Laporan Praktikum Struktur Data

Semester Gasal 2021/2022

NIM	**71200634**
Nama Lengkap	**Egi Granaldi Ginting**
Minggu ke / Materi	**10 / Priority Queue**

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

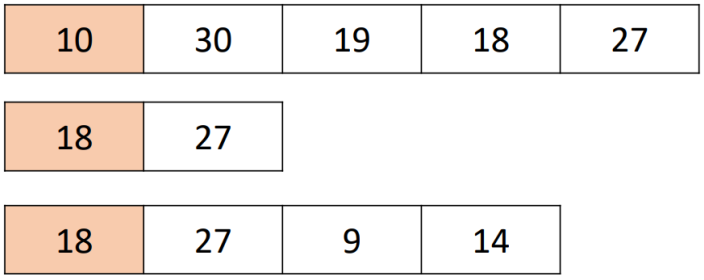
PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2021

Queue

- Struktur data First In first Out (FIFO)

- Data masuk dan keluar sesuai urutan, tidak ada perbedaan prioritas

- Misal:
  - Enqueue 10, 30, 19, 18, 27
  - Dequeue 3x
  - Enqueue 9, 14



Gambar 1 Contoh queue

Variasi dari Queue

- Deque
  - Double ended Queue
  - Masuk dan keluar dari ujung kedua antrian
- Priority Queue
  - Seperti Queue biasa, tetapi mempertahankan prioritas data yang ada di antrian
  - Tidak sepenuhnya berlaku FIFO karena ada faktor prioritas

Priority Queue

Priority Queue adalah queue dengan basis Highest Priority In First Out, berbeda dengan queue yang sangat bergantung kepada waktu kedatangan (elemen yang datang pertama akan selalu diambil atau dihapus pertama pula), sedangkan pada *Priority Queue* elemen yang akan diambil atau dihapus adalah elemen yang memiliki prioritas tertinggi.

- Antrian dengan prioritas yang berbeda – beda
  - Setiap elemen/data memiliki nilai prioritas (semakin kecil, semakin tinggi prioritasnya)
- Memungkinkan elemen/data dengan prioritas tertinggi bisa keluar terlebih dahulu
  - Data pertama keluar bukan berdasarkan posisinya, melainkan berdasarkan tingkat prioritasnya.

Contoh Priority Queue

- Antrian user program vs system/service /kernel program

- Antrian pesawat yang akan mendarat di bandara (berdasarkan rute, jenis

penerbangan, jumlah bahan bakar, dsb)

- Antrian kendaraan di jalan raya (ada beberapa jenis kendaraan yang harus didahulukan)

- Antrian pemesanan ojol (pelanggan platinum memiliki prioritas lebih tinggi)

Operasi – operasi pada Priority Queue

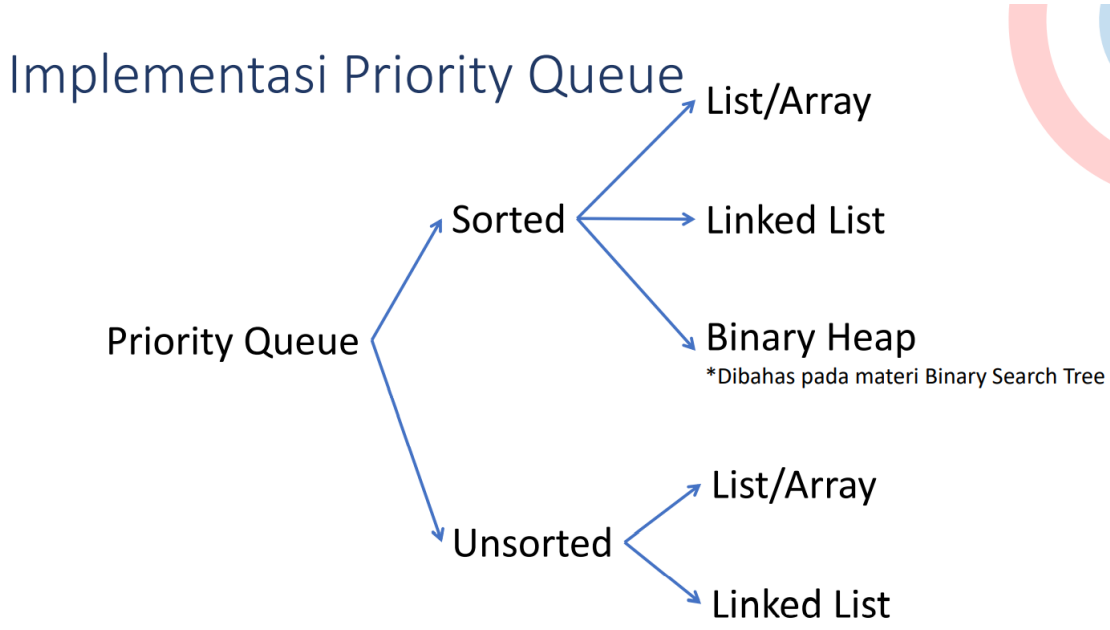
- Insert/Add : Memasukkan data ke dalam Priority Queue
  - Nilai dan tingkat prioritas
- Delete/Remove : Mengeluarkan data dengan tingkat prioritas tertinggi dari antrian
  - Prioritas : semakin kecil, semakin tinggi prioritasnya
  - 1 (prioritas tertinggi),2,3,4,5,6,….
- Peek/Min : Mendapatkan nilai di dalam antrian yang memiliki prioritas tertinggi tanpa mengeluarkannya dari antrian
- Operasi tambahan
  - Tampilkan isi antrian, ubah nilai prioritas salah satu data, kosongkan antrian, remove data tertentu.

