KOMPARASI BUCKET SORT dan BUBBLE SORT

Muhammad Akbar Faiz (1303184019)
Pratama Azmi Atmajaya(1303180096)

S1 Teknologi Informasi
Jl. Telekomunikasi Terusan Buah Batu Bandung 40257
Email : akbarfaiz@student.telkomuniversity.ac.id ,pramzz@student.telkomuniversity.ac.id

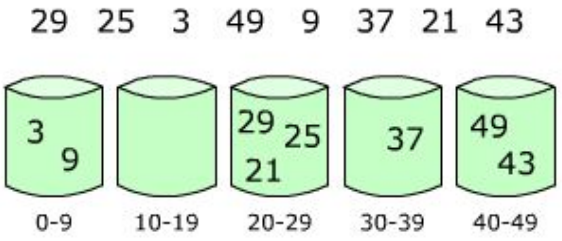
PENDAHULUAN

Pengurutan data dalam struktur data sangat penting untuk data yang bertipe data numerik ataupun karakter sehingga digunakan secara luas dalam aplikasi dimana data tersebut tersusun dengan pola tertentu sehingga tersusun secara teratur berdasarkan aturan tertentu.Menurut Book-shelf, definisi algoritma pengurutan adalah algoritma untuk meletakan kumpulan elemen data ke dalam urutan tertentu berdasarkan satu atau beberapa kunci dari tiap-tiap elemen.Ada beberapa keuntungan dari pengurutan ini memudahkan dan mempercepat pencarian serta pengecekan elemen dari suatu data.Pengurutan data erat kaitannya dengan pencarian data dan pengelolaan data, dimana dalam kedua hal tersebut terdapat beberapa proses dalam memilah-milah data untuk kebutuhan tertentu.Pencarian data dan pengelolaan data yang efektif tidak dapat terlepas dari faktor keterurutan data yang baik.Apabila data-data yang ada tidak diolah dengan tepat maka akan tercecer dan sulit dipergunakan untuk kepentingan tertentu.Oleh sebab itu, dibutuhkannya suatu algoritma yang dapat mengurutkan data sesuai aturan tertentu agar efektif dan efisien.Dalam makalah ini akan membahas perbandingan penggunaan algoritma Bucket sort dan Bubble sort dalam mengurutkan elemen-elemen dalam suatu data.Bucket sort ini sendiri jika diartikan ke dalam bahasa indonesia per kata akan menjadi urut dan ember dimana algoritma ini akan memilah elemen ke dalam ember tertentu sesuai aturan yang ditetapkan.Sedangkan Bubble sort memiliki arti gelembung dimana proses pengurutan data dengan cara pertukaran data dengan data di sebelahnya secara terus menerus sampai pada satu iterasi tertentu tidak ada lagi perubahan yang signifikan.

Kata kunci : Pengurutan,data,elemen,susun,algoritma.

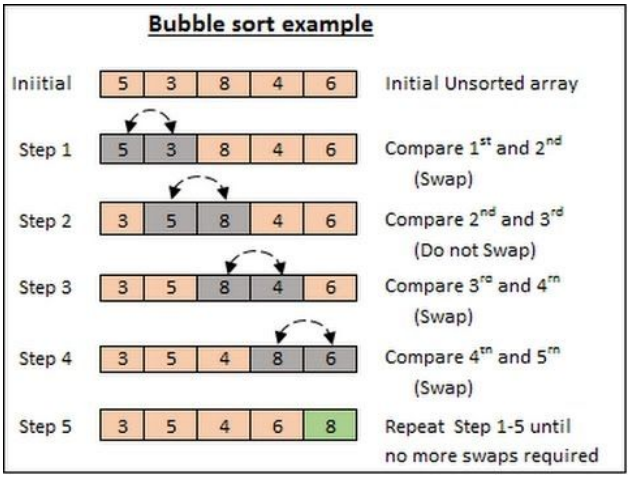
I. PENJELASAN PROGRAM

Algoritma yang akan digunakan untuk mengurutkan data adalah *Bucket sort* dan *Bubble sort*.Di dalam algoritma *Bucket sort* array dari elemen dialokasi ke dalam beberapa ember.Sesuai namanya,setiap ember akan menampung beberapa elemen yang mirip atau serupa dan ember-ember tersebut akan diurutkan satu sama lain menggunakan algoritma pengurutan lainnya ataupun dengan menerapkan pengurutan ember secara rekursif.



