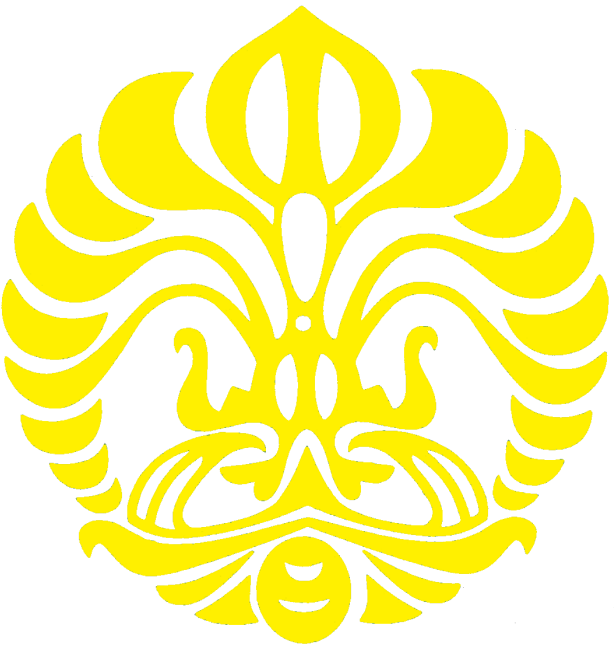
LAPORAN PROYEK

IMPLEMENTASI *QUEUE* : ANTRIAN PADA SEBUAH RESTORAN



Dosen Pengampu :

Dra. Bevina Desjwiandra Handari, M.Si., Ph.D.

Disusun oleh :

1.	Alexandria Samantha Nicole	2006568765
2.	Angelica Patricia Djaya Saputra	2006522000
3.	Latifa Aulia Esmananda	2006486935
4.	Richard Mulyadi	2006568595

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS INDONESIA

2021

KATA PENGANTAR

Puji dan syukur kami panjatkan ke hadirat Allah SWT, yang atas rahmat-Nya dan karunianya kami dalam menyelesaikan laporan ini tepat pada waktunya. Kami juga ingin mengucapkan terima kasih sebanyak-banyaknya kepada Dosen Pengampu Struktur Data Kelas B yaitu Dra. Bevina Desjwiandra Handari, M.Si, Ph.D. atas bimbingannya selama ini. Adapun tema dari laporan ini adalah "Implementasi *Queue* dan Algoritmanya."

Penulis menyadari adanya ketidaksempurnaan dalam penyusunan laporan proyek struktur data ini. Namun, penulis berharap laporan ini dapat memberikan manfaat bagi para pembaca. Demi perkembangan laporan, penulis juga mengharapkan adanya masukan berupa kritik atau saran. Terima kasih.

Depok, 15 Desember 2021

Penulis

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam kehidupan sehari-hari, kita telah sering menemukan antrian, baik itu di sebuah supermarket, loket, dan masih banyak tempat lainnya. Seiring perkembangan zaman, manusia semakin hidup dengan budaya praktis dan cepat sehingga menunggu merupakan salah satu keadaan yang paling dihindari. Dalam meningkatkan keefektifan waktu suatu antrian dengan memanfaatkan teknologi, penulis mempersembahkan algoritma *queue* yang mampu diimplementasikan dalam sebuah restoran.

Antrian dapat direpresentasikan sebagai *Queue* dalam struktur data yang berpedoman pada prinsip FIFO (*First In First Out*). Pada mata kuliah Struktur Data, penulis diharapkan mampu memberikan implementasi dari materi *queue* yang telah dipelajari secara teori di kelas. Oleh karena itu, penulis membuat laporan "Implementasi *Queue* : Antrian pada Sebuah Restoran.". Implementasi *queue* bisa menggunakan *array* atau *linked list*. Pada algoritma kali ini, kami mengimplementasikan *queue* dengan menggunakan *array* yang disebut sebagai *queue* statis.

Misalkan terdapat suatu restoran yang hanya menerima pesanan reservasi melalui aplikasi dengan sistem *pre-order* maksimal satu hari sebelumnya. Pembeli yang ingin melakukan reservasi akan memilih menu yang akan dipesan dan memberi tahu waktu tempuh mereka menuju restoran. Algoritma ini akan mengatur urutan paling efektif berdasarkan waktu tempuh tiap pembeli menuju restoran, di mana makanan baru mulai disiapkan setelah pembeli sampai di restoran agar kualitas makanan tetap terjamin. Dengan disusunnya algoritma tersebut dalam sebuah aplikasi akan membuat semua *customer* restoran tersebut tidak perlu menunggu pesanan terlalu lama.

