ALGORITMA DAN STRUKTUR DATA

INF1083

LAPORAN PRAKTIKUM 10 : Sorting

Oleh :

Akhmad Qasim

2211102441237

Teknik Informatika
Fakultas Sains & Teknologi
Universitas Muhammadiyah Kalimantan Timur

Samarinda, 2023

Laporan Praktikum 10:
Sorting

Pokok Bahasan:

- ❖ Sorting
- ❖ Teknik-Teknik Sorting

Tujuan Pembelajaran:

- ✓ Memahami implementasi algoritma Sorting dengan menggunakan teknik-teknik Sorting.
- ✓ Memahami perbedaan algoritma masing-masing teknik Sorting.

Percobaan & Latihan:

1. **Bubble Sort**
   a. **Berikan tampilan output dari kedua program diatas!**
   Tampilan output:

```
[17, 20, 26, 31, 44, 54, 55, 77, 93]
[20, 30, 40, 50, 60, 70, 80, 90, 100, 110]
```

   b. **Jelaskan perbedaan algoritma kedua fungsi pada program diatas!**
   Pada algoritma Bubble Sort pertama, setiap iterasi dilakukan penukaran elemen (swapping) pada pasangan angka yang dibandingkan (pairwise comparison), sehingga tidak terdapat optimasi pada jumlah iterasi. Sedangkan pada algoritma Bubble Sort yang kedua, terdapat optimasi dengan menambahkan variabel exchanges yang akan menandai apakah terdapat penukaran elemen pada iterasi tersebut. Jika tidak ada penukaran elemen pada iterasi tersebut, maka iterasi dapat dihentikan karena sudah tidak ada lagi pasangan angka yang perlu ditukar. Dengan demikian, algoritma kedua menjadi lebih efisien daripada algoritma Bubble Sort pertama, terutama pada data dengan jumlah elemen yang besar dan hampir terurut secara ascending (data yang hampir terurut secara descending masih akan memerlukan banyak iterasi).

   c. **Algoritma mana yang terbaik dari kedua fungsi tersebut? Jelaskan alasannya!**
   Kedua algoritma tersebut sama-sama menggunakan metode bubble sort. Namun, shortBubbleSort memiliki sedikit perbedaan dengan bubbleSort

dalam hal optimasi, yaitu menggunakan variabel exchanges untuk menghentikan loop jika tidak ada pertukaran elemen. Sehingga, jika dalam iterasi terakhir tidak ada pertukaran elemen, algoritma akan langsung berhenti.

Oleh karena itu, shortBubbleSort akan lebih cepat daripada bubbleSort pada kasus-kasus di mana data sudah hampir terurut atau hanya memiliki sedikit perubahan. Namun, pada kasus di mana data tidak terurut secara acak, keduanya akan memiliki performa yang sama.

2. **Selection Sort**
   a. **Berikan tampilan output dari program diatas!**
   Tampilan output:

```
[17, 20, 26, 31, 44, 54, 55, 77, 93]
```

   b. **Berikan penjelasan dari baris ke-2 hingga ke-10!**
   Fungsi tersebut namanya Selection Sort, untuk menyelesaikannya dengan mencari nilai terbesar pada list dan menukarnya dengan elemen terakhir dari list yang belum terurut. Kemudian, terus melakukan hal yang sama dengan list yang lebih kecil (terakhir - 1) hingga semua elemen dalam list sudah terurut.

   Pada implementasi kode di atas, fungsi selectionSort menerima satu parameter yaitu sebuah list yang akan diurutkan. Fungsi kemudian melakukan perulangan untuk mengisi setiap slot dalam list dengan elemen yang sudah terurut. Perulangan dimulai dari posisi terakhir dalam list dan terus berkurang hingga elemen pertama.

   Dalam setiap perulangan, fungsi menemukan elemen terbesar dari slot yang belum terurut, menggunakan variabel positionOfMax untuk menyimpan posisi dari elemen tersebut. Setelah elemen terbesar ditemukan, fungsi menukar elemen terakhir dalam list yang belum terurut dengan elemen terbesar yang ditemukan. Hal ini dilakukan agar elemen terbesar berada di posisi terakhir dan tidak lagi dianggap dalam proses sorting selanjutnya. Setelah semua perulangan selesai, list akan terurut secara ascending (dari yang terkecil ke yang terbesar).

3. **Insertion Sort**
   a. **Berikan tampilan output dari program diatas!**
   Tampilan output:

```
[17, 20, 26, 31, 44, 54, 55, 77, 93]
```

b. Berikan penjelasan dari baris ke-2 hingga ke-11!