Gambar 1. Contoh Bucket sort kepada suatu data

Sedangkan untuk algortima *Bubble sort* akan mengurutkan data dalam array yang saling bersebelahan secara berulang-ulang sampai tidak ada lagi perubahan urutan antar data.



Gambar 2. Contoh Bubble sort kepada suatu data

II. STRATEGI ALGORITMA

Ada beberapa hal yang akan diperkirakan dalam algoritma *Bucket sort* dan *Bubble sort*.Hal-hal ini akan dipertimbangkan dalam strategi algoritma.

2.1 Kompleksitas Waktu

Setiap algoritma memiliki kompleksitas waktu yang berbeda.Untuk rumus perhitungan kompleksitas waktu dari *Bucket sort* sebagai berikut :

$$C(n) = \sum_{i=0}^{k} \sum_{j=1}^{n} 1 = \sum_{i=0}^{k} [\, n_i - 1 + 1] = \sum_{i=0}^{k} (\, n_i)$$

Dari rumus diatas,kompleksitas waktu dari *Bucket sort* dapat diketahui.Untuk itu dapat dibagi menjadi 3 kondisi yang mungkin terjadi :

Tabel 1 Kompleksitas Waktu *Bucket sort*

Kondisi	Persamaan	Keterangan
Worst	O(n^2)	Kasus terburuk terjadi ketika elemen dalam ember berada dalam urutan terbalik.
Average	O(n)	Ini terjadi ketika elemen didistribusikan secara acak dalam array. Bahkan jika elemen tidak didistribusikan secara seragam, bucket sort berjalan dalam waktu linear.
Best	O(n+k)	ketika elemen didistribusikan secara seragam dalam ember dengan jumlah elemen yang hampir sama di setiap ember. Kompleksitas menjadi lebih baik ketika elemen di dalam setiap ember sudah diurutkan.

Adapun rumus untuk kompleksitas waktu dari *Bubble sort* sebagai berikut :

$$C(n) = \sum_{i=0}^{n} \sum_{j=0}^{n-i-1} 1 = \sum_{i=0}^{n} [(n - i - 1) - 0 + 1]$$

$$C(n) = \sum_{i=0}^{n} n - i = (n - 1) + (n - 2) + \dots 1 = \frac{(n-1)\,n}{2}$$

Dari rumus diatas,kompleksitas waktu dari *Bucket sort* dapat diketahui.Untuk itu dapat dibagi menjadi 3 kondisi yang mungkin terjadi :

Tabel 2 Kompleksitas Waktu *Bubble sort*

Kondisi	Persamaan	Keterangan
Worst	O(n^2)	Kasus terburuk terjadi ketika data yang seharusnya diawal berada diurutan terakhir.
Average	O(n^2)	Ini terjadi ketika elemen yang seharusnya berada ditengah terletak pada urutan terakhir dalam kondisi awal.
Best	O(n)	Data yang akan diurutkan telah terurut sebelumnya. Sehingga proses perbandingan hanya dilakukan sebanyak (n-1) kali, dengan satu kali iterasi.

2.2 Kelas Efisiensi

Berdasarkan rumus kompleksitas waktu yang ada,bisa ditentukan kelas efisiensi dari kedua algoritma yang dibandingkan.Untuk *Bucket sort* memiliki kelas efisiensi linear.Sedangkan *Bubble sort* memiliki kelas efisiensi kuadratik.

III. FUNGSIONAL PROGRAM

3.1 Penjelasan Pseudocode

Langkah-langkah pada pengurutan data dengan algoritma *Bucket sort* adalah sebagai berikut.

