)

```
ji BubbleSortBinSearch
$ ./Sort-Search
Type the length of list : 4
Type the lst[0]'s number : 4
Type the lst[1]'s number : 1
Type the lst[2]'s number : 3
Type the lst[3]'s number : 2

This is the unsorted list.
4 1 3 2

This is the sorted list by Bubble Sort.
1 2 3 4

If you want to stop, type the negative integer.
What do you want to find? : 1
The location of 1 : 0

If you want to stop, type the negative integer.
What do you want to find? : 2
The location of 2 : 1

If you want to stop, type the negative integer.
What do you want to find? : 3
The location of 3 : 2

If you want to stop, type the negative integer.
What do you want to find? : 4
The location of 4 : 3

If you want to stop, type the negative integer.
What do you want to find? : 5
The location of 5 : -1
```

-> 2. If the length of number list is even. (Pay attention to the last output. If there is no number in the list, you will see the negative number.)

```
$ ./Sort-Search
Type the length of list : 6
Type the lst[0]'s number : 4
Type the lst[1]'s number : 6
Type the lst[2]'s number : 1
Type the lst[3]'s number : 3
Type the lst[4]'s number : 2
Type the lst[5]'s number : 5

This is the unsorted list.
4 6 1 3 2 5

This is the sorted list by Bubble Sort.
1 2 3 4 5 6

If you want to stop, type the negative integer.
What do you want to find? : 1
The location of 1 : 0

If you want to stop, type the negative integer.
What do you want to find? : 2
The location of 2 : 1

If you want to stop, type the negative integer.
What do you want to find? : 3
The location of 3 : 2

If you want to stop, type the negative integer.
What do you want to find? : 4
The location of 4 : 3

If you want to stop, type the negative integer.
What do you want to find? : 5
The location of 5 : 4

If you want to stop, type the negative integer.
What do you want to find? : 6
The location of 6 : 5

If you want to stop, type the negative integer.
What do you want to find? : 7
The location of 7 : -1
```

-> 3. If the length of number list is odd.