210411100085_Modul4_QueueDeque

April 26, 2022

1 Konsep

1.1 Queue

Pengertian Queue

Queue merupakan struktur data yang penambahan datanya dilakukan pada rear sedangkan penghapusan datanya berada di front. Konsep ini dinamakan FIFO (First In First Out) jadi data yang masuk pertama kali akan keluar/dihapus terlebih dahulu. Visualisasinya seperti ini.

```
"""


|25|26|64|28|54|30| => Sedangkan ini merupakan bagian rear
 ||
 \/
Ini merupakan
bagian front


"""
```

Operasi yang ada pada Queue

createQueue()

createQueue() digunakan untuk membuat data queue yang berupa list.

```
def createQueue() :
    return []
```

enqueue()

enqueue() digunakan untuk menambahkan data pada ujung front queue.

```
def enqueue(q,data):
    q.append(data)
    return(q)
```

enqueue()

enqueue() digunakan untuk menambahkan data pada ujung front queue.

```
def enqueue(q,data):
    q.append(data)
    return(q)
```

dequeue()

dequeue() digunakan untuk menghapus data pada ujung rear queue.

```
def dequeue(q):
    data=q.pop(0)
    return(data)
```

isEmpty()

isEmpty() digunakan untuk mengecek apakah queue dalam keadaan kosong atau tidak.

```
def isEmpty(q):
    return (q==[])
```

size()

size() digunakan untuk mengetahui ukuran dari queue tersebut.

```
def size(q):
    return (len(q))
```

1.2 Deque

Pengertian Deque

Deque merupakan struktur data yang penambahan dan penghapusan data dapat dilakukan pada kedua unjunnya. Kedua ujung pada deque disebut front dan rear, pada front rear menggunakan indeks ke 0 atau indeks awal sedangkan pada rear menggunakan indeks ke terakhir.

```
"""

          Ini merupakan bagian rear pada bagian in dapat dilakukan
          penambahan maupun penghapusan data
                /\
                ||
|25|26|64|28|54|30|
 ||
 \/
Ini merupakan bagian front
pada bagian in dapat dilakukan
penambahan maupun penghapusan data



"""
```

Operasi yang ada pada Deque

createDeque()

createDeque() digunakan untuk membuat deque yang diimplementasikan dengan list.

```
def createDeque() :
    return []
```

addFront()

addFront() digunakan untuk menambahkan data yang berada pada ujung front.

```
def addFront(d,val) :
    d.insert(0,val)
    return d
```

addRear()

addRear() digunakan untuk menambahkan data yang berada pada ujung rear.

```
def addRear(d,val) :
    d.append(val)
    return d
```

removeFront()

removeFront() digunakan untuk menghapus data yang berada pada ujung front.

```
def removeFront(d) :
    val = d.pop(0)
    return val
```

removeRear()

removeRear() digunakan untuk menghapus data yang berada pada ujung rear.

```
def removeRear(d) :
    val = d.pop()
    return val
```

isEmpty()

isEmpty() digunakan untuk mengecek apakah deque dalam keadaan kosong atau tidak.

```
def isEmpty(d) :
    return d==[]
```

size()

size() digunakan untuk mengetahui ukuran dari deque tersebut.

```
def size(d) :
    return len(d)
```

2 Implementasi

2.1 Scheduling / Penjadwalan

[3]:
```
def createQueue() :
    return []

def enqueue(q,data):
    q.insert(0,data)
    return(q)

def dequeue(q):
    data=q.pop()
    return(data)

def isEmpty(q):
    return (q==[])

def size(q):
    return (len(q))
```

[4]:
```
def taskScheduling(timeLimit) :
    taskData  = createQueue()
    taskTotal = int(input("-> Masukkan proses yang akan dijadwal CPU = "))

    for i in range(taskTotal) :
        taskName = input("=> Nama proses ke-%d : " % (i))
        taskTime = int(input("=> Waktu proses :"))
        enqueue(taskData,[taskName,taskTime])

    print("""
Waktu proses CPU = %d
Antrian proses beserta waktunya = %s
    """ % (timeLimit,taskData) )
    numIteration = 1

    while not(isEmpty(taskData)) :
        print("\nIterasi ke-%d : " % (numIteration))
        numIteration+=1
        nowTask   = dequeue(taskData)
        tempLimit = timeLimit

        while nowTask[1] > 0 and tempLimit > 0 :
            nowTask[1]-=1
            tempLimit-=1

        if nowTask[1] > 0 :
```

```
            print("Proses %s sedang diproses dan sisa waktu proses = %d " %␣
↪(nowTask[0],nowTask[1]) )
            enqueue(taskData,nowTask)

        else :
            print("Proses %s telah selesai diproses" % (nowTask[0]) )

        print("Data proses yang tersisa ",taskData)



taskScheduling(3)
```

```
-> Masukkan proses yang akan dijadwal CPU = 3
=> Nama proses ke-0 : A
=> Waktu proses :5
=> Nama proses ke-1 : B
=> Waktu proses :9
=> Nama proses ke-2 : C
=> Waktu proses :2

Waktu proses CPU = 3
Antrian proses beserta waktunya = [['C', 2], ['B', 9], ['A', 5]]


Iterasi ke-1 :
Proses A sedang diproses dan sisa waktu proses = 2
Data proses yang tersisa  [['A', 2], ['C', 2], ['B', 9]]

Iterasi ke-2 :
Proses B sedang diproses dan sisa waktu proses = 6
Data proses yang tersisa  [['B', 6], ['A', 2], ['C', 2]]

Iterasi ke-3 :
Proses C telah selesai diproses
Data proses yang tersisa  [['B', 6], ['A', 2]]

Iterasi ke-4 :
Proses A telah selesai diproses
Data proses yang tersisa  [['B', 6]]

Iterasi ke-5 :
Proses B sedang diproses dan sisa waktu proses = 3
Data proses yang tersisa  [['B', 3]]

Iterasi ke-6 :
```

```
Proses B telah selesai diproses
Data proses yang tersisa  []
```