I. TUJUAN

- a. Praktikan mampu memahami algoritma bubble sort.
- b. Praktikan mampu mengimplementasikan algoritma bubble sort

II. ALAT DAN BAHAN

- a. Laptop
- b. Modul
- c. Pycharm

III TEORI DASAR

Bubble sort merupakan cara pengurutan yang sederhana. Konsep dari ide dasarnya adalah seperti "gelembung air" untuk elemen struktur data yang semestinya berada pada posisi awal. Pada dasarnya konsep utama dari pengurutan adalah memanipulasi index dari list array, dengan cara menukar isi array dengan array lainnya. Cara kerja dari algoritma ini adalah mengulang proses pembandingan antara tiap-tiap elemen array dan menukarnya apabila urutannya salah. Pembandingan pada elemen-elemen ini akan terus diulang hingga tidak perlu dilakukan penukaran lagi.

Beberapa kelebihan dari algoritma *Bubble sort* adalah sebaga berikut :

- a. Algoritma yang simpel
- b. Mudah untuk diubah menjadi kode.
- c. Definisi terurut terdapat dengan ielas dalam algoritma.
- d. Cocok untuk pengurutan data dengan elemen kecil telah terurut.

Beberapa kekurangan dari algoritma *Bubble sort* adalah sebagai berikut :

a. Tidak etektit dalam pengurutan data berskala besar.

b. Langkah pengurutan yang terlalu panjang.

Kompleksitas Algoritma Bubble sort

Kompleksitas Algoritma *Bubble sort* dapat dilihat dari beberapa jenis kasus, yaitu *worst-case*, *average-case*, dan *best-case*.

a. Kondisi Best-Case

Dalam kasus ini, data yang akan disorting telah terurut sebelumnya, sehingga proses perbandingan hanya dilakukan sebanyak (n-1) kali, dengan satu kali *pass*. Contoh *Best-Case* dapat dilihat pada pengurutan data "1 2 3 4" di bawah ini.

Pass Pertama

(1 2 3 4) menjadi (1 2 3 4)

(1234) menjadi (1234)

(1 2 3 4) menjadi (1 2 3 4)

Dari proses di atas, dapat dilihat bahwa tidak terjadi penukaran posisi satu kalipun, sehingga tidak dilakukan *pass* selanjutnya. Perbandingan elemen dilakukan sebanyak tiga kali. Proses perbandingan pada kondisi ini hanya dilakukan sebanyak (n-1) kali. Dengan kata lain, pada kondisi *Best-Case* algoritma *Bubble sort* termasuk pada algoritma lanjar.

b. Kondisi *Worst-case*

Dalam kasus ini, data terkecil berada pada ujung *array*. Contoh *Worst-case* dapat dilihat pada pengurutan data "4 3 2 1" di bawah ini

Pass Pertama

(4 3 2 1) menjadi (3 4 2 1)

(3 4 2 1) menjadi (3 2 4 1)

(3 2 4 1) menjadi (3 2 1 4)

Pass Kedua

(3 2 1 4) meniadi (2 3 1 4)

```
(2 3 1 4) menjadi (2 1 3 4)
```

(2 1 3 4) menjadi (2 1 3 4)

Pass Ketiga

(2 1 3 4) menjadi (1 2 3 4)

(1 2 3 4) menjadi (1 2 3 4)

(1234) menjadi (1234)

Pass keempat

(1 2 3 4) menjadi (1 2 3 4)

(1 2 3 4) menjadi (1 2 3 4)

(1 2 3 4) menjadi (1 2 3 4)

Dari langkah pengurutan di atas, terlihat bahwa setiap kali melakukan satu *pass*, data terkecil akan bergeser ke arah awal sebanyak satu *step*. Dengan kata lain, untuk menggeser data terkecil dari urutan keempat menuju urutan pertama, dibutuhkan *pass* sebanyak tiga kali, ditambah satu kali *pass* untuk memverifikasi.

c. Kondisi Average-Case

Pada kondisi *average-case*, jumlah *pass* ditentukan dari elemen mana yang mengalami penggeseran ke kiri paling banyak. Hal ini dapat ditunjukkan oleh proses pengurutan suatu *array*, misalkan saja (1 8 6 2). Dari (1 8 6 2), dapat dilihat bahwa yang akan mengalami proses penggeseran paling banyak adalah elemen 2, yaitu sebanyak dua kali.

Pass Pertama

(1862) menjadi (1862)

(1862) menjadi (1682)

(1682) menjadi (1628)

Pass Kedua

(1628) menjadi (1628)

(1628) menjadi (1268)

(1 2 6 8) menjadi (1 2 6 8)

Pass Ketiga

(1 2 6 8) menjadi (1 2 6 8)

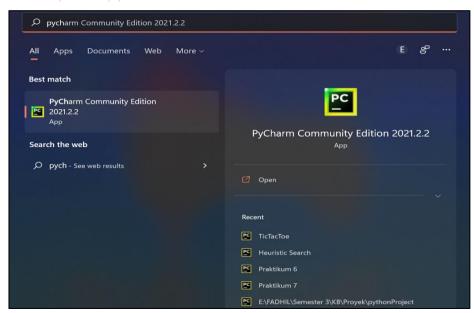
(1 2 6 8) menjadi (1 2 6 8)

(1 2 6 8) menjadi (1 2 6 8)

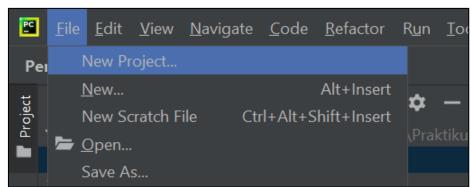
Dari proses pengurutan di atas, dapat dilihat bahwa untuk mengurutkan diperlukan dua buah *pass*ing, ditambah satu buah *pass*ing untuk memverifikasi. Dengan kata lain, jumlah proses perbandingan dapat dihitung.