1.2 Tujuan

Tujuan dari algoritma yang dibahas adalah untuk meningkatkan keefektifan waktu yang dibutuhkan dari saat restoran membuka jam pesanan hingga seluruh pembeli mengambil pesanannya. Selain itu, tujuan dari laporan ini adalah untuk memenuhi tugas akhir dari mata kuliah “Struktur Data”.

BAB II
PEMBAHASAN

2.1 Soal

Berikut permasalahan dari tautan https://www.codechef.com/problems/WIQ.

Anda memiliki n permintaan dalam antrian, dan setiap permintaan memiliki pemilik yang berbeda. Artinya, ada n pemilik, dan masing-masing pemilik memiliki tepat satu permintaan. Permintaan diberi nomor dari 1 hingga n, dan awalnya berurutan dalam antrian. Artinya, Permintaan 1 di kepala (awal) dari antrian, dan Permintaan n di ekor (akhir). Proses dimulai pada $waktu = 1$, yaitu menit pertama. Namun, tidak semua pemilik sudah siap dari awal. Pemilik Permintaan i memiliki b_i yang terkait dengannya, yang merupakan menit dari mana dia akan siap.

Setiap kali Anda memilih permintaan pertama dalam antrian. Misalkan itu adalah Permintaan i. Jika pemiliknya belum siap (yaitu, b_i benar-benar lebih besar dari waktu saat ini), Anda menempatkan permintaan di akhir antrian. Jika tidak, Anda melakukan permintaan, yang membutuhkan waktu beberapa menit. Tetapi proses memeriksa apakah seorang pemilik siap atau tidak membutuhkan waktu satu menit dengan sendirinya. Asumsikan bahwa operasi lain dilakukan secara instan.

Perhatikan bahwa kami memeriksa apakah pemilik sudah siap pada saat itu, meskipun pekerjaan tidak dimulai pada menit ini. Artinya, jika kita memeriksa pada $waktu = 5$, kita tahu bahwa pekerjaan tidak akan dimulai sampai $waktu = 6$, tetapi kita masih memeriksa apakah pemiliknya siap pada $waktu = 5$. Silakan lihat contoh dan penjelasan yang diberikan di bawah ini untuk pemahaman yang lebih baik tentang proses.

Untuk setiap permintaan, keluarkan menit saat permintaan selesai. Artinya, Permintaan i memiliki durasi a_i menit, sehingga harus dijalankan dari $waktu = x$, untuk beberapa x, hingga $waktu = (x + a_i - 1)$. Dan jawabannya adalah $(x + a_i - 1)$.

- Input
  - Baris pertama input berisi satu bilangan bulat T, yang menunjukkan jumlah kasus uji. Setiap kasus uji berisi tiga baris.
  - Baris pertama berisi satu bilangan bulat n, yang menunjukkan jumlah permintaan. Baris kedua berisi n bilangan bulat yang dipisahkan spasi di mana bilangan bulat ke-i menunjukkan a_i.
  - Baris ketiga berisi n bilangan bulat yang dipisahkan spasi di mana bilangan bulat ke-i menunjukkan b_i.

- Output

  Untuk setiap kasus pengecekan, keluarkan satu baris yang berisi n bilangan bulat, bilangan bulat ke-i adalah menit ketika permintaan i selesai.

- Kendala
  - $1 \leq T \leq 150$
  - Jumlah n dalam semua kasus uji tidak akan melebihi 10^6
  - $1 \leq a_i, b_i \leq 10^{12}$

- Penjelasan

  Terdapat antrian di mana berisi dua permintaan ($n = 2$) dengan data sebagai berikut.

  $a_1 = 3$; $a_2 = 2$; $b_1 = 2$; $b_2 = 2$

  a_i : waktu menyelesaikan permintaan ke-i

  b_i : waktu pemilik permintaan ke-i siap

  Kita akan melihat apa yang terjadi setiap menit :
  - $waktu = 1$

    Terdapat antrian dengan urutan (1, 2). Pemilik permintaan 1 akan diperiksa kesiapannya. Proses pemeriksaan ini memakan waktu satu menit penuh. Karena permintaan 1 belum siap maka antriannya sekarang menjadi (2, 1).

