Universitas Padjadjaran

SORTING & SEARCHING

Mira Suryani, S.Pd., M.Kom
S-1 Teknik Informatika
Jatinangor, 15 November 2018

Tujuan

- Mahasiswa dapat menjelaskan pengertian array dan bisa menerangkan operasi dasar menggunakan array dengan benar
- Mahasiswa dapat : Mengoperasikan dan membuat program dari semua algoritma array dengan benar.

Pokok Bahasan

- **Sorting**
- **Searching**

Sorting

- Sorting bisa didefinisikan sebagai suatu pengurutan data yang sebelumnya tersusun secara acak menjadi data yang tersusun secara teratur menurut aturan tertentu.
- Urutan dari data yang kecil ke yang lebih besar (menaik) disebut dengan Ascending dan yang menurun disebut dengan Descending.

Bubble Sort

- Diberi nama "Bubble" karena proses pengurutan secara berangsur-angsur bergerak/berpindah ke posisi yang tepat , seperti gelembung yang keluar dari sebuah gelas bersoda.
- Bubble sort mengurutkan data dengan cara membandingkan elemen sekarang dengan elemen berikutnya.
- jika elemen sekarang lebih besar dari elemen berikutnya maka elemen tersebut ditukar (untuk pengurutan ascending)
- algoritma ini seolah olah menggeser satu per satu elemen dari kenan ke kiri atau kiri ke kanan, tergantung jenis pengurutannya.
- Ketika suatu tahap proses telah selesai, maka bubble sort akan mengalami proses selanjutnya, demikian seterusnya.
- Bubble sort berhenti jika seluruh array telah diperiksa dan tidak ada pertukaran lagi yang bisa dilakukan,serta tercapai pengurutan yang telah diinginkan

Contoh : Dari data 22 10 15 3 8 2

Proses 1 :

22	10	15	3	8	2
10	22	15	3	8	2
10	15	22	3	8	2
10	15	3	22	8	2
10	15	3	8	22	2
10	15	3	8	2	22

Pengecekan dimulai dari data yang paling awal, kemudian dibandingkan dengan data di depannya,jika data didepannya lebih besar maka akan di tukar.

Proses 2:

10	15	3	8	2	22
10	15	3	8	2	22
10	3	15	8	2	22
10	3	8	15	2	22
10	3	8	2	15	22

pengecekan dilakukan sampai dengan data yang kedua terakhir karena data terakhir pasti sudah paling besar.

Contoh : Dari data 22 10 15 3 8 2

Proses 3 :

10	3	8	2	15	22
3	10	8	2	15	22
3	8	10	2	15	22
3	8	2	10	15	22

Proses 4 :

3	8	2	10	15	22
3	8	2	10	15	22
3	2	8	10	15	22

Proses 5 :

3	2	8	10	15	22
2	3	8	10	15	22

Fungsi Bubble Sort

```
void swap (int& x, int& y){
     int temp = x;
     x = y;
     y = temp;
}

void bubbleSort(larik& a, int n) {
     for (int i=n-1; i > 0; i--){
          for (int j=0; j < i; j++) {
               if (a[j] > a[j+1] )
                    swap (a[j], a[j+1]);
          }
     }
}
```

- Tentukan algoritma 2 bentuk lain? (lihat diktat)

Bubble Sort (cara 2)

Dari data 22 10 15 3 8 2

- Proses 1 :

22	10	15	3	8	2	
22	10	15	3	2	8	
22	10	15	2	3	8	
22	10	2	15	3	8	
22	2	10	15	3	8	
2		22	10	15	3	8

- Pengecekan dimulai dari data yang paling akhir, kemudian dibandingkan dengan data di depannya,jika data didepannya lebih besar maka akan di tukar.

Bubble Sort (cara 2)

- Proses 2:
- 2 22 10 15 3 8
  2 22 10 15 3 8
  2 22 10 3 15 8
  2 22 3 10 15 8
  2 3 22 10 15 8

  pengecekan dilakukan sampai dengan data ke-2 karena data pertama pasti sudah paling kecil.

Bubble Sort (cara 2)

- Proses 3 :
- 2 3 22 10 15 8
  2 3 22 10 8 15
  2 3 22 8 10 15
  2 3 8 22 10 15

  Proses 4:
- 2 3 8 22 10 15
  2 3 8 22 15 10
  2 3 8 15 22 10

- Proses 5 :
- 2 3 8 15 22 10
  2 3 8 15 10 22

  Pengurutan berhenti.