Priority Queue dibedakan menjadi dua tipe, yaitu :

- *Ascending Priority Queue*, yaitu Queue yang diurutkan dengan prioritas naik, yaitu dari elemen yang memiliki prioritas terendah hingga prioritas tertinggi.
- *Descending Priority Queue*, yaitu Queue yang diurutkan dengan prioritas menurun, yaitu dari elemen yang memiliki prioritas tertinggi hingga prioritas terendah.

Untuk mempresentasikan Priority Queue dapat dilakukan dengan dua cara, yaitu Set dan List.Dengan Set,data dimasukkan kedalam Queue berdasarkan waktu kedatangan, sedangkan pengambilannya tetap berdasarkan prioritas.Keuntungan dari Set adalah operasi Enqueue sangat cepat dan sederhana, tetapi operasi Dequeue menjadi sangat kompleks karena diperlukan pencarian elemen dengan prioritas tertinggi ,sedangkan dengan List, data di Enqueue dan di Dequeue berdasarkan prioritas.

Implementasi Priority Queue

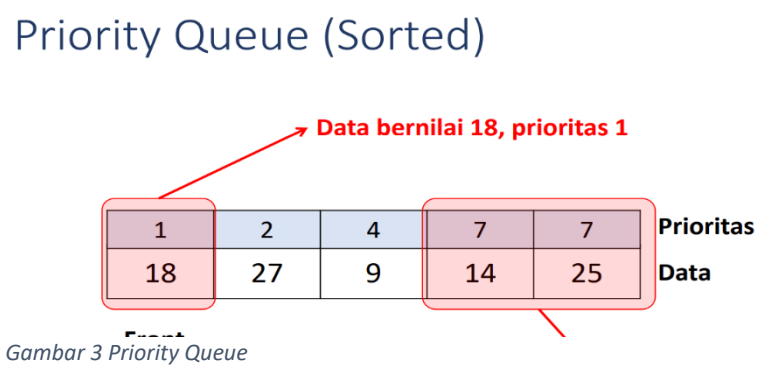


Gambar 2 Implementasi Priority Queue

Sorted vs Unsorted

- Sorted

  - Data pada sorted diletakkan sesuai dengan urutan prioritas
  - Jika ada data baru, maka disisipkan pada posisi sesuai dengan tingkat prioritasnya, proses insert membutuhkan waktu untuk pencarian tempat penyisipan
  - Data paling depan pasti adalah data dengan prioritas tertinggi
  - Data yang keluar adalah data yang paling depan

