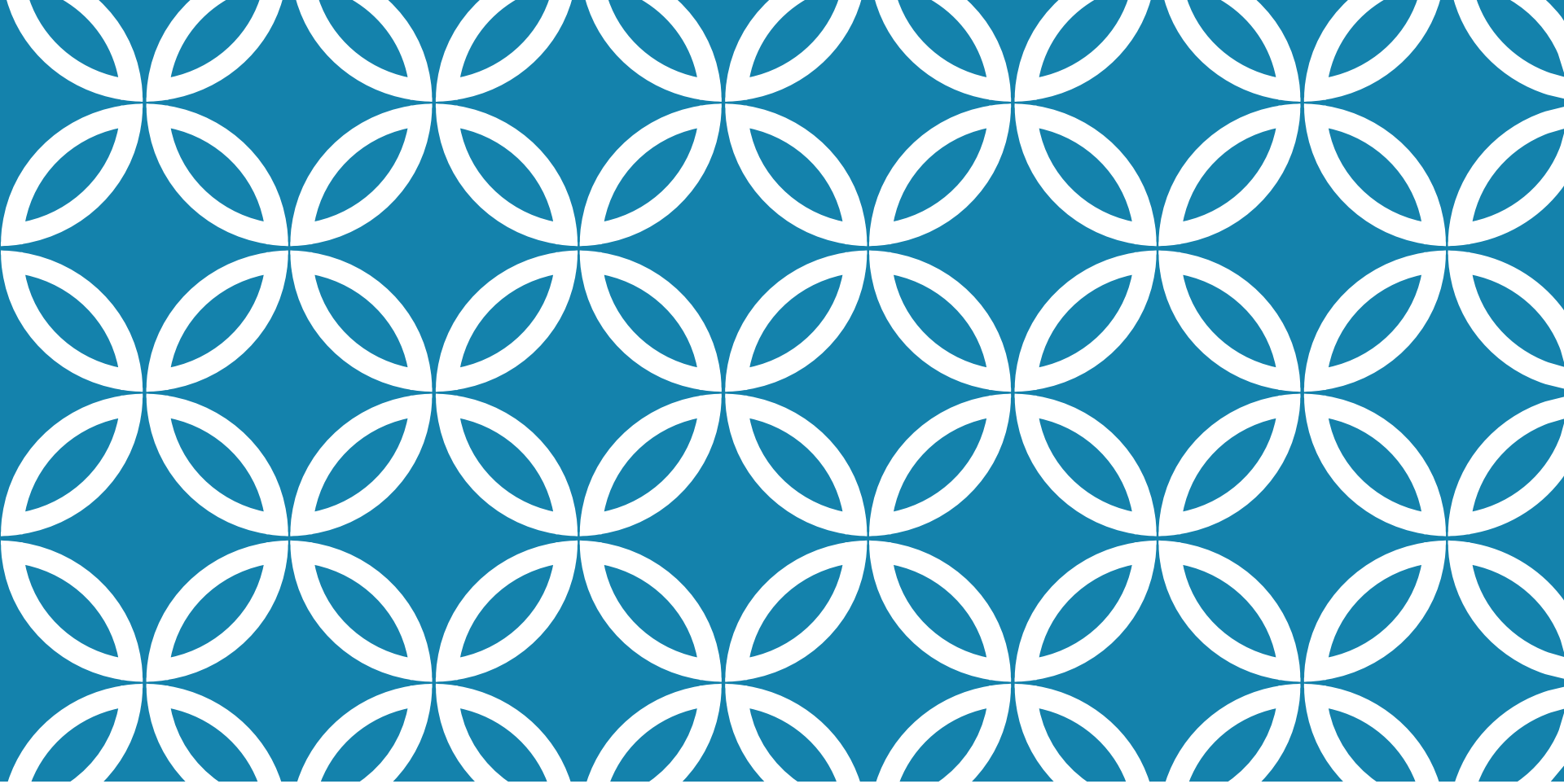


ARRAY STATIS

OPERASI PADA ARRAY

Pencariaan

- Proses mencari suatu data yang terdapat dalam suatu array. Proses ini menghasilkan nilai benar atau salah.
- Metode Pencarian:
  - Sequential / Linear Search
  - Binary Search

