Pengurutan/Sorting-2

Indah Agustien Siradjuddin

Quick Sort

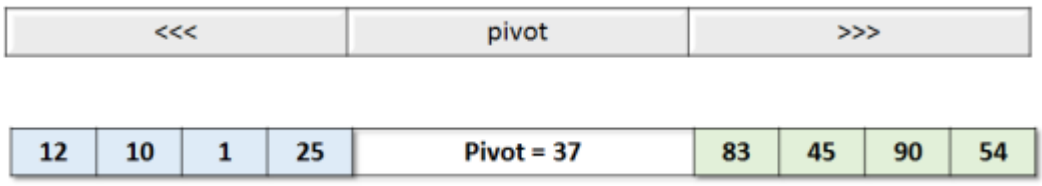
Algoritma *sorting* berikutnya adalah *Quick Sort*. Tahapan pengurutan algoritma ini sama halnya dengan algoritma *Merge Sort*, yaitu proses *divide* dan *combine*. Hanya saja Jika pada algoritma merge sort, proses divide, dilakukan dengan cara membagi data menjadi dua buah bagian, dan proses pengurutan terjadi pada proses *merging*/*combine*. Maka pada algoritma Quick Sort, terdapat perbedaan.

Algoritma ini dibagi menjadi dua tahap, yaitu :

1. *Divide*
2. *Combine*

Divide

Berbeda dengan algoritma *merge sort* yang membagi data menjadi dua bagian, tanpa syarat apapun, maka pada algoritma *quick sort* ini, pembagian data menjadi dua buah bagian, harus memenuhi ketentuan bahwa, data yang berada di bagian sebelah kiri (bagian data pertama), haruslah data-data yang memiliki nilai lebih kecil dari nilai pivot (tidak harus urut). Sedangkan, data-data yang berada disebelah kanan (bagian data kedua), adalah data-data yang memiliki nilai lebih besar dari nilai pivot, seperti yang terlihat pada Gambar 1.



Gambar 1. Pembagian data berdasarkan nilai pivot

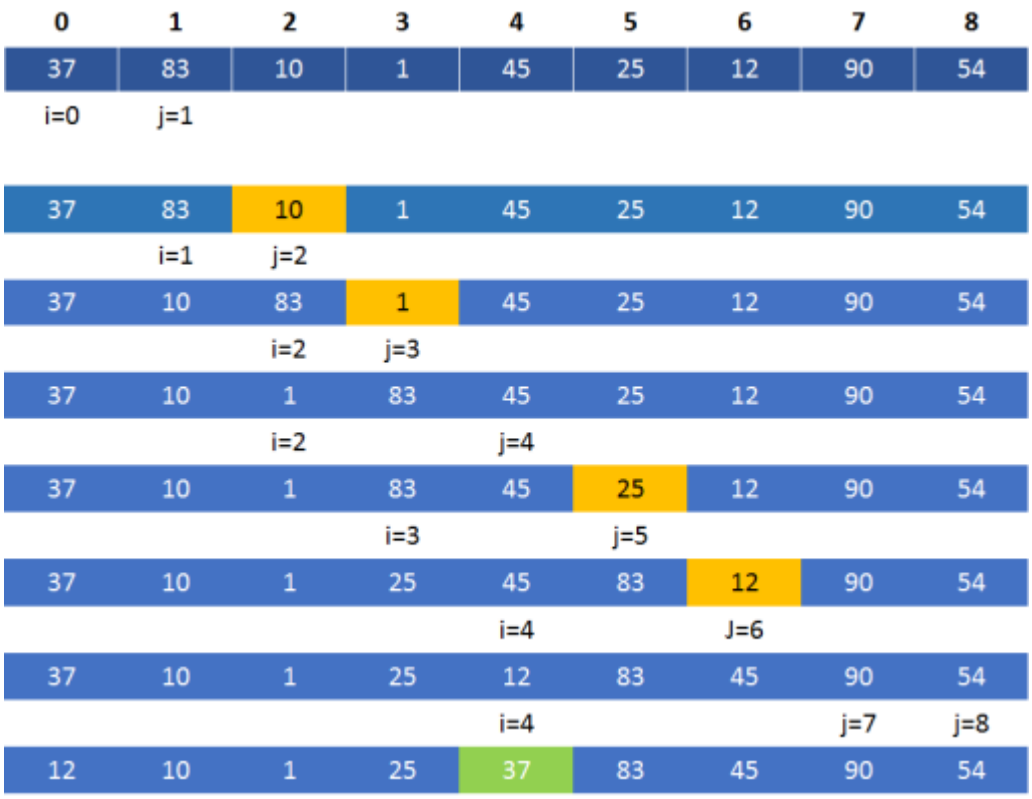
Gambar 1 menunjukkan contoh data yang terbagi berdasarkan nilai pivot. Misalkan nilai pivot=37, maka data yang terdapat pada sebelah kiri pivot, memiliki nilai lebih kecil dari pada pivot, yaitu [12, 10, 1, 25], sedangkan data-data yang berada di sebelah kanan pivot, memiliki nilai lebih besar dari pada pivot, yaitu [83, 45, 90, 54].