Fungsi diatas adalah salah satu algoritma pengurutan (sorting) yang menggunakan metode penyisipan (insertion). Algoritma ini dimulai dari elemen kedua dalam daftar indeks pertama, kemudian membandingkan nilai dari elemen tersebut dengan nilai elemen-elemen sebelumnya. Jika nilai dari elemen sebelumnya lebih besar, maka elemen tersebut akan digeser ke kanan satu per satu sampai menemukan posisi yang tepat untuk menyisipkan elemen yang diurutkan tersebut. Proses ini akan terus berlanjut hingga semua elemen dalam daftar telah terurut.

Pada intinya, algoritma ini akan membagi daftar menjadi dua bagian: bagian yang diurutkan dan bagian yang belum diurutkan. Kemudian, elemen pertama pada bagian yang belum diurutkan akan diambil dan disisipkan pada posisi yang tepat di dalam bagian yang diurutkan. Proses ini terus dilakukan hingga seluruh elemen pada bagian yang belum diurutkan telah disisipkan pada posisi yang tepat di dalam bagian yang diurutkan.

4. Shell Sort

a. Berikan tampilan output dari program diatas!

Tampilan output:

```
After increments of size 4 The list is [20, 26, 44, 17, 54, 31, 93, 55, 77]
After increments of size 2 The list is [20, 17, 44, 26, 54, 31, 77, 55, 93]
After increments of size 1 The list is [17, 20, 26, 31, 44, 54, 55, 77, 93]
[17, 20, 26, 31, 44, 54, 55, 77, 93]
```

b. Jelaskan kedua fungsi pada program diatas!

Kedua fungsi di atas merupakan implementasi dari algoritma Shell Sort, sebuah algoritma sorting yang membagi list menjadi beberapa sublist, kemudian mengurutkan sublist tersebut menggunakan algoritma insertion sort dengan jarak tertentu.

Pada fungsi shellSort, list dipecah menjadi beberapa sublist dengan gap yang berkurang setiap iterasi. Setiap sublist diurutkan menggunakan fungsi gapInsertionSort. Setelah iterasi selesai, list akan terurut.

Pada fungsi gapInsertionSort, sublist diurutkan menggunakan algoritma insertion sort dengan gap yang diberikan. Algoritma insertion sort ini dimodifikasi untuk memperhitungkan gap, yaitu dengan mengurutkan

elemen-elemen yang berjarak gap dengan menggunakan metode pengurutan insertion sort.

c. Pada baris ke-11 terdapat variabel "sublistcount", jelaskan untuk apa variabel tersebut!

Berfungsi sebagai pengatur jumlah sub-list yang dibuat selama sorting menggunakan algoritma Shell Sort. Pada setiap iterasi, sublistcount akan dibagi 2 sehingga panjang sub-list yang dibuat menjadi lebih kecil dan jumlah perulangan sorting yang dilakukan semakin banyak. Semakin banyak perulangan sorting yang dilakukan, semakin optimal hasil sorting yang diperoleh.

5. Merge Sort

a. Berikan tampilan output dari program diatas!

Tampilan output:

```
Splitting  [54, 26, 93, 17, 77, 31, 44, 55, 20]
Splitting  [54, 26, 93, 17]
Splitting  [54, 26]
Splitting  [54]
Merging  [54]
Splitting  [26]
Merging  [26]
Merging  [26, 54]
Splitting  [93, 17]
Splitting  [93]
Merging  [93]
Splitting  [17]
Merging  [17]
Merging  [17, 93]
Merging  [17, 26, 54, 93]
Splitting  [77, 31, 44, 55, 20]
Splitting  [77, 31]
Splitting  [77]
Merging  [77]
Splitting  [31]
Merging  [31]
Merging  [31, 77]
```

```
Splitting  [44, 55, 20]
Splitting  [44]
Merging  [44]
Splitting  [55, 20]
Splitting  [55]
Merging  [55]
Splitting  [20]
Merging  [20]
Merging  [20, 55]
Merging  [20, 44, 55]
Merging  [20, 31, 44, 55, 77]
Merging  [17, 20, 26, 31, 44, 54, 55, 77, 93]
[17, 20, 26, 31, 44, 54, 55, 77, 93]
```

b. Analisa hasil program diatas menggunakan jumlah item dalam "alist", jumlah splitting dan merging yang dilakukan!
Dapat dilihat bahwa jumlah item dalam "alist" adalah 9. Kemudian, dilakukan splitting pada daftar tersebut dengan cara membagi daftar menjadi beberapa bagian yang lebih kecil.