Bubble Sort (cara 3)

- Proses 1 :

22	10	15	3	8	2
10	22	15	3	8	2
10	22	15	3	8	2
3	22	15	10	8	2
3	22	15	10	8	2
2	22	15	10	8	3

- Pengecekan dimulai dari data yang di posisi pertama, kemudian dibandingkan dengan data di depannya,jika data didepannya lebih besar maka akan di tukar.
- Selanjutnya pembandingan antara data yang posisi pertama dengan dengan data yang ketiga, keempat dstnya

Bubble Sort (cara 3)

- Proses 2:

2	22	15	10	8	3
2	15	22	10	8	3
2	10	22	15	8	3
2	8	22	15	10	3
2	3	22	15	10	8

- pengecekan dilakukan mulai yang kedua karena yang pertama sudah pasti yang terkecil, dengan data yang ke tiga dstnya.

Bubble Sort (cara 3)

- Proses 3:

2	3	22	15	10	8
2	3	15	22	10	8
2	3	10	22	15	8
2	3	8	22	15	10

- Proses 4:

2	3	8	22	15	10
2	3	8	15	22	10
2	3	8	10	22	15

- Proses 5:

2	3	8	10	22	15
2	3	8	10	15	22

Pengurutan berhenti.

Selection Sort

- Cara kerja metode ini didasarkan pada pencarian elemen dengan nilai terkecil. kemudian dilakukan penukaran dengan elemen ke-i.
- Secara singkat metode ini bisa dijelaskan sebagai berikut.
  - Pada langkah pertama, dicari data yang terkecil dari data pertama sampai terakhir. Kemudian data tersebut ditukar dari data pertama. Dengan demikian, data pertama sekarang mempunyai nilai paling kecil dibanding dengan data lain.
  - Pada langkah kedua, data terkecil kita cari mulai dari data kedua sampai data terakhir. Data terkecil yang kita peroleh kita tukar dengan data kedua.
  - Demikian seterusnya sampai seluruh data terurut.

Selection Sort

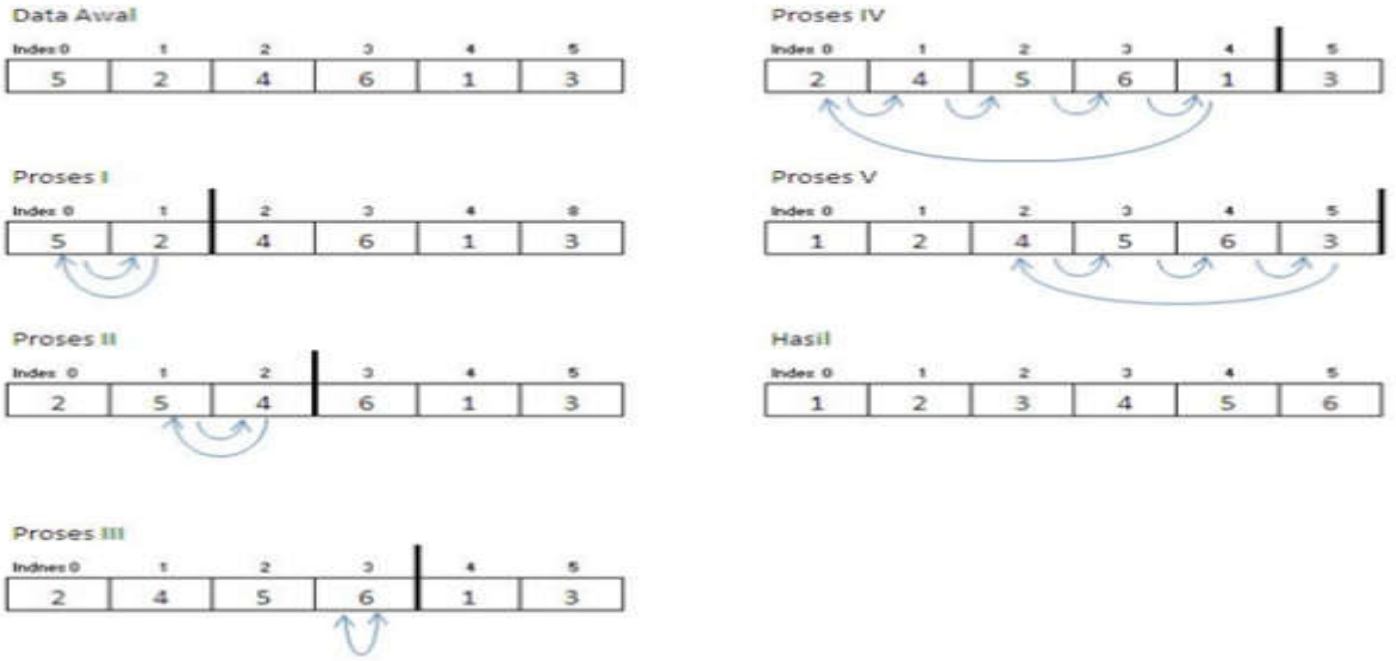
Iterasi ke-	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
awal	22	10	15	3	2	8
i = 0 , posisi = 4	2	10	15	3	22	8
i = 1, posisi = 3	2	3	15	10	22	8
i = 2, posisi = 5	2	3	8	10	22	15
i = 3, posisi = 3	2	3	8	10	22	15
i = 4, posisi = 5	2	3	8	10	15	22
akhir	2	3	8	10	15	22