Algoritma untuk membagi data atau partisi data menjadi dua bagian berdasarkan nilai pivot tersebut adalah sebagai berikut :

```
partisi(A,p,q) # A=data ; p=indeks awal data yang akan dipartisi, q=inde
ks akhir data yang akan dipartisi
1. x=A[p]     #pivot=x=a[p]
2. i=p
3. for j in range(p+1, q+1) :
4.    if A[j]<=x:
5.        i=i+1
6.        A[i],A[j]=A[j],A[i]
7. A[p],A[i]=A[i],A[p]
```

Pada algoritma tersebut, nilai pivot merupakan data yang terdapat pada indeks awal dari data yang akan dipartisi, misalkan 0. Baris pertama menunjukkan bawa data pivot disimpan dalam variabel x.
Untuk mencari data-data yang bernilai kurang dari pivot, maka dilakukan penelusuran, mulai dari data setelah indeks pivot, sampai dengan data terakhir yang akan dipartisi. Pada proses tersebut, pencarian direpresentasikan dengan variabel j, lihat baris ketiga.
Setelah didapatkan data yang memiliki nilai lebih kecil dari pivot, maka data tersebut ditukar dengan data yang terdapat pada data disebelah pivot, lihat baris kelima dan keenam.
Posisi data terakhir yang memiliki nilai lebih kecil dari data pivot, disimpan dalam variabel i, sehingga pada baris ketujuh, dilakukan pertukaran dengan data tersebut dengan data pivot (lihat baris ketujuh)
.

Ilustrasi partisi ini dapat dilihat pada Gambar 2.



Gambar 2. Ilustrasi Partisi pada Quicksort

Code

Berikut fungsi untuk partisi data, dengan tiga parameter, yaitu A adalah data , p adalah indeks awal dari data yang akan dipartisi, dan q adalah indeks terakhir dari data yang akan dipartisi

In [4]:
```
def partition (A,p,q):
    x=A[p]     #pivot=x=a[p]
    i=p
    for j in range(p+1, q+1) :
        if A[j]<=x:
            i=i+1
            A[i],A[j]=A[j],A[i]
        print(A)
    A[p],A[i]=A[i],A[p]
    return i
```

In [5]:
```
a=[37,83,10,1,45,25,12,90,54]
partition(a,0,len(a)-1)
print(a)
```

```
[37, 83, 10, 1, 45, 25, 12, 90, 54]
[37, 10, 83, 1, 45, 25, 12, 90, 54]
[37, 10, 1, 83, 45, 25, 12, 90, 54]
[37, 10, 1, 83, 45, 25, 12, 90, 54]
[37, 10, 1, 25, 45, 83, 12, 90, 54]
[37, 10, 1, 25, 12, 83, 45, 90, 54]
[37, 10, 1, 25, 12, 83, 45, 90, 54]
[37, 10, 1, 25, 12, 83, 45, 90, 54]
[12, 10, 1, 25, 37, 83, 45, 90, 54]
```