Pada proses sorting menggunakan algoritma quicksort, terdapat 5 kali proses splitting yang dilakukan. Proses splitting pertama pada daftar [54, 26, 93, 17, 77, 31, 44, 55, 20] menghasilkan dua daftar, yaitu [54, 26, 93, 17] dan [77, 31, 44, 55, 20]. Kemudian, proses splitting kedua menghasilkan dua daftar lagi, yaitu [54, 26] dan [93, 17], dan [77, 31] serta [44, 55, 20]. Demikian seterusnya hingga pada akhirnya daftar tersebut terpecah menjadi daftar yang terdiri atas satu elemen saja.

Setelah proses splitting selesai dilakukan, kemudian dilakukan proses merging atau penggabungan daftar yang telah terpecah menjadi bagian-bagian kecil. Setiap proses merging yang dilakukan akan ditampilkan pada console bersama dengan isi daftar pada saat itu. Pada akhirnya, hasil pengurutan atau sorting daftar [54, 26, 93, 17, 77, 31, 44, 55, 20] sesuai dengan algoritma quicksort adalah [17, 20, 26, 31, 44, 54, 55, 77, 93].

c. Lakukanlah uji coba dengan jumlah item "alist" yang berbeda, kemudian analisa hasil jumlah splitting dan merging-nya!
Tampilan kode Python:

```python
def mergeSort(alist):
    print("Splitting ", alist)
    if len(alist) > 1:
        mid = len(alist) // 2
        lefthalf = alist[:mid]
        righthalf = alist[mid:]

        mergeSort(lefthalf)
        mergeSort(righthalf)

        i = 0
        j = 0
        k = 0
        while i < len(lefthalf) and j < len(righthalf):
            if lefthalf[i] < righthalf[j]:
                alist[k] = lefthalf[i]
                i = i + 1
            else:
                alist[k] = righthalf[j]
                j = j + 1
            k = k + 1

        while i < len(lefthalf):
            alist[k] = lefthalf[i]
            i = i + 1
            k = k + 1

        while j < len(righthalf):
            alist[k] = righthalf[j]
            j = j + 1
            k = k + 1
    print("Merging ", alist)


alist = [9, 1, 8, 2, 7, 3, 6, 4, 5]
mergeSort(alist)
print(alist)
```

Penjelasan:
Pada hasil uji coba tersebut, jumlah splitting dan merging-nya sama yaitu:

Splitting: 8 kali
Merging: 8 kali

Pada setiap splitting, jumlah item pada list dipecah menjadi dua bagian hingga hanya tersisa satu item pada setiap bagian, sedangkan pada merging, dua bagian yang telah diurutkan digabungkan kembali menjadi satu list dengan urutan yang sudah terurut. Pada kasus pertama, list memiliki 9 item sehingga pada saat splitting, terjadi 8 kali pembagian list menjadi dua bagian dan pada saat merging, terjadi 8 kali penggabungan dua bagian yang telah diurutkan. Pada kasus kedua, list juga memiliki 9 item sehingga juga terjadi 8 kali pembagian list menjadi dua bagian dan pada saat merging, terjadi 8 kali penggabungan dua bagian yang telah diurutkan. Hal ini menunjukkan bahwa jumlah splitting dan merging pada merge sort bergantung pada jumlah item pada list yang diurutkan.

6. Quick Short

a. Berikan tampilan output dari program diatas!

Tampilan output:

```
[17, 20, 26, 31, 44, 54, 55, 77, 93]
```

b. Berikan penjelasan ketiga fungsi dari program diatas!

Program di atas merupakan implementasi algoritma quick sort menggunakan rekursif. Terdapat tiga fungsi yang digunakan:

1. quickSort(alist): Fungsi ini merupakan wrapper function untuk quickSortHelper dan menerima argumen list alist yang akan diurutkan. Fungsi ini akan memanggil quickSortHelper dengan argumen alist, indeks pertama, dan indeks terakhir pada list.
2. quickSortHelper(alist, first, last): Fungsi ini merupakan fungsi utama yang melakukan rekursif dalam melakukan pengurutan. Fungsi ini menerima tiga argumen, yaitu alist yang merupakan list yang akan diurutkan, first yang merupakan indeks pertama pada list, dan last yang merupakan indeks terakhir pada list. Fungsi ini akan memanggil fungsi partition untuk membagi list menjadi dua bagian, kemudian memanggil dirinya sendiri untuk bagian kiri dan kanan dari splitpoint (hasil dari pemisahan list pada fungsi partition).
3. partition(alist, first, last): Fungsi ini digunakan untuk membagi list menjadi dua bagian, yaitu bagian yang lebih kecil dari pivotvalue dan bagian yang lebih besar dari pivotvalue. Fungsi ini menerima tiga argumen, yaitu alist yang merupakan list yang akan dipartisi, first yang merupakan indeks pertama pada list, dan

