#include <stdio.h>

#define SIZE 6

int main()

{

    int A[SIZE] = {2, 4, 3, 5, 6, 1};

    int length = SIZE;

    int max = A[0];

    int min = A[0];

    int i = 1;

    while (i < length)

    {

        if (max < A[i])

        {

            max = A[i];

           // printf("%d", A[i]);

        }

        if (min > A[i])

        {

            min = A[i];

        }

        i++;

    }

    printf("max: %d\nmin: %d", max,min);

    return 0;

}

This algorithms big O is O(n), n is the size of the array.

The algorithm goes trough the entire array once comparing each element with the current max and min numbers. And the number of times it does this is dependent on the size of the array(n).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include <stdio.h>

#define SIZE 6

int main()

{

    int A[SIZE] = {1, 2, 3, 5, 6, 8};

    int length = SIZE;

    int num;

    int number;

    int i = 1;

    printf("please enter a number:");

    scanf("%d",&num);

    while (i < length)

    {

       if (num == A[i])

       {

            number = num;

       }

       if(num !=A[i])

       {

         i++;

       }

    }

    printf("%d",number);

    return 0;

This algorithms bigO is O(n), the size of the array).it goes through the array once to check and compare all numbers, and does this the amount of times the size of the array.

#include <stdio.h>

#include <stdlib.h>

#define SIZE 10

void merge\_sort(int, int, int[]  );

int main()

{

    int numlist[SIZE]= {0};

    printf("please enter %d integers:", SIZE);

    for (int i = 0; i < SIZE; i++)

    {

        scanf("%d",&numlist[i]);

    }//end for

    merge\_sort( 0,SIZE-1,numlist);

    for (int i = 0; i < SIZE; i++)

    {

        printf("%d",numlist[i]);

    }//end for

    return 0;

}//END MAIN

void merge\_sort(int low, int high, int A[])

{

    if (low<high)

    {

        int mid = (low + high)/ 2;

        merge\_sort(low, mid, A);

        merge\_sort(mid +1, high,A);

        int left = low;

        int right = mid +1;

        int tempA[SIZE];

        int ptrtemp = 0;

        while (left <= mid && right <= high)

        {

            if (A[left] <= A[right])

            {

                tempA[ptrtemp++] = A[left++];

            }//end if

            else

            {

                tempA[ptrtemp++] = A[right++];

            }//end else

        }//end while

        while (left <= mid)

        {

            tempA[ptrtemp++] = A[left++];

        }//end while

        while (right <= high)

        {

            tempA[ptrtemp++] = A[right++];

        }//end while

        for (int i = low; i <= high; i++)

        {

            A[i] = tempA[i- low];

        }//end for

    }//END IF

}//END MERGE

Graphical user interface, text, application

Description automatically generated

![Text, letter

Description automatically generated]()

![A picture containing calendar

Description automatically generated]()