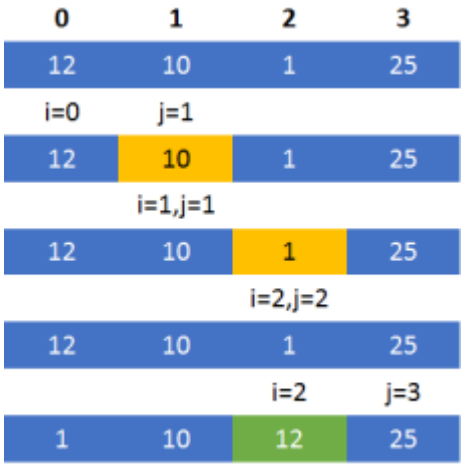
Kembali ke Menu Awal

Combine

Tahap kedua pada algoritma quicsort adalah tahapan combine. Pada tahapan ini, hanya memerintahkan partisi yang berulang kali. Data yang sudah dipartisi, akan terdiri dari dua bagian data, yaitu bagian pertama dan kedua.
Bagian pertama dilakukan partisi lagi, menjadi dua buah bagian, begitu juga bagian kedua, begitu seterusnya, sampai tidak ada data lagi yang akan dipartisi. Sehingga hasil akhir adalah, data akan terurut.

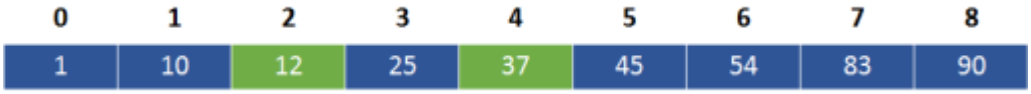
Pada contoh Gambar 2 tersebut, nilai pivot berada pada indeks ke-4. Nilai pivot di akhir proses partisi, menempati indeks yang sebenarnya, ketika data tersebut dalam keadaan terurut. Sehingga proses partisi akan dilakukan pada data-data sebelum nilai pivot, dan data setelah nilai pivot.

Proses *combine* yang dilakukan dengan cara memerintahkan partisi berulang, seperti yang terlihat pada Gambar 3.



Gambar 3. Combine dengan cara partisi

Gambar 3 tersebut menunjukkan partisi kembali data-data sebelum nilai pivot = 37. Proses partisi yang kedua ini menghasilkan nilai pivot baru yaitu 12, dimana data ini berada pada indeks ke-2. Akhir dari kedua partisi yang telah dilakukan, dimana telah dihasilkan nilai pivot 37 dengan indeks ke-4, serta nilai pivot 12 dengan indeks ke-2, menunjukkan bahwa, ketika data terurut, nilai 37 dan nilai 12 memang menempati posisi indeks ke-4 dan ke-2, seperti yang ditunjukkan pada Gambar 4.



Gambar 4. Indeks akhir setelah data terurut

Code

Berikut adalah fungsi combine untuk algoritma quicksort

In [6]:
```
def Quicksort(A,p,q):
    if p<q:
        r=partition(A,p,q)
        Quicksort(A, p, r-1)
        Quicksort(A, r+1, q)
```

In [8]:

```
a=[37,83,10,1,45,25,12,90,54]
Quicksort(a,0,len(a)-1)
```

```
[37, 83, 10, 1, 45, 25, 12, 90, 54]
[37, 10, 83, 1, 45, 25, 12, 90, 54]
[37, 10, 1, 83, 45, 25, 12, 90, 54]
[37, 10, 1, 83, 45, 25, 12, 90, 54]
[37, 10, 1, 25, 45, 83, 12, 90, 54]
[37, 10, 1, 25, 12, 83, 45, 90, 54]
[37, 10, 1, 25, 12, 83, 45, 90, 54]
[37, 10, 1, 25, 12, 83, 45, 90, 54]
[12, 10, 1, 25, 37, 83, 45, 90, 54]
[12, 10, 1, 25, 37, 83, 45, 90, 54]
[12, 10, 1, 25, 37, 83, 45, 90, 54]
[1, 10, 12, 25, 37, 83, 45, 90, 54]
[1, 10, 12, 25, 37, 83, 45, 90, 54]
[1, 10, 12, 25, 37, 83, 45, 90, 54]
[1, 10, 12, 25, 37, 83, 45, 54, 90]
[1, 10, 12, 25, 37, 54, 45, 83, 90]
```

Kembali ke Menu Awal