- $waktu = 2$

  Pemilik permintaan 2 diperiksa kesiapannya. Proses pemeriksaan ini memakan waktu satu menit penuh. Karena siap ($b_2 \leq waktu \Leftrightarrow 2 \leq 2$), maka permintaan 2 akan diproses.
- $waktu = 3\ dan\ 4$

  Dua menit (a_2) ini digunakan untuk menyelesaikan permintaan 2, sehingga permintaan 2 selesai dalam menit 4 dan pemilik permintaan 2 keluar dari antrian.
- $waktu = 5$

  Urutan antrian sekarang (1). Pemilik permintaan 1 diperiksa kesiapannya. Proses pemeriksaan ini memakan waktu satu menit penuh. Karena siap ($b_1 \leq waktu \Leftrightarrow 2 \leq 5$), maka permintaan 1 akan diproses.
- $waktu = 6, 7,\ dan\ 8$

  Tiga menit (a_1) ini digunakan untuk menyelesaikan permintaan 1, sehingga permintaan 1 selesai dalam menit 8 dan pemilik permintaan 1 keluar dari antrian. Karena antrian sudah kosong, algoritma berhenti.

2.2 Modifikasi Permasalahan

Permasalahan pada tautan https://www.codechef.com/problems/WIQ kami lakukan modifikasi seperti penjelasan berikut.

- Input

  Pada program kami dilakukan *random* pada input sehingga algoritma dapat dijalankan untuk berbagai macam kasus. Untuk hal yang di *random* terdapat jumlah pemesan, jumlah dan jenis pesanan tiap orang, dan waktu siap tiap pemesan (b_i). Namun, kami tetap menyediakan algoritma yang membutuhkan input user (jika dibutuhkan).

- Menu dan Waktu Saji (a_i)

  Kami memberikan 6 (enam) pilihan menu yang memiliki waktu saji yang berbeda-beda. Berikut menu kami.

No.	Pesanan	Waktu Saji (menit)
1.	Pizza	5
2.	Burger	4
3.	Hotdog	3
4.	Fries	2
5.	Sundae	1
6.	Cola	1

Pada menu yang kami berikan, terdapat waktu saji (a_i) untuk setiap makanan atau minuman yang ada pada menu tersebut sehingga disaat hasil *random* menunjukkan angka 1 artinya merujuk pada waktu saji makanan pertama, angka 2 artinya merujuk pada waktu saji makanan kedua, dan selanjutnya.

- Visualisasi *Output*

  Pada algoritma modifikasi kami, diberikan juga visualisasi *output* yang *user-friendly* sehingga dapat memudahkan siapa pun saat mencoba *running* algoritma yang kami berikan.

2.3 Cara Menjawab Modifikasi Permasalahan

Langkah-langkah algoritma kami:

1. *Import* semua modul yang dibutuhkan (*tabulate* dan *random*)
2. Print tabel menu beserta waktu saji yang dibutuhkan (a_i)
3. *Random* jumlah pembeli (n)
4. *Random* jumlah pesanan setiap pembeli (pesan_berapa)
5. *Random* menu yang dipesan setiap pembeli sebanyak pesan_berapa (list_pesanan)
6. *Random* waktu siap setiap pembeli (b_i)
7. Simpan waktu siap (b_i) yang telah di-*random* yang dibutuhkan oleh setiap pembeli ke dalam list barisan

8. Terdapat barisan pembeli (barisan), variabel waktu (*time*), dan list kosong untuk menyimpan urutan final (waktu akhir)
9. Di dalam sebuah loop, periksa waktu siap yang dibutuhkan oleh pembeli pertama dalam barisan :
    a. Jika waktu siap (b_i) lebih besar dari waktu saat itu (*time*), pembeli yang berada pada urutan pertama dipindahkan ke urutan terakhir
    b. Jika waktu siap (b_i) lebih kecil atau sama dengan waktu saat itu, pembeli yang berada pada urutan pertama dipindahkan ke list urutan final (waktu akhir)

    Lakukan proses sampai barisan pembeli habis. Sebelum melanjutkan ke pembeli berikutnya, tambahkan 1 pada variabel *time* karena dibutuhkan 1 menit untuk memeriksa waktu siap setiap pembeli
10. Print hasil akhir yaitu waktu yang dibutuhkan oleh setiap pembeli untuk menerima pesanan yang dipesannya

2.4 Program

Terlampir link *coding* dari algoritma dengan menggunakan bahasa Python3 sebagai berikut: https://bit.ly/DapurKelompok1

Kemudian, untuk *screenshot input,* proses dan *output* dapat dilihat pada lampiran.

BAB III
PENUTUP

3.1 Kesimpulan

Algoritma kami bertujuan untuk meningkatkan keefektifan antrian dalam suatu restoran. Misalkan terdapat suatu restoran yang hanya menerima pesanan reservasi melalui aplikasi dengan sistem *pre-order* maksimal satu hari sebelumnya. Pembeli yang ingin melakukan reservasi akan memilih menu yang akan dipesan dan memberi tahu waktu tempuh mereka menuju restoran sebagai input. Kemudian, program akan memberi output waktu pesanan akan keluar.