1. Membuat suatu inisial array yaitu ember kosong sebanyak jumlah array yang diberikan.
2. Memasukan setiap objek pada array yang ingin diurutkan ke dalam ember yang sesuai.
3. Melakukan pengurutan pada setiap objek yang ada di tiap ember, disini menggunakan *Insertion sort*.

Langkah-langkah pada pengurutan data dengan algortima *Bubble sort* adalah sebagai berikut.

1. Jumlah iterasi sama dengan banyaknya bilangan dikurang 1.
2. Di setiap iterasi, jumlah pertukaran bilangannya sama dengan jumlah banyaknya bilangan.

3. Dalam algoritma Bubble Sort, meskipun deretan bilangan tersebut sudah terurut, proses sorting akan tetap dilakukan.
4. Tidak ada perbedaan cara yang berarti untuk teknik algoritma Bubble Sort Ascending dan Descending.

3.2 Pseudocode

3.2.1 Bucket Sort

```
function bucket_sort(arr : list of unlisted items
,SlotBucket : int)->list of sorted items
  I.S : Array telah Terbentuk dengan n elements
dan belum ter-urut
  F.S : Array telah terurut dengan algoritma
BucketSort

  Arr : Adalah Array dengan N elements
  Slotbucket : Jumlah bucket yang akan
digunakan

inisiasi:
  buckets = []
  bucketLocation = 0
  Sortedbucket = []

Algoritma

for i in 0 to SlotBucket
    buckets.insert([])

for i in 0 to length(arr)
    bucketLocation = int(arr[i] * SlotBucket)
    buckets[bucketLocation].insert(arr[i])

for i in 0 to length(buckets)
    buckets[i] = insertion_sort(buckets[i])

for i in 0 to length(SlotBucket)
    for j in 0 to length(buckets[i])
        Sortbucket.insert(buckets[i][j])

return (Sortedbucket)
```

3.2.2 Bubble Sort

```
function bubblesort(arr : list of unlisted items)->
list of sorted items

  I.S : Array telah terbentuk dengan n elements
dan belum terurut
  F.S : Array Telah Terurut dengan algoritma
bubblesort

  arr : Array dengan N Elements

Algortima

  for i in 0 to length(arr)
    for j in 0 to length(arr) - i - 1
      if arr[i] > arr[j+1] then
        swap { arr[j] , arr[j+1]}
  return arr
```

3.2.3 Insertion Sort

Ada nya algoritma *Insertion sort* dikarenakan kebutuhan pada pengurutan di dalam ember *Bucket sort*.

```
function insertion_sort(arr : list of unlisted
items)-> list of sorted items

  I.S : Array telah terbentuk dengan n elements
dan belum terurut
  F.S : Array Telah Terurut dengan algoritma
Insertionsort

  arr : Array dengan N Elements

Algoritma

  for i in 0 to length(arr)
    j = i - 1

    while j >= 0 and array[i] < array[j]:
      array[j + 1] = array[j]
      j = j - 1

  return arr
```

3.3 Hasil Running Time

Hasil ini didapat dari pengujian program dengan jumlah inputan tertentu.Dari hasil waktu berjalannya kedua algoritma dapat dibandingkan satu sama lain untuk melihat keefesiensinya.