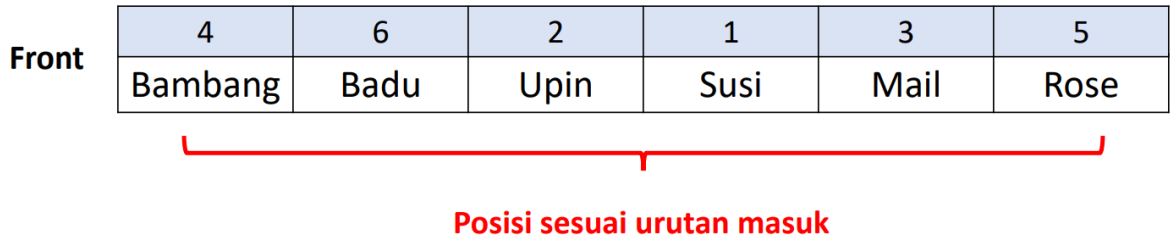


Gambar 3 Priority Queue

- Unsorted
  - Data diletakkan sesuai urutan masuk, tidak diurutkan berdasarkan prioritas
  - Data yang keluar adalah data dengan prioritas tertinggi, harus mencari posisi data dengan pioritas tertinggi terlebih dahulu.

Contoh Priority Queue (Unsorted) – Operasi Add

Masukkan data berikut ini ke dalam priority queue (unsorted)
(Bambang,4) (Badu,6) (Upin, 2) (Susi, 1) (Mail, 3) (Rose, 5)



Gambar 4 Operasi Add Priority Queue(Unsorted)

Sorted vs Unsorted vs Heap

Operation	Sorted	Unsorted	Heap
Add	O(n)	O(1)	O(log n)
Remove	O(1)	O(n)	O(log n)
Peek	O(1)	O(n)	O(1)

Gambar 5 Perbedaan Sorted, Unsorted, dan Heap

Implementasi Priority Queue (Unsorted)

- Berbasis List
  - **Operasi Add** : gunakan fungsi append() dari list
  - **Operasi Remove** : lakukan pencarian index dari data dengan prioritas tertinggi. Lalu gunakan fungsi del atau pop (index) dari list
  - **Operasi Peek** : Lakukan pencarian data dengan prioritas tertinggi
- Berbasis Linked List
  - **Operasi Add** : Insert pada Tail
  - **Operasi Remove** : Lakukan pencarian node dengan prioritas tertinggi. Lalu hapus node tersebut. Kemungkinan : depan/tengah/belakang

- **Operasi Seek** : Lakukan pencarian node dengan prioritas tertinggi

```
class Node:
    def __init__(self, data, priority):
        self._data = data
        self._priority = priority
        self._next = None
```

Gambar 6 Priority Queue dengan Unsorted Linked List

Implementasi Priority Queue (Sorted)

- Berbasis List
  - Operasi Add : Sisipkan pada posisi yang sesuai supaya tetap urut
  - Operasi Remove : hapus data paling depan (prioritas tertingg)
  - Operasi Peek : Ambil nilai dari data paling depan
- Berbasis Linked List
  - Operasi Add : Sisipkan node baru di posisi yang sesuai sehingga tetap urut
  - Operasi Remove : Hapus node paling depan
  - Operasi Peek : Ambil nilai dari node paling depan

```
class Node:
    def __init__(self, data, priority):
        self._data = data
        self._priority = priority
        self._next = None
        self._prev = None
```