IV. LANGKAH KERJA

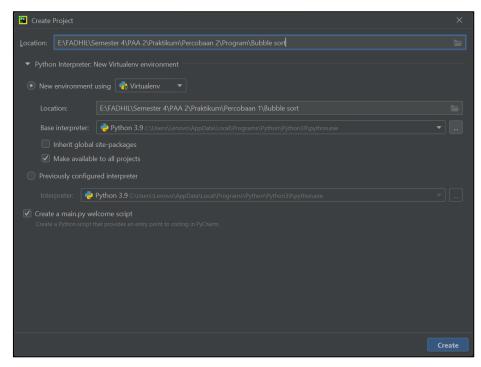
a. Buka aplikasi pycharm.



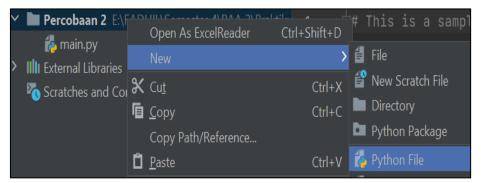
b. Klik file, new project



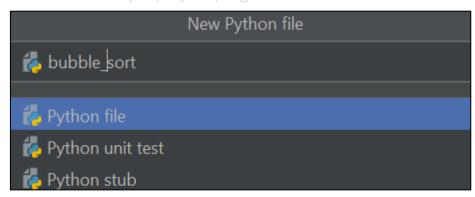
c. Tentukanlah letak *file project* dan beri nama *project* lalu klik *create*.



d. Klik kanan pada nama file yang telah di buat, kemudian pilih new file.



e. Tentukan nama *file project* yang di buat.



f. Masukkan kode program berikut

```
#Muhammad Fadhil
#F55120068

import time

def bubbleSort(arr):
    n = len(arr)

for i in range(n):
    if arr[j] > arr[j + 1]:
        arr[j], arr[j + 1] = arr[j + 1], arr[j]

arr = [l
    l = int(input("Masukkan panjang array:"))

for i in range(l):
    k = int(input("Nilai array :"))

arr.append(k)

print("Array", arr)

times = time.time()

bubbleSort(arr)

print("Nsorted Array is : ")

for i in range(len(arr)):
    print("%d" % arr[i])

print("%d" % arr[i])

print("%d" % second" %(time.time()-times))
```

V. HASII PERCOBAAN

```
"E:\FADHIL\Semester 4\PAA 2\Praktikum\Percobaan 1\Percobaan 2\Scripts\python.exe"
Masukkan panjang array:
Nilai array:

Nilai array:

Nilai array:

O
0.0 Second

O
0.0 Second

O
0.0010113716125488281 Second

1
0.0010113716125488281 Second

4
0.0010113716125488281 Second

8
0.0010113716125488281 Second

Process finished with exit code 0
```

VI ANALISIS

Dari percobaan yang telah dilakukan dapat dianalisis bahwa untuk mengimplementasi algoritma bubble sort, pertama kita mengimport Ibrary time, yang berfungsi menghitung lama waktu nilai array di proses, kemudian masukkan kode program "def bubbleSort(arr):",untuk mendefinisikan algoritma buble sort dengan menggunakan variabel array "n = len(arr)" untuk menghetahui panjang atau jumlah dari data array yang di masukkan. Terdapat perulangan yaitu "for in range(n):" digunakan untuk memasukkan jumlah array, kemudian ada variabel "for j in range (0, n-i-1)" dimana 0 s=diambil dari angka array , "n-i-1" merupakan formula dari pengurangan jumlah array yang dikurangi dengan I dan 1 hingga selesai. Variabel "if arr[j] > arr[j + 1]" . arr[j] merupakan kondisi yang digunakan untuk mengurutkan nilai array. "I = int (input (masukkan nilai array))" sebagai interface untuk memasukkan panjang array dan untuk menginputnya diberi variabel untuk menambahkan nilai array pada urutan. "times= time.time()" untuk menghitung waktu

VII. KESIMPULAN

Pada percobaan ini kita dapat mengetahui bahwa algoritma bubble sort merupakan proses pengurutan secara berkala sehingga dapat terurut keposisi yang tepat, hal tersebutlah sehingga dinamakan *Bubble*(gelembung). Algoritma ini akan mengurutkan data dari yang terbesar ke yang terkecil atau sebaliknya yang biasa disebut dengan pengurutan naik (ascending) atau pengurutan turun (descending).

Dengan begitu kita dapat menyimpulkan bahwa algoritma bubble sort merupakan pengurutan data dengan cara menukar

data dengan data yang berada disebelahnya yang akan dilakukar perulangan sehingga dalam iterasi tersebut data akan terurut secara *ascending* maupun *discending*. PERCOBAAN II

Bubble Sort