last yang merupakan indeks terakhir pada list. Fungsi ini akan memilih pivotvalue (biasanya diambil dari tengah list atau acak), kemudian membagi list menjadi dua bagian, yaitu bagian kiri dan kanan. Pada bagian kiri, akan ditemukan elemen yang lebih besar dari pivotvalue dan pada bagian kanan akan ditemukan elemen yang lebih kecil dari pivotvalue. Setelah itu, elemen yang lebih besar pada bagian kiri akan ditukar dengan elemen yang lebih kecil pada bagian kanan. Proses ini akan terus berlanjut hingga selesai dan nilai rightmark akan menjadi splitpoint yang nantinya akan digunakan pada fungsi quickSortHelper untuk membagi list menjadi dua bagian.

Kesimpulan:

Sorting adalah proses menempatkan elemen dari koleksi dalam beberapa jenis urutan. Misalnya, list kata dapat diurutkan menurut abjad atau panjangnya.. Terdapat beberapa teknik sorting yang umum digunakan, antara lain bubble sort, selection sort, insertion sort, shell sort, merge sort, dan quick sort.

1. Bubble Sort Bubble sort adalah teknik sorting yang paling sederhana dan mudah dipahami. Teknik ini bekerja dengan membandingkan dua nilai data yang berdekatan, kemudian menukar posisi apabila data pertama lebih besar dari data kedua. Bubble sort bekerja dengan iterasi melalui data hingga seluruh data terurut secara teratur. Kelemahan bubble sort adalah kurang efisien ketika mengurutkan data dalam jumlah yang besar.
2. Selection Sort Teknik selection sort memulai pengurutan dengan memilih elemen terkecil pada data dan menukar elemen tersebut dengan elemen pada posisi pertama. Kemudian, teknik ini mencari elemen terkecil kedua dan menukar elemen tersebut dengan elemen pada posisi kedua. Proses ini dilakukan terus menerus hingga seluruh data terurut secara teratur. Selection sort memiliki kelemahan yang sama dengan bubble sort, yaitu tidak efisien untuk mengurutkan data dalam jumlah yang besar.
3. Insertion Sort Teknik insertion sort bekerja dengan mengurutkan data satu per satu, dimulai dari elemen kedua hingga elemen terakhir. Teknik ini membandingkan nilai data yang akan diurutkan dengan data sebelumnya, dan memindahkan data ke posisi yang tepat agar data tersebut terurut. Insertion sort lebih efisien daripada bubble sort dan selection sort, namun masih terbatas dalam mengurutkan data dalam jumlah yang besar.
4. Shell Sort Shell sort adalah pengembangan dari teknik insertion sort. Teknik ini mengurutkan data dengan membandingkan elemen pada jarak tertentu, kemudian memindahkan elemen tersebut ke posisi yang benar. Setelah iterasi pertama selesai, jarak antar elemen dikurangi dan proses pengurutan diulang hingga seluruh data

terurut secara teratur. Shell sort lebih efisien daripada teknik insertion sort, namun masih kurang efisien ketika mengurutkan data dalam jumlah yang sangat besar.

5. Merge Sort Teknik merge sort membagi data menjadi dua bagian secara rekursif hingga hanya tersisa satu elemen pada setiap bagian. Kemudian, teknik ini melakukan pengurutan dan penggabungan kedua bagian tersebut hingga seluruh data terurut secara teratur. Merge sort merupakan teknik sorting yang efisien dan baik digunakan ketika mengurutkan data dalam jumlah yang besar.
6. Quick Sort Quick sort adalah teknik sorting yang bekerja dengan memilih suatu elemen pivot, kemudian membagi data menjadi dua bagian, satu bagian yang lebih kecil dari pivot dan satu bagian yang lebih besar dari pivot. Kemudian, teknik ini melakukan rekursi pada setiap bagian data dan memilih pivot pada setiap bagian hingga seluruh data terurut secara teratur. Quick sort adalah teknik sorting yang paling cepat dan efisien, namun membutuhkan kompleksitas pemrograman yang tinggi.