Algoritma ini akan mengatur urutan paling efektif berdasarkan waktu tempuh tiap pembeli menuju restoran, dimana makanan baru mulai disiapkan setelah pembeli sampai di restoran agar kualitas makanan tetap terjamin. Dengan disusunnya algoritma tersebut dalam sebuah aplikasi akan membuat semua *customer* restoran tidak perlu menunggu pesanan terlalu lama.

REFERENSI

Fudail. (2017, 31 Mei). Contest Page | CodeChef. *Code Chef*. Diakses pada 3 Desember 2021, pranala: https://www.codechef.com/problems/WIQ

LAMPIRAN

Input

```
Ada berapa antrian?: 5
Jumlah Pembeli:  5  orang
------------------Pesanan ke-1----------------
INPUT
Ingin memesan berapa?: 1
Masukkan pesanan sesuai angka pada menu: [4]
Berapa lama perjalanan anda menuju restoran?: 4
KONFIRMASI WAKTU DAN PESANAN
Pesanan Pembeli ke-1:  ['Fries']
Waktu Saji Pesanan Pembeli ke-1:  2
Waktu Siap Pembeli ke-1:  4
------------------Pesanan ke-2----------------
INPUT
Ingin memesan berapa?: 4
Masukkan pesanan sesuai angka pada menu: [1,3,6,4]
Berapa lama perjalanan anda menuju restoran?: 11
KONFIRMASI WAKTU DAN PESANAN
Pesanan Pembeli ke-2:  ['Pizza', 'Hotdog', 'Cola', 'Fries']
Waktu Saji Pesanan Pembeli ke-2:  11
Waktu Siap Pembeli ke-2:  11
------------------Pesanan ke-3----------------
INPUT
Ingin memesan berapa?: 2
Masukkan pesanan sesuai angka pada menu: [3,4]
Berapa lama perjalanan anda menuju restoran?: 11
KONFIRMASI WAKTU DAN PESANAN
Pesanan Pembeli ke-3:  ['Hotdog', 'Fries']
Waktu Saji Pesanan Pembeli ke-3:  5
Waktu Siap Pembeli ke-3:  11

------------------Pesanan ke-4----------------
INPUT
Ingin memesan berapa?: 5
Masukkan pesanan sesuai angka pada menu: [1,5,6,5,4]
Berapa lama perjalanan anda menuju restoran?: 3
KONFIRMASI WAKTU DAN PESANAN
Pesanan Pembeli ke-4:  ['Pizza', 'Sundae', 'Cola', 'Sundae', 'Fries']
Waktu Saji Pesanan Pembeli ke-4:  10
Waktu Siap Pembeli ke-4:  3
------------------Pesanan ke-5----------------
INPUT
Ingin memesan berapa?: 1
Masukkan pesanan sesuai angka pada menu: [2]
Berapa lama perjalanan anda menuju restoran?: 5
KONFIRMASI WAKTU DAN PESANAN
Pesanan Pembeli ke-5:  ['Burger']
Waktu Saji Pesanan Pembeli ke-5:  4
Waktu Siap Pembeli ke-5:  5
```