Tabel 3 *Running Time Bucket sort*

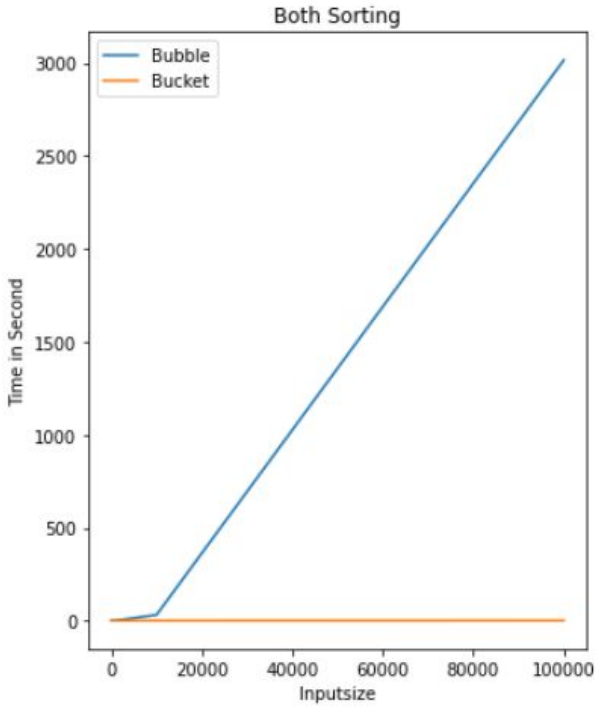
Ukuran Input N(n)	Running Time (microsecond)
10	38.862
100	193.35
1000	2236.98
10000	26561.66
100000	288027.80

Tabel 3 *Running Time Bubble sort*

Ukuran Input N(n)	Running Time (microsecond)
10	388.86
100	2074.48
1000	268568.754
10000	31122339.70
100000	3015600455.99

3.4 Perbandingan Antar Algoritma

Setelah mendapatkan hasil running time dari kedua algoritma pengurutan maka dapat dibuat grafik untuk menunjukan perbedaan yang signifikan dari kedua algoritma.



Gambar 3. Perbandingan Bubble sort dan Bucket sort

IV. SCREEN SHOOT OUTPUT PROGRAM

```
np.random.seed(5)
nums = np.random.random(10)
print(f"Array Sebelum di Sort \n{nums}")
begin = time.time()
print(f"\nArray setelah di sort dengan bucket sort\n")
print(bucket_sort(nums,len(nums)))
end = time.time()
print(f"\nDengan Waktu Eksekusi {end-begin} Second")
```

```
Array Sebelum di Sort
[0.22199317 0.87073231 0.20671916 0.91861091 0.48841119 0.61174386
 0.76590786 0.51841799 0.2968005  0.18772123]

Array setelah di sort dengan bucket sort

[0.18772123 0.20671916 0.22199317 0.2968005  0.48841119 0.51841799
 0.61174386 0.76590786 0.87073231 0.91861091]

Dengan Waktu Eksekusi 0.0005364418029785156 Second
```

Gambar 4. Output Bucket sort

```
np.random.seed(5)
nums = np.random.random(10)
print(f"Array Sebelum di Sort \n{nums}")
begin = time.time()
print(f"\nArray setelah di sort dengan Bubble sort \n{bubblesort(nums)}")
end = time.time()
print(f"\nDengan Waktu Eksekusi {end-begin} second")
```

```
Array Sebelum di Sort
[0.22199317 0.87073231 0.20671916 0.91861091 0.48841119 0.61174386
 0.76590786 0.51841799 0.2968005  0.18772123]

Array setelah di sort dengan Bubble sort
[0.18772123 0.20671916 0.22199317 0.2968005  0.48841119 0.51841799
 0.61174386 0.76590786 0.87073231 0.91861091]

Dengan Waktu Eksekusi 0.00037097930908203125 second
```

Gambar 5. Output Bubble sort

V. REFERENSI

[1] Parta, https://kreatip.id/materi/implementasi-algoritma/sorting. Diakses tanggal 9 Desember 2020.

[2] Neeraj Mishra, https://www.thecrazyprogrammer.com/2017/02/bucket-sort-in-c.html.Diakses tanggal 9 Desember 2020.

[3] Prashant Yadav, Bucket Sort Algorithm | LearnersBucket.Diakses tanggal 15 Desember 2020.

[4] Sutiono S.Kom., M.Kom., M.T., Apa itu Algoritma Bubble Sort? - DosenIT.com.Diakses tanggal 29 Desember 2020.

[5] JohnPaul Adamovsky, https://pages.pathcom.com/~vadco/binary.html.Diakses tanggal 3 Januari 2021.