Gambar 7 Priority Queue dengan Sorted Linked List

BAGIAN 2: SOAL UNGUIDED

Unguided 9.1

```
class PriorityQueueList:
    def __init__(self):
        self._data = []
        self._cetak = []
    def __len__(self):
        return len(self._data)
    def isEmpty(self):
        return len(self._data) == 0
    def add(self, priority, data):
        self._data.append((priority, data))
    def remove(self):
        if self.isEmpty():
            print("Priority Queue Kosong!")
        min = 1000
        index = 0
        for i in range(len(self._data)):
            if self._data[i][0] < min:
                min = self._data[i][0]
                index = i
        item = self._data[index]
        del self._data[index]
        return item
    def peek(self):
        min = 100
        data = ''
        index = 0
        for i in range(len(self._data)):
            if self._data[i][0] < min:
                min = self._data[i][0]
                data = self._data[i][1]
                index = i
        print(min, data)
    def print(self):
        for i in range(len(self._data)):
            self._cetak.append(self._data[i])
        self._cetak.sort(reverse=True)
        while self._cetak:
            next = self._cetak.pop()
            print(next)
    def removePriority(self, prio):
        if self.isEmpty():
            print("Priority Queue Kosong!")
```

```
        min = 100
        index = []
        for i in range(len(self._data)):
            if self._data[i][0] == prio:
                index.append(self._data[i])
        for j in index:
            self._data.remove(j)

qlist = PriorityQueueList()
qlist.add(4, "Lamborghini")
qlist.add(3, "Ferrari")
qlist.add(1, "Porsche")
qlist.add(2, "GTR")
qlist.print()
print()

qlist.remove()
qlist.print()
print()

qlist.peek()
print()

qlist.add(2, "Innova")
qlist.add(2, "Avanza")
qlist.add(1, "Rubicorn")
qlist.add(1, "Aventador")
qlist.print()
print()

qlist.removePriority(1)
qlist.print()
```

```
PS C:\Users\M S I> & "C:/Users/M S I/AppData/Local/Programs/Python/Python39/python.exe" "d:/Semester 3/Praktikum Struktur Data/Laporan5(1).py"
(1, 'Porsche')
(2, 'GTR')
(3, 'Ferrari')
(4, 'Lamborghini')

(2, 'GTR')
(3, 'Ferrari')
(4, 'Lamborghini')

2 GTR

(1, 'Aventador')
(1, 'Rubicorn')
(2, 'Avanza')
(2, 'GTR')
(2, 'Innova')
(3, 'Ferrari')
(4, 'Lamborghini')

(2, 'Avanza')
(2, 'GTR')
(2, 'Innova')
(3, 'Ferrari')
(4, 'Lamborghini')
PS C:\Users\M S I>
```

Penjelasan :

Pada Unguided 9.1 kali ini kita diminta untuk membuat sebuah PriorityQueueSorted dengan menggunakan list.Setelah itu kita diminta untuk mengimplementasikan fungsi yaitu add,remove,peek,removepriority, dan menampilkan semua data akhir.Fungsi_init_ merupakan konstruktor dari program kali ini.Tugasnya untuk menginisialisasi nilai dari head, tail, dan jumlah data di dalam Priority Queue tersebut.Fungsi isEmpty digunakan untuk mengecek apakah priority queue kosong atau tidak.Fungsi ini juga akan berguna pada saat akan menghapus data.Fungsi _len_ berguna untuk mendapatkan informasi berapa jumlah data yang ada di dalam Priority Queue.Pada implementasi add kali ini kita menambahkan 4 merk mobil dengan prioritasnya masing-masing. Setelah itu kita menggunakan implementasi remove dimana kita menghapus head dari data tersebut yaitu “Porsche” dengan prioritas 1 yaitu prioritas tertinggi.Setelah itu kita mengambil data dengan prioritas tertinggi yaitu GTR dengan prioritas 2.Setelah itu kita menambah data lagi.Setelah itu kita menghapus data dengan priority yang telah diberikan pada source code yaitu 1.Maka data yang memiliki priority 1 akan dihapus.

