Mikail Hüseyin AKAR

20060981

27.03.2022

# Sıralama algoritmaları 2 ödev raporu

## Sistem bilgileri:

Dil: C

Ortam: Visual Studio Code

Bilgisayar özellikleri:

OS: Windows 11 Home 21H2

CPU: I7-10750H

RAM: 16(2*8)GB DDR4 3200Mhz

Süre hesaplama fonksiyonu: <time.h> dosyasına ait clock() fonksiyonu

## Kodlar:

```c
void heap(int array[], int a, int i) {
  int max = i;
  int left = 2 * i + 1;
  int right = 2 * i + 2;

  if (left < a && array[left] > array[max])
    max = left;

  if (right < a && array[right] > array[max])
    max = right;

  if (max != i) {
    int temp = array[i];
    array[i] = array[max];
    array[max] = temp;

    heap(array, a, max);
  }
}

void heapSort(int array[], int a) {
  for (int i = a / 2 - 1; i >= 0; i--)
    heap(array, a, i);

  for (int i = a - 1; i >= 0; i--) {
    int temp = array[0];
    array[0] = array[i];
    array[i] = temp;

  heap(array, i, 0);
  }
}
```

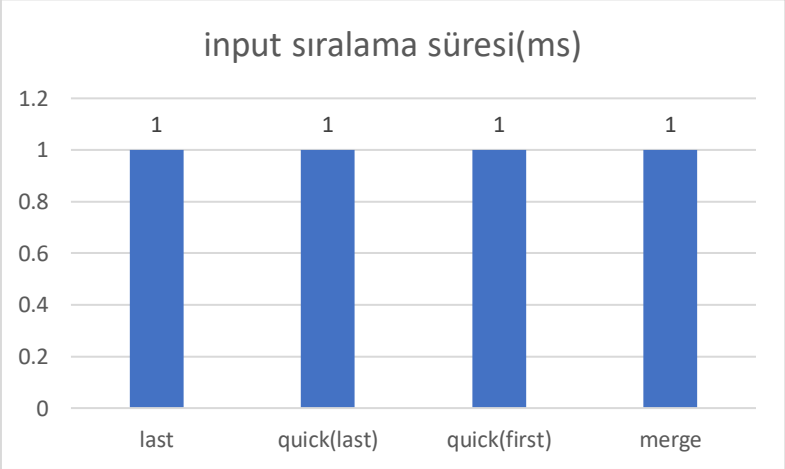
```c
void quickSortlast(int array[], int first, int last){
  if (first < last){
    int pivot = array[last];
    int i = (first - 1);

    for (int j = first; j < last; j++) {
      if (array[j] <= pivot) {
        i++;
        int t = array[i];
        array[i] = array[j];
        array[j] = t;
      }
    }
    int s = array[i + 1];
    array[i + 1] = array[last];
    array[last] = s;

    int a = i + 1;
    quickSort(array, first, a - 1);
    quickSort(array, a + 1, last);
  }
}
```

```
void quickSortfirst(int array[],int first,int last){
  if (first < last){
    int temp;
    int pivot = first;
    int i = first;
    int j = last;
    while (i < j){
      while (array[i] <= array[pivot] && i < last)
        i++;
      while (array[j] > array[pivot])
        j--;
      if (i < j){
        temp = array[i];
        array[i] = array[j];
        array[j] = temp;
      }
  }

  temp=array[pivot];
    array[pivot]=array[j];
    array[j]=temp;
    quickSort(array,first,j-1);
    quickSort(array,j+1,last);
  }
}
```

```
void merge(int array[], int a, int b, int c){
  int n1 = b - a + 1;
  int n2 = c - b;

  int L[n1], M[n2];

  for (int i = 0; i < n1; i++)
    L[i] = array[a + i];
  for (int j = 0; j < n2; j++)
    M[j] = array[b + 1 + j];

  int i, j, k;
  i = 0;
  j = 0;
  k = a;

  while (i < n1 && j < n2){
    if (L[i] <= M[j]) {
      array[k] = L[i];
      i++;
    } else {
      array[k] = M[j];
      j++;
    }
    k++;
  }

  while (i < n1){
    array[k] = L[i];
    i++;
    k++;
  }

  while (j < n2){
    array[k] = M[j];
    j++;
    k++;
  }
}
```