OPERASI PADA ARRAY

Sequential / Linear Search:

- Tanpa Boolean
  - Tanpa Sentinel
  - Dengan Sentinel
- Dengan Boolean

SEQUENTIAL SEARCHING

Tanpa boolean dengan sentinel:

1. Tidak menggunakan variabel boolean.
2. Mempunyai tambahan elemen di akhir array untuk menyimpan data cari apabila data cari tidak ditemukan.

CONTOH

```
Procedure SeqSearchWithSentinel(input L : LarikInt, input n:integer, Input
x :integer,output idx : Integer)
{ I.S: x dan elemen-elemen larik L[1..N] sudah terdefinisi nilainya.}
{ F.S: idx berisi indeks larik L yang berisi nilai x.     Jika x tidak
ditemukan, maka idx diisi dengan nilai -1. }

KAMUS :
    i : integer   { pencatat indeks larik }

 ALGORITMA:
   L[n + 1] ← x    { sentinel }
    i ←1
   while (L[i] ≠ x) do
     i ← i + 1
   endwhile    { L[i] = x }   { Pencarian selalu berhasil menemukan x }
    { Kita harus menyimpulkan apakah x ditemukan pada elemen sentinel atau
bukan }

   if idx = n + 1 then  { x ditemukan pada elemen sentinel }
      idx ← -1   { berarti x belum ada pada larik L semula }
   else    { x ditemukan pada indeks < n + 1 }
      idx ← i
   endif
```

SEQUENTIAL SEARCHING

Tanpa boolean tanpa sentinel:

1. Tidak menggunakan variabel boolean.
2. Tidak mempunyai tambahan elemen di akhir array.

SEQUENTIAL SEARCH

Dengan boolean:

1. Menggunakan variabel boolean.

2. Menghasilkan nilai **TRUE** atau **FALSE** di akhir pencarian.

BINARY SEARCH

- Data yang disimpan di dalam larik harus sudah terurut, misalkan elemen array sudah terurut menurun.
- Pada proses pencarian, diperlukan dua buah indeks array, yaitu indeks terkecil dan indeks terbesar.
- Indeks terkecil sebagai indek ujung kiri larik dan indeks terbesar sebagai indeks ujung kanan larik.
- Misalkan indeks kiri adalah *i dan indeks kanan adalah j. Pada mulanya, kita inisialisasi i dengan 1 dan j dengan n.*

BINARY SEARCH

Langkah 1:

- **Bagi dua elemen larik pada elemen tengah. Elemen tengah adalah elemen dengan indeks *k = (i + j) div 2*.**
- (Elemen tengah, *L[k], membagi larik menjadi dua bagian, yaitu bagian kiri L[i..j] dan bagian kanan L[k+1..j])*

Langkah 2:

- **Periksa apakah *L[k] = x*. *Jika L[k] = x, pencarian selesai sebab x*** sudah ditemukan. Tetapi, jika *L[k] „ x, harus ditentukan apakah pencarian akan dilakukan di larik bagian kiri atau di bagian kanan.*
- Jika *L[k] < x, maka pencarian dilakukan lagi pada larik bagian kiri. Sebaliknya, jika L[k] > x, pencarian dilakukan lagi pada larik bagian kanan.*

Langkah 3:

- **Ulangi Langkah 1 hingga *x ditemukan atau i > j (yaitu, ukuran larik*** sudah nol!)

ILUSTRASI PENCARIAN BAGIDUA

Misalkan diberikan larik *L dengan delapan buah elemen yang sudah terurut menurun seperti di bawah ini:*

81	76	21	18	16	13	10	7
i =1	2	3	4	5	6	7	8=*j*

Misalkan elemen yang dicari adalah *x = 16*.

