ALGORİTMA ANALİZİ

ÖDEV 1 | SORU 1-B

GRUP 1

İrem ATILGAN

17061036

07.11.2020

SORU 1

N elemanlı bir dizide birbirine en yakın değere sahip iki elemanın bulunması isteniyor.

B) Alternatif Yaklaşım : Alternatif Yaklaşım'da ilk olarak mevcut dizi Quick Sort sıralama algoritması ile artacak şekilde sıralanır. Daha sonra sıralanan dizideki tüm elemanlar gezilerek kendilerinden bir önceki sayı ile aralarındaki fark bulunur ve minimum fark ile karşılaştırılır (Linear Search). İki aşamanın karmaşıklıkları ele alınırsa:

Quick Sort

Best Case $: O(N * log_2 N)$
Average Case $: \theta(1.39 * N * log_2 N)$
Worst Case $: \Omega(N^2)$
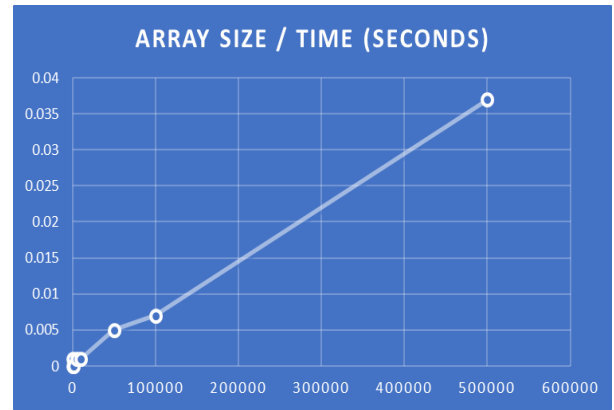
Linear Search

Best Case $: O(N)$
Average Case $: \theta(N)$
Worst Case $: \Omega(N)$

Sonuç :

Best Case $: O(N * log_2 N)$
Average Case $: \theta(1.39 * N * log_2 N)$
Worst Case $: \Omega(N^2)$

Array Size	Time (seconds)
10	0.001
100	0
1000	0
5000	0.001
10000	0.001
50000	0.005
100000	0.007
500000	0.037