Unguided 2

```
class Node:
    def __init__(self, data, priority):
        self._data = data
        self._priority = priority # 1 (tertinggi), 2, 3, 4, ...
        self._next = None

class PriorityQueueUnsorted:
    def __init__(self):
        self._head = None
        self._tail = None
        self._size = 0
    def is_empty(self):
        if self._size == 0:
            return True
        else:
            return False
    def __len__(self):
        return self._size
    def print_all(self):
        if self.is_empty() == True:
            print('Priority Queue is empty')
        else:
            bantu = self._head
            while bantu != None:
                print('(', bantu._data, ',', bantu._priority, ')', end=' ')
                bantu = bantu._next
        print()
    def add(self, data, priority):
        baru = Node(data, priority)
        if self.is_empty():
            self._head = baru
            self._tail = baru
        else:
            self._tail._next = baru
            self._tail = baru
        self._size = self._size + 1
    def remove(self):
        if self.is_empty() == False:
            if self._size == 1:
                bantu = self._head
                self._head = None
                self._tail = None
                del bantu
```

```
            else:
                min_priority = self._head._priority

                hapus = self._head
                while hapus != None:
                    if hapus._priority < min_priority:
                        min_priority = hapus._priority
                    hapus = hapus._next

                hapus = self._head
                while hapus._priority != min_priority:
                    hapus = hapus._next

                if hapus == self._head:

                    self._head = self._head._next
                    del hapus
                else:
                    bantu = self._head
                    while bantu._next != hapus:
                        bantu = bantu._next
                    bantu._next = hapus._next
                    del hapus
                    self._tail = self._head
                    while self._tail._next != None:
                        self._tail = self._tail._next
            self._size = self._size - 1
    def peek(self):
        if self.is_empty() == True:
            return None
        else:
            if self._size == 1:
                return tuple([self._head._data, self._head._priority])
            else:
                min_priority = self._head._priority
                bantu = self._head

                while bantu != None:
                    if bantu._priority < min_priority:
                        min_priority = bantu._priority
                    bantu = bantu._next
                bantu = self._head
                while bantu._priority != min_priority:
                    bantu = bantu._next
                return tuple([bantu._data, bantu._priority])
```

```
    def remove_priority(self, priority):
        if self.is_empty() == False:
            if self._size == 1:
                bantu = self._head
                self._head = None
                self._tail = None
                del bantu
            else:
                hapus = self._head
                while hapus._priority != priority:
                    hapus = hapus._next
                bantu = self._head
                while bantu._next != hapus:
                    bantu = bantu._next
                bantu._next = hapus._next
                del hapus
                self._tail = self._head
                while self._tail._next != None:
                    self._tail = self._tail._next
                self._size = self._size - 1

unsorted = PriorityQueueUnsorted()
unsorted.add("Innova", 5)
unsorted.add("Avanza", 1)
unsorted.add("Porsche", 3)
unsorted.add("GTR", 4)
unsorted.print_all()

data, priority = unsorted.peek()
print(data)
print(priority)
print()

unsorted.remove()
unsorted.print_all()
print()

unsorted.add("Ayla", 2)
unsorted.add("Mazda", 1)
unsorted.add("Agya", 3)
unsorted.add("Brio", 2)
unsorted.print_all()
print()

unsorted.remove_priority(2)
```

```
unsorted.print_all()

unsorted.add("Ayla", 2)
unsorted.add("Mazda", 1)
unsorted.add("Agya", 3)
unsorted.add("Brio", 2)
unsorted.print_all()
print()

unsorted.remove_priority(2)
unsorted.print_all()
```

```
PS C:\Users\M S I> & "C:/Users/M S I/AppData/Local/Programs/Python/Python39/python.exe" "d:/Semester 3/Praktikum Struktur Data/laporan 5(2).py"
( Innova , 5 ) ( Avanza , 1 ) ( Porsche , 3 ) ( GTR , 4 )
Avanza
1

( Innova , 5 ) ( Porsche , 3 ) ( GTR , 4 )

( Innova , 5 ) ( Porsche , 3 ) ( GTR , 4 ) ( Ayla , 2 ) ( Mazda , 1 ) ( Agya , 3 ) ( Brio , 2 )

( Innova , 5 ) ( Porsche , 3 ) ( GTR , 4 ) ( Mazda , 1 ) ( Agya , 3 ) ( Brio , 2 )
PS C:\Users\M S I>
```

Penjelasan :

Pada Unguided 2 ini kita diminta untuk menambahkan fungsi remove_priority(self, priority) pada class PriorityQueueUnsroted yang bertujuan untuk menghapus seluruh data yang memiliki tingkat prioritas sesuai dengan nilai priority yang diberikan.Pada PriorityqueueUnsorted kali ini kita mwnambah data terlebih dahulu yaitu (Innova , 5) (Avanza , 1) (Porsche , 3) (GTR , 4). Setelah itu kita mengambil data dengan prioritas tertinggi yaitu 1, dimana prioritas 1 dimiliki oleh Avanza.Untuk proses penghapusan data dilakukan dengan mencari posisi data dengan tingkat prioritas tertinggi, lalu perlu pengecekan data dengan tingkat prioritas tertinggi tersebut posisinya ada di head, atau tail, atau di tengah.Setelah itu kita menghapus data dengan prioritas tertinggi. Setelah itu kita Kembali menambah data (Ayla , 2) (Mazda , 1) (Agya , 3) (Brio , 2). Lalu kita menghapus priority(2) dimana priority 2 yang terlebih dahulu didapatkan adalah Ayla maka data Ayla akan dihapus dan data brio yang berada di tail tidak akan dihapus.

Daftar Pustaka :

http://library.binus.ac.id/eColls/eThesisdoc/Bab2HTML/2009100390MTIFBab2/page11.html