



#include <stdio.h>

#define SIZE 10

void swap(int \*xp, int \*yp)

{

    int temp = \*xp;

    \*xp = \*yp;

    \*yp = temp;

}

// A function to implement bubble sort

void bubbleSort(int arr[], int n)

{

   int i, j;

   for (i = 0; i < n-1; i++)

       // Last i elements are already in place

       for (j = 0; j < n-i-1; j++)

           if (arr[j] > arr[j+1])

              swap(&arr[j], &arr[j+1]);

}

int backwardLinearSearch(int \*a, int b, int array[])

{

    for (int i = SIZE - 1; i >= 0; i--)

    {

        if(array[i] == b && &array[i] != a) return i;

        click++;

    }

    return -1;

}

int main()

{

    int k = 10, array[SIZE] = {1, 8, 15, 5, 12, 7, 14, 25, 5};

    for (int i = 0; i < SIZE; i++)

    {

        // using pointer so the program dont use the same element.

        int \*a = &array[i], b = k - \*a, index;

        index = backwardLinearSearch(a, b, array);

        if (index != -1)

        {

            printf("%d + %d = %d\n",\*a, array[index],k);

            time\_complexity();

            return 0;

        }

    }

    printf("no match found!\n");

    return 0;

}

این کدی است که من برای سوال 4 در زبان سی استفاده کرده ام. *با توجه به این کد زمان اجرای برنامه در ارایه سورت شده و نشده تفاوت خوبی خواهد داشت!  
توضيح ساده بخواهم بدهم الگوريتم در حالت كلي وقتي به يك عدد ميرسد ميايد و مكملش نسبت به كا را حساب ميكند و از انتهاي ارايه دنبالش ميگردد..*

*اگر ارايه سورت شده باشد وقتي به دنبال عدد مكمل ميگردد خيلي سريا تر به آن ميرسد چون عدد مكمل حتما از خود عدد بزرگ تر خواهد بود(در غير اين صورت در پيمايش هاي قبلي پيدا ميشد)، و چون ارايه سورت شده وقتي از انتها به ابتدا بياييد خيلي سريع تر پيدايش ميكند.*

*ولي اگر ارايه سورت نشده باشد ممكن است عدد مكملش قبل از خود عدد باشد و رسيدن به ان بيشتر طول ميكشد.*

*با همين مثالي كه در خود كد گذاشته ام اين موضوع به وضوح ديده ميشد.*

*پس ارايه سورت شده پيچيدگي زماني كوتاه تري و اجراي كوتاه تري خواهد داشت..*

*Algorithm rate of growth = O(n^2)*