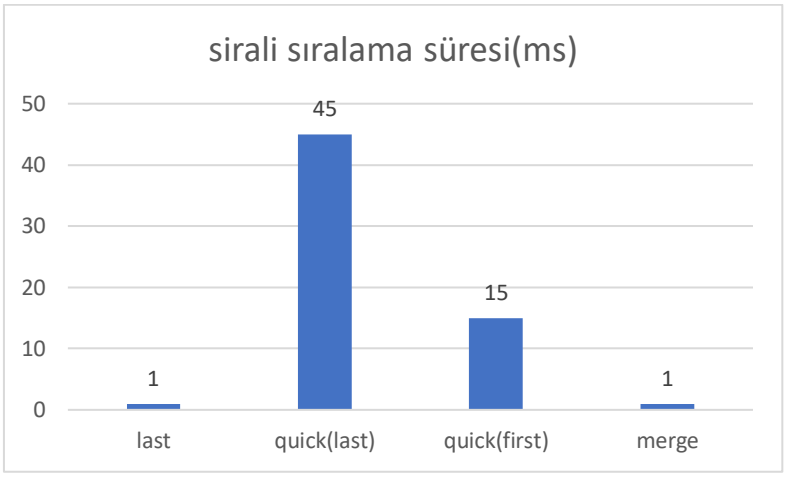
Sonuçlar:

1) input.txt için sıralama süreleri



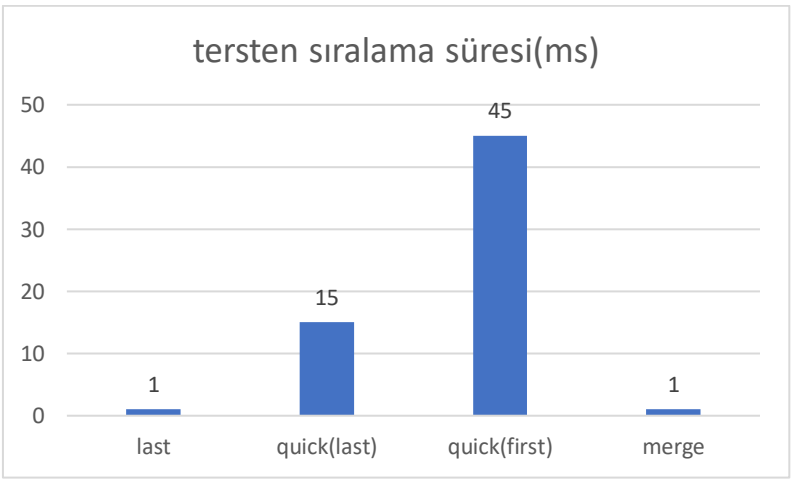

“input” dosyasında yer alan sayıları küçükten büyüğe sıralamak için kodlar tarafından kullanılan sürede fark gözükmüyor.

2) sirali.txt için sıralama süreleri




“sirali” sayılar için quicksort algoritması daha fazla zaman harcıyor. Özellikle pivot son eleman olduğunda ilk eleman olması durumuna göre 3 kat daha yavaş.

3) tersten_sirali.txt için sıralama süreleri




“tersten_sirali” sayılar için yine quicksort algoritması daha fazla zaman harcıyor. Bu sefer pivot ilk eleman olduğunda son eleman olması durumuna göre 3 kat daha yavaş.

Yorum:

Bir sayı dizisini sıralamak için lastsort ve mergesort algoritmaları her durumda aynı sürede ve hızlı çalışıyor.

Quicksort algoritmasındaysa pivot sayısı ortanca sayısına ne kadar yakınsa o kadar hızlı diziyi sıralıyor. Pivot sayısı dizinin en büyük sayısı olma durumunda en küçük sayısı olmasına göre diziyi sıralamak için daha fazla zaman harcanıyor.