Fungsi Selection Sort

```
void selectionSort(larik& a, int n) {
      int posisi;
      for (int i=0; i < n-1; i++){
            posisi = i;
            for (int j=i+1; j < n; j++) {
                  if (a[posisi] > a[j]) {
                        posisi = j;
                  }
            }
            swap (a[i], a[posisi]);
      }
}
```

Insertion Sort

- Insertion sort adalah sebuah metode pengurutan data dengan menempatkan setiap elemen data pada posisinya dengan cara melakukan perbandingan dengan data – data yang ada



Fungsi Insertion Sort

```
void insertionSort(larik& a, int n) {
     for (int i = 1; i < n; i++) {
        for (int j = i; j >= 1; j--){
            if (a[j] < a[j-1]) {
                swap(a[j], a[j-1]);
            }
            else
                break;
        }
    }
}
```

Shell Sort

- Metode ini dikembangkan oleh Donald L. Shell pada tahun 1959.
- Dalam metode ini jarak antara dua elemen yang dibandingkan dan ditukarkan tertentu.
- Secara singkat metode ini dijelaskan sebagai berikut:
  - Pada langkah pertama, ambil elemen pertama dan bandingkan dan dengan elemen pada jarak tertentu dari elemen pertama tersebut. Kemudian elemen kedua dibandingkan dengan elemen lain dengan jarak yang sama seperti jarak yang sama seperti diatas. Demikian seterusnya sampai seluruh elemen dibandingkan.
  - Pada langkah kedua proses diulang dengan langkah yang lebih kecil,
  - pada langkah ketiga jarak tersebut diperkecil lagi seluruh proses dihentikan jika jarak sudah sama dengan satu.

Proses Shell Sort

Jarak	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
awal	22	10	15	3	2	8
Jarak = 3	22	10	15	3	2	8
	3	10	15	22	2	8
	3	2	15	22	10	8
	3	2	8	22	10	15
Jarak = 1	3	2	8	22	10	15
	2	3	8	22	10	15
	2	3	8	22	10	15
	2	3	8	22	10	15
	2	3	8	10	22	15
	2	3	8	10	15	22
Akhir	2	3	8	10	15	22

Fungsi Shell Sort

```
void shellSort(larik& a, int n) {
   int j;

  //Pengurangan jarak / gap sebanyak 2
   for (int gap = n / 2; gap > 0; gap /= 2)  {
       for (int i = gap; i < n; i++) {
          int temp = a[i];
          for (j=i; j>=gap && temp<a[j - gap]; j -= gap){
                a[j] = a[j - gap];
          }
          a[j] = temp;
       }
    }
}
```

Searching (Pencarian)

- Searching adalah metode pencarian informasi dalam suatu aplikasi dengan suatu kunci(key).
- Pencarian diperlukan untuk mencari suatu informasi khusus dari kumpulan data pada saat lokasi yang pasti dari informasi tersebut sebelumnya tidak diketahui.
- Pencarian selalu dinyatakan dengan referensi pada adanya sekelompok data yang tersimpan secara terorganisasi.
- Pencarian akan dilakukan selama data yang dicari belum ditemukan dan data masih ada, sebaliknya pencarian akan selesai jika data ditemukan atau data sudah tidak ada lagi.