Proses

- **Untuk *input* sesuai *user***

```
#MEMINTA INPUT
import random
from tabulate import tabulate

table=[["Pizza",5],["Burger",4],["Hotdog",3],["Fries",2],["Sundae",1],["Cola",1]]
row=[1,2,3,4,5,6]
print( '\033[1m\033[34m'+"SELAMAT DATANG DI RESTORAN KELOMPOK 1"+'\033[30m\033[0m')
print("Berikut menu kami:")
print(tabulate(table,headers=["No.","Pesanan","Waktu Saji (menit)"],tablefmt="fancy_grid",showindex=row,numalign='center'))
print("================================================")
n=int(input("Ada berapa antrian?: ")) #range jumlah pembeli ganti disini
print("Jumlah Pembeli: ",n, " orang")

list_menu=['Pizza','Burger','Hotdog','Fries','Sundae','Cola']
list_bi=[5,4,3,2,1,1]
ai=[]
bi=[]
```

```
for i in range (n):
  print('\033[1m\033[36m'+"-------------------Pesanan ke-{0}----------------".format(i+1)+'\033[30m\033[0m')
  print('\033[1m\033[30m'+"INPUT"+'\033[30m\033[0m')
  pesan_berapa =  int(input("Ingin memesan berapa?: "))            #list berapa pesanan sampe berapa pesanan
  list_pesanan_print=[]
  list_pesanan=[]
  menuinput=[]
  random_pesanan = eval(input("Masukkan pesanan sesuai angka pada menu: "))
  for r in range (pesan_berapa):
    list_pesanan_print.append(list_menu[random_pesanan[r]-1])
    menuinput.append(list_bi[random_pesanan[r]-1])
  biinput=int(input("Berapa lama perjalanan anda menuju restoran?: "))
  print('\033[1m\033[30m'+"KONFIRMASI WAKTU DAN PESANAN"+'\033[30m\033[0m')
  print("Pesanan Pembeli ke-{0}: ".format(i+1), list_pesanan_print)
  print("Waktu Saji Pesanan Pembeli ke-{0}: ".format(i+1), sum(menuinput))
  print("Waktu Siap Pembeli ke-{0}: ".format(i+1), biinput)
```

```
  try:
    ai.append(sum(menuinput))
  except:
    ai.append(menuinput)
  bi.append(biinput)
time=1
waktuakhir=[]

barisan=[]
for i in range (n):
  barisan.append(i)
  waktuakhir.append(0)

while len(barisan)!=0:
  if bi[barisan[0]]>time:
     barisan.append(barisan[0])
     barisan.remove(barisan[0])

  elif bi[barisan[0]]<=time:
    time=time+ai[barisan[0]]
    waktuakhir[barisan[0]]=time
    barisan.remove(barisan[0])
  time=time+1
print("================================================")
print('\033[1m\033[32m'+"Total Waktu Pesanan Diterima"+'\033[30m\033[0m')
for i in range (len(waktuakhir)):
  print("Pembeli ke-{0} : {1} menit".format(i+1,waktuakhir[i]))
```

- **Untuk *random input***

```
#MENGGUNAKAN RANDOM

import random
from tabulate import tabulate

table=[["Pizza",5],["Burger",4],["Hotdog",3],["Fries",2],["Sundae",1],["Cola",1]]
row=[1,2,3,4,5,6]
print( '\033[1m\033[34m'+"SELAMAT DATANG DI RESTORAN KELOMPOK 1"+'\033[30m\033[0m')
print("Berikut menu kami:")
print(tabulate(table,headers=["No.","Pesanan","Waktu Saji (menit)"],tablefmt="fancy_grid",showindex=row,numalign='center'))
print("==================================================")
n=random.randint(1,5)
print("Jumlah Pembeli: ",n, " orang")

list_menu=['Pizza','Burger','Hotdog','Fries','Sundae','Cola']
list_bi=[5,4,3,2,1,1]
ai=[]
bi=[]

for i in range (n):
  print("--------------------------------------------------")
  pesan_berapa = random.randint(1,5)
  list_pesanan_print=[]
  list_pesanan=[]
  menuinput=[]
  for r in range (pesan_berapa):
    random_pesanan = random.randint(1,6)
    list_pesanan.append(random_pesanan)
    list_pesanan_print.append(list_menu[random_pesanan-1])
    menuinput.append(list_bi[random_pesanan-1])
  print("Pesanan Pembeli ke-{0}: ".format(i+1), list_pesanan_print)
  biinput=random.randint(1, 11)
  print("Waktu Saji Pesanan Pembeli ke-{0}: ".format(i+1), sum(menuinput))
  print("Waktu Siap Pembeli ke-{0}: ".format(i+1), biinput)
```

```
  try:
    ai.append(sum(menuinput))
  except:
    ai.append(menuinput)
  bi.append(biinput)
time=1
waktuakhir=[]

barisan=[]
for i in range (n):
  barisan.append(i)
  waktuakhir.append(0)

while len(barisan)!=0:
  if bi[barisan[0]]>time:
     barisan.append(barisan[0])
     barisan.remove(barisan[0])

  elif bi[barisan[0]]<=time:
    time=time+ai[barisan[0]]
    waktuakhir[barisan[0]]=time
    barisan.remove(barisan[0])
  time=time+1
print("==================================================")
print('\033[1m\033[32m'+"Total Waktu Pesanan Diterima"+'\033[30m\033[0m')
for i in range (len(waktuakhir)):
  print("Pembeli ke-{0} : {1} menit".format(i+1,waktuakhir[i]))
```

Output

```
Total Waktu Pesanan Diterima
Pembeli ke-1 : 22 menit
Pembeli ke-2 : 34 menit
Pembeli ke-3 : 40 menit
Pembeli ke-4 : 14 menit
Pembeli ke-5 : 19 menit
```