Dijkstra dan Prim

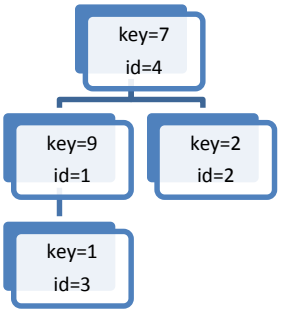
Pada modul ini akan diimplementasikan priority queue yang memiliki handle, dengan menggunakan heap. Fungsi handle adalah menghubungkan elemen-elemen priority queue dengan objek-objek. Karena itu, kita perlu memodifikasi kelas PriorityQueue di minggu sebelumnya. Jika pada minggu sebelumnya heap hanya menyimpan key, kali ini diimplementasikan heap yang tidak hanya menyimpan key, tetapi juga menyimpan id objek.

1. Buatlah kelas Data. Kelas ini mempunyai atribut berupa id bertipe **int** dan key bertipe **int**.
2. Buatlah kelas Priority Queue dengan atribut-atribut sebagai berikut:
   - private Data[] heap
     Atribut ini menyimpan data dalam heap.
   - private int positions[]
     Atribut ini menyimpan lokasi atau posisi suatu id di dalam heap, positions[i] adalah posisi data dengan id=i pada heap. Misalnya data dengan id=1 disimpan di dalam indeks ke-5 pada heap, maka position[1]=5. Contoh lainnya, jika data dengan id=9 di simpan dalam indeks ke-2 pada heap, maka position[9]=2.
   - private int length
     Atribut ini merupakan ukuran maksimal dari heap.
   - public int heapSize
     Atribut ini merupakan jumlah elemen yang sudah ada di dalam heap.
3. Buatlah constructor untuk mengisi atribut-atribut pada kelas PriorityQueue.
```
public PrioQueue(int length){
    this.heapSize=0;
    this.length=length;
    this.heap=new Data[length+1];
    this.positions= new int[length+1];
}
```
4. Tambahkan method getLeft(), getParent(), getRight() seperti pada modul di minggu sebelumnya.
```
private int getLeft(int i){
    return i<<1;
}

private int getRight(int i){
    return (i<<1)|1;
}

private int getParent(int i){
    return i>>1;
}
```
5. Buatlah method **void minHeapify(int i)**.
   Method ini akan membentuk heap di mulai dari indeks ke-i pada heap. Karena kelas PriorityQueue memiliki atribut positions untuk lokasi tiap data dengan id-i pada heap, maka ketika dilakukan *swap* pada dua data di dalam heap, maka isi dari array positions juga berubah.

Perhatikan contoh di bawah ini:



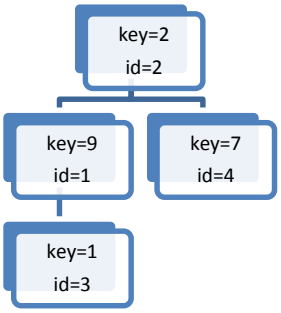
Berdasarkan gambar di atas, isi dari atribut heap adalah sebagai berikut

Indeks	**1**	**2**	**3**	**4**
Key	7	9	2	1
Id	4	1	2	3

Sementara itu, isi dari array locations adalah sebagai berikut:

Indeks	**1**	**2**	**3**	**4**
Positions[indeks]	2	3	4	1

Ketika dilakukan heapify pada indeks ke-1, maka bentuk heap adalah:



Isi dari atribut heap:

Key	**2**	**9**	**7**	**1**
Id	2	1	4	3

Data dengan id=4 bertukar posisi dengan data dengan id=2. Dengan demikian isi dari atribut positions indeks ke-4 akan ditukar dengan isi dari indeks ke-2 (positions[2] ditukar dengan positons[4]).

2	**1**	**4**	**3**

Implementasi:

```
public void minHeapify(int i){
    int left, right, smallest,temp2;
    Data temp;
    left=this.getLeft(i);
    right=this.getRight(i);
    smallest=i;

    if(left<=this.heapSize&&this.heap[left].getKey()<this.heap[smallest].getKey()){
        smallest=left;
    }
    if(right<=this.heapSize&&this.heap[right].getKey()<this.heap[smallest].getKey()){
        smallest=right;
    }
    if(smallest!=i){
        temp=this.heap[smallest];
        this.heap[smallest]=this.heap[i];
        this.heap[i]=temp;


        int a, b;
        a=heap[smallest].getId();
        b=heap[i].getId();

        temp2=positions[a];
        positions[a]=positions[b];
        positions[b]=temp2;
    }
    minHeapify(smallest);
}
```

Tugas:

Tambahkanlah method-method berikut ini pada kelas PriorityQueue:

- minHeapInsert(int *id*, int *key*)
  Method ini mengembalikan true jika data bisa dan berhasil dimasukkan ke heap, dan mengembalikan false jika sebaliknya. Data bisa dimasukkan jika *id* dari suatu data valid (**1<=id<=this.length**). Sebuah *id* juga dikatakan valid jika *id* dari suatu data yang akan dimasukkan ke dalam heap belum ada atau belum tersimpan di heap.
- boolean decreaseKey(int id, int key) yang menurunkan key dari suatu data dengan id=*id*.
  Method ini mengembalikan true jika berhasil menurunkan key, dan false jika sebaliknya.
- Data extractMin() yang akan mengembalikan data dengan key paling kecil yang ada di dalam heap.

Buatlah kelas Tester untuk menguji implementasi yang Anda buat.