Langkah 1: *i = 1 dan j = 8 Indeks elemen tengah k = (1 + 8) div 2 = 4 (elemen yang diarsir)*

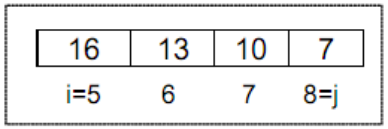
81	76	21	18	16	13	10	7
1	2	3	4	5	6	7	8

kiri kanan

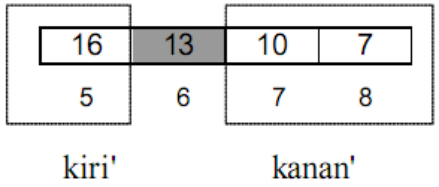
ILUSTRASI PENCARIAN BAGIDUA

Langkah 2:

- **Pembandingan:** ***L[4] = 16? Tidak! Harus diputuskan apakah*** pencarian akan dilakukan di bagian kiri atau di bagian kanan dengan pemeriksaan sebagai berikut:
- *Pembandingan: L[4] > 16? Ya! Lakukan pencarian pada larik bagian* kanan dengan *i = k + 1 = 5 dan j = 8 (tetap)*

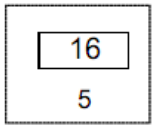


- **Langkah 1':**
  - *i = 5 dan j = 8 Indeks elemen tengah k = (5 + 8) div 2 = 6 (elemen yang diarsir)*

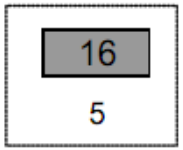


ILUSTRASI PENCARIAN BAGIDUA

- **Langkah 2':**
  - ***Pembandingan: L[6] = 16? Tidak! Harus diputuskan apakah*** pencarian akan dilakukan di bagian kiri atau di bagian kanan dengan pemeriksaan sebagai berikut:
  - *Pembandingan: L[6] > 16? Tidak! Lakukan pencarian pada larik bagian kiri dengan i = 5 (tetap) dan j = k - 1 = 5*



  - **Langkah 1":**
    - *i = 5 dan j = 5 Indeks elemen tengah k = (5 + 5) div 2 = 5 (elemen yang diarsir)*



  - **Langkah 2":**
    - *L[5] = 16? Ya! (x ditemukan, proses pencarian selesai)*

PROCEDURE BINARY SEARCH UNTUK DATA TERURUT MENURUN

```
procedure BinarySearch1(input L : LarikInt, input n : integer,  input x : integer,
output idx : integer)

KAMUS
  i, j : integer      { indeks kiri dan indek kanan larik }
  k : integer          { indeks elemen tengah}   ketemu : boolean    { flag untuk
menentukan apakah X ditemukan }

 ALGORITMA:
   i ← 1       { ujung kiri larik }
   j ← n       { ujung kanan larik }
  ketemu  ← false   { asumsikan x belum ditemukan }
while (not ketemu) and (i <= j) do
 k ← (i + j) div 2  {bagidua larik L pada posisi k}
   if (L[k] = x) then
        ketemu ← true
   else  { L[k] != x }
      if (L[k] > x) then { Lakukan pencarian pada larik bagian kanan, set indeks
ujung    kiri larik yang baru }
          i ← k + 1
      else            { Lakukan pencarian pada larik bagian kiri, set indeks ujung
kanan larik yang baru }
          j ← k - 1
      endif
   endif
endwhile   { ketemu = true or i > j}
 if ketemu then  { x ditemukan }
    idx ← k
else            { x tidak ditemukan di dalam larik  }
    idx ←-1
   endif
```

OPERASI ARRAY STATIS

4. Pengurutan (Sorting)
    a. Bubble Sort
    b. Selection Sort
    c. Insertion Sort
    d. Radix Sort
    e. Merge Sort
    f. Quick Sort.