SONUÇ

İki yaklaşımın da dizi boyutuna bağlı olarak çalışma süreleri aşağıdaki grafikte karşılaştırmalı olarak verilmiştir. Görüldüğü üzere ikinci yaklaşım, karmaşıklığı Brute-Force yaklaşımına göre çok daha düşük olduğu için, en yüksek başarıyı göstermektedir.
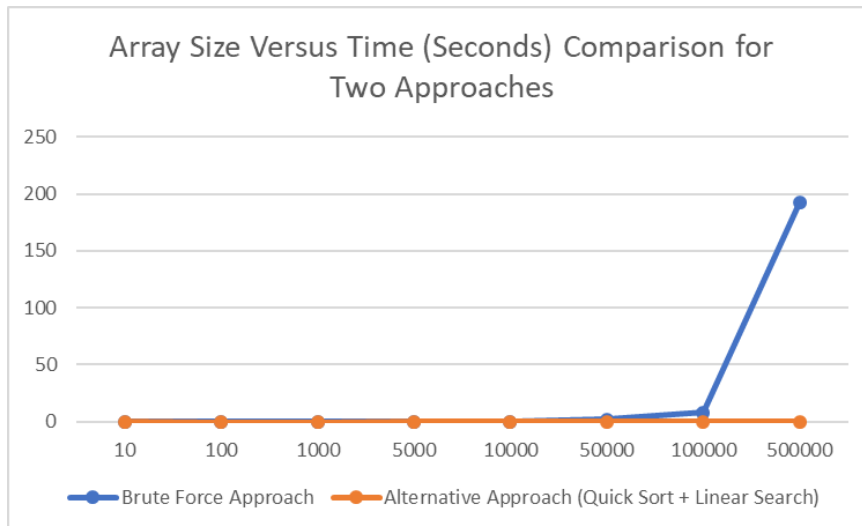
PROGRAM KODLARI

```c
1    #include <stdio.h>
2    #include <stdlib.h>
3    #include <time.h>
4
5    int* set_array(int*, int);           //Set array with random numbers
6    void quick_sort(int*, int, int);     //Quick sort algorithm
7    void swap(int*,int*);                //Swap two numbers
8    int* alt_way(int*, int);             //Alternative way for finding two elements that gives minimum difference in an array
9    int* linear_search(int*, int);       //Search whole array to find minimum difference between sequential numbers
10
11   int main()
12   {
13       int length;      //array length
14       int* array;
15       int* indices;    //indices found for minimum difference
16
17       printf("\nPlease enter the length of the array : ");scanf("%d",&length);
18
19       array = (int*)malloc(sizeof(int)*length);
20
21       if(length > 1)
22       {
23           array = set_array(array,length);
24           indices = alt_way(array,length);
25           printf("MIN DIF = %d FOUND BETWEEN NUMBERS [%d] & [%d]",abs(array[indices[0]]-array[indices[1]]),array[indices[0]],array[indices[1]]);
26           //free pointers from memory
27           free(indices);
28           free(array);
29
30       }
31       else
32           printf("The array size should be more than 1");
33
34       return 0;
35   }
```

```c
37   int* set_array(int* arr, int length) {
38       int i;
39       srand(time(0)); //Use current time as seed for random generator
40       for(i = 0; i < length ; i++)
41           arr[i] = rand();
42
43       return arr;
44   }
45
46   //Alternative way for finding two elements that give minimum difference in an array
47   int* alt_way(int* arr, int length)
48   {
49       clock_t begin = clock();      //the time algorithm had started working
50       quick_sort(arr,0,length-1); //sort the array (in ascending order)
51       printf("SORTED!..\n");
52       int* indices = linear_search(arr,length);   //find indices that give minimum difference by calculating difference of two sequential elements
53       printf("COMPLETED!..\n");
54       clock_t end = clock();                  //the time brute force algorithm has finished
55       double time_spent = (double)(end-begin) / CLOCKS_PER_SEC; //calculate the time passed since the algorithm had started
56       printf("ALGORITHM PROGRESSING TIME = %lf\n",time_spent);
57
58       return indices;
59   }
```

```c
61   //Sort array with Quick Sort
62   void quick_sort(int* arr, int left, int right) {
63       if(left < right)
64       {
65           int i = left;
66           int j = right;
67           int pivot = left;
68           while(i < j)
69           {
70               while(arr[pivot] >= arr[i] && i < right)    i++;
71               while(arr[pivot] < arr[j])  j--;
72               if(i < j) {
73                   swap(&arr[i],&arr[j]); //swap the numbers which one of them is larger and one of them is smaller than pivot
74               }
75
76           }
77           swap(&arr[pivot],&arr[j]);  //set pivot to its new place
78
79           //divide array into two parts and keep sorting
80           quick_sort(arr,left,j-1);
81           quick_sort(arr,j+1,right);
82       }
83   }
```

```
92      //Search whole array to find the indices that give minimum difference between two sequential elements
93    int* linear_search(int* arr, int length) {
94          int i;
95          int* indices = (int*)malloc(sizeof(int)*2); //indices found for minimum difference
96
97          int dif;                                    //difference between two numbers
98          int min_dif;                                //minimum difference found
99
100         //set the minimum difference with the difference of first two elements
101         indices[0] = 0;
102         indices[1] = 1;
103         min_dif = abs(arr[0]-arr[1]);
104
105         for(i = 2; i < length; i++)
106         {
107             dif = abs(arr[i]-arr[i-1]); //calculate the difference between two sequential numbers
108             if(dif < min_dif)
109             {
110                 min_dif = dif;
111                 indices[0] = i;
112                 indices[1] = i-1;
113             }
114         }
115
116         return indices;
117    }
```

EKRAN ÇIKTILARI

```
Please enter the length of the array : 100
SORTED!..
COMPLETED!..
ALGORITHM PROGRESSING TIME = 0.001000
MIN DIF = 2 FOUND BETWEEN NUMBERS [17798] & [17796]
---------------------------------
Process exited after 2.297 seconds with return value 0
Press any key to continue . . .
```

```
Please enter the length of the array : 1000
SORTED!..
COMPLETED!..
ALGORITHM PROGRESSING TIME = 0.000000
MIN DIF = 0 FOUND BETWEEN NUMBERS [880] & [880]
---------------------------------
Process exited after 2.847 seconds with return value 0
Press any key to continue . . .
```

```
Please enter the length of the array : 10000
SORTED!..
COMPLETED!..
ALGORITHM PROGRESSING TIME = 0.001000
MIN DIF = 0 FOUND BETWEEN NUMBERS [30] & [30]
---------------------------------
Process exited after 1.941 seconds with return value 0
Press any key to continue . . .
```

```
Please enter the length of the array : 50
SORTED!..
COMPLETED!..
ALGORITHM PROGRESSING TIME = 0.002000
MIN DIF = 8 FOUND BETWEEN NUMBERS [29351] & [29343]
---------------------------------
Process exited after 3.707 seconds with return value 0
Press any key to continue . . .
```