Pencarian sekuensial secara Linear (linear search)

- Data yang ada di bandingkan satu persatu secara berurutan dengan yang dicari.
- Pada dasarnya, pencarian ini hanya melakukan pengulangan dari 1 sampai dengan jumlah data.
- Pada setiap perulangan , di bandingkan data ke-i dengan yang dicari.
- Apabila sama, berarti data telah ditemukan .
- Sebaliknya apabila sampai akhir pengulangan , tidak ada data yang sama berarti data tidak ditemukan.

Fungsi Linear Search

```
main() {
   larik x;
   int n,kunci,found,lokasi;
   cout << "Kunci Pencarian data : " ; cin >> kunci;
   linearSearch(x, n, kunci, found, lokasi);
   if (found)
           cout << "Ditemukan di posisi : " << lokasi+1 ;
   else
            cout << "Tidak ditemukan";

}

void linearSearch(larik a, int n, int kunci, int& found, int& lokasi){
          found = lokasi = 0;
          while (!found && lokasi < n) {
                    if (a[lokasi] == kunci){
                              found = 1;
                    }
                    else {
                              lokasi=lokasi+1;
                    }
          }
}
```

- Kelebihan :
  - Relatif lebih cepat dan efisien untuk data yang terbatas
  - Algoritma sederhana
- Kekurangan :
  - Kurang cepat untuk data dalam jumlah besar
  - Beban komputasi cenderung lebih besar

Binary Search

- Salah satu syarat pencarian biner (binary search) dapat dilakukan adalah data sudah dalam keadaan terurut. Dengan kata lain, apabila data belum dalam keadaan terurut , pencarian biner tidak dapat dilakukan .
- Langkah dalam pencarian biner adalah :
  - Mula-mula diambil dari posisi awal=1 dan posisi akhir = n
  - cari posisi data tengah dengan rumus posisi tengah = (posisi awal + posisi akhir) div 2
  - data yang di cari dibandingkan dengan data tengah:
    - Jika sama, data ditemukan, Proses selesai
    - Jika lebih kecil, proses dilakukan kembali tetapi posisi akhir dianggap sama dengan posisi tengah - 1,
    - Jika lebih besar , proses dilakukan kembali tetapi posisi awal dianggap sama dengan posisi tengah + 1.
  - Ulangi langkah kedua hingga data ditemukan , atau tidak ditemukan.
  - Pencarian biner ini akan berakhir jika data ditemukan posisi awal lebih besar dari pada posisi akhir. Jika posisi awal sudah lebih besar dari posisi akhir berarti data tidak diketemukan.

Fungsi Binary Search

```
void binarySearch(larik a, int n, int kunci, int& found, int& lokasi){
        int left = 0, right = n-1;
        found =  0;
        while (!found && left <= right) {
                lokasi = (left + right) / 2;  // titik tengah
                if (a[lokasi] == kunci){
                        found = 1;
                }
                else
                        if (a[lokasi] < kunci){
                                left = lokasi + 1;
                        }
                        else {
                                right = lokasi - 1;

                        }
        }
}
```

- Kelebihannya :
  - Untuk data dalam jumlah besar, waktu searching lebih cepat
  - Beban komputasi lebih kecil
- Kekuranganya :
  - Data harus sudah di-sorting lebih dulu (dalam keadaan terurut)

Latihan dan Tugas

1. Buatlah program modular Matriks berikut:
   a. Penjumlahan 2 buah matriks
   b. Perkalian 2 buah matriks
   c. Transpose suatu matriks
   d. Mencari jumlah dari setiap baris dan kolom suatu matriks. Hasil penjumlahan disimpan dalam suatu array 1 dimensi

      Contoh :

A =	1	0	4	jBaris =	5	jKolom =	8
	5	2	1		8		3
	2	1	2		5		7

Fungsi yang diperlukan antara lain : main, inputMatriks, cetakMatriks, cariJumlahBaris, cariJumlah Kolom, cetakLarik.

2. Buatlah program modular untuk mencari nilai standart deviasi dari kumpulan data bertipe integer


ANY
QUESTIONS?

Sesi Berakhir
TERIMA KASIH