BUBBLE SORT

Perbandingan data dilakukan dari posisipertama atau posisi terakhir bergeser satu persatu sampai semua data dibandingkan. Jika terdapat N data dan data terkoleksi dari urutan 0 sampai dengan N-1 maka algoritma pengurutan dengan metode bubble sort adalah sebagai berikut:

1. Bandingkan posisi data i = 0 dan j = 1.
2. Jika data diposisi I lebih besar daripada data diposisi j, maka data diposisi iditukar dengan data diposisi j (swap). Jika tidak penukaran posisi tidak dilakukan.
3. Kemudian, lakukan perbandingan data diposisi i= 1 dan data diposisi j = 2.Lakukan langkah 2, begitu juga untuk data berikutnya hingga i= N-2 dan j= N-1.
4. Ulangi langkah 1, 2 dan 3 untuk data diposisi 0 sampai dengan data diposisi N-2, karena data di posisi N-1 adalah data dengan nilai terbesar. Untuk tahap selanjutnya data yang dibandingkan akan semakin berkurang sebab data dengan nilai yang lebih besar akan terposisi dibagian sebelah kanan data.

CONTOH BUBBLE SORT

Array Awal:	5	3	7	9	2	3	6	4	3	1

L. 1	3	5	7	2	3	6	4	3	1	9
L. 2	3	5	2	3	6	4	3	1	7	9
L. 3	3	2	3	5	4	3	1	6	7	9
L. 4	2	3	3	4	3	1	5	6	7	9
L. 5	2	3	3	3	1	4	5	6	7	9
L. 6	2	3	3	1	3	4	5	6	7	9
L. 7	2	3	1	3	3	4	5	6	7	9
L. 8	2	1	3	3	3	4	5	6	7	9
L. 9	1	2	3	3	3	4	5	6	7	9

ALGORIMTA BUBBLE SORT

```
Procedure bubble_sort(y : array[1..5] of integer,
n:integer)
Kamus :
       temp, i, j : integer
Algoritma :
       for i = n down to 1
              for j = 1 to j = i
                     if y[j] > y[j+1] then
                       temp ← y[j]
                       y[j] ← y[j+i]
                       y[j+1] ← temp
                                end if
                       end for
                end for
```

SELECTION SORT

Metode pengurutan dengan mencari nilai data terkecil dimulai dari data diposisi 0 hingga diposisi N-1.

Jika terdapat N data dan data terkoleksi dari urutan 0 sampai dengan N-1 maka algoritma pengurutan dengan metode selection sort adalah sebagai berikut:

- Cari data terkecil dalam interval j= 0 sampai dengan j= N-1
- Jika pada posisi pos ditemukan data yang terkecil, tukarkan data diposisi pos dengan data di posisi i
- Ulangi langkah 1 dan 2 dengan j=j+I sampai dengan j= N-1, dan seterusnya sampai j = N - 1

CONTOH SELECTION SORT

Bila diketahui data awal berupa: 44 55 12 42 94 18 6 67, maka langkah per langkah pengurutan dengan metode selection sort adalah sebagai berikut:

44	**55**	**12**	**42**	**94**	**18**	**06**	**67**	**Data Awal**
06	55	12	42	94	18	44	67	Tukarkan data ke 1 dengan data ke 7
06	12	55	42	94	18	44	67	Tukarkan data ke 2 dengan data ke 3
06	12	18	42	94	55	44	67	Tukarkan data ke 3 dengan data ke 6
06	12	18	42	94	55	44	67	Data ke 4 tidak ditukarkan
06	12	18	42	44	55	94	67	Data ke 5 ditukarkan dengan data ke 7
06	12	18	42	44	55	94	67	Data ke 6 tidak ditukarkan
06	12	18	42	44	55	67	94	Data ke 7 ditukarkan dengan data ke 8
06	**12**	**18**	**42**	**44**	**55**	**67**	**94**	**Data setelah terurut**

LATIHAN

Buat algoritma (pseudocode), programnya (source code) dan screenshot running program selection sort !