

# Searching

(sequential & binary search)

*Umi Sa'adah, S.Kom*

Politeknik Elektronika Negeri Surabaya  
e-mail : [umi@eepis-its.edu](mailto:umi@eepis-its.edu)

# Overview

- ❖ Konsep & Istilah
- ❖ Sequential Search
- ❖ Binary Search
- ❖ Perbandingan Unjuk Kerja (*Performance*)
- ❖ Review







## *Konsep dan Istilah*

- ❖ Internal Search – algoritma pencarian yang dilakukan dalam main memory komputer
- ❖ External Search – algoritma pencarian yang melibatkan external media menambah main memory

## *Konsep dan Istilah*

- ❖ Key – sebuah subset dari isi sebuah data yang digunakan untuk perbandingan selama proses pencarian
- ❖ Big-O Notation – notasi yang digunakan untuk mengindikasikan kenaikan (*Order of growth*) unjuk kerja dari sebuah algoritma searching

# Search

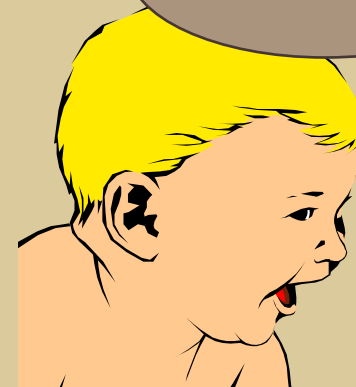
| [ 0 ]   | [ 1 ]   | [ 2 ]   | [ 3 ]   | [ 4 ]   | ... | [ 700 ]   |
|---|---|---|---|---|-----|---|
| Number 701466868<br> | Number 281942902<br> | Number 233667136<br> | Number 580625685<br> | Number 506643548<br> | ... | Number 155778322<br> |

Setiap record dalam list memiliki sebuah associated key.

Pada contoh ini, key-nya = ID numbers.

Berikan sebuah key, bagaimana cara menemukan recordnya dari list secara efektif ?

Number 580625685





# *Sequential Search*

- ❖ Begin at the beginning (or the end)
- ❖ Cek seluruh record dalam array atau list, baca satu persatu.
- ❖ Temukan record sesuai dengan key yang dicari
- ❖ Proses Searching berhenti karena salah satu alasan
  - Success – Found the target key
  - End of List – No more records to compare
- ❖ Diaplikasikan pada Array (sorted & unsorted) atau Linked List

# *Sequential Search - Unsorted*

## The List

|         |
|---------|
| Gordon  |
| Andrew  |
| Michael |
| Bill    |
| Scott   |
| Luke    |
| Adrian  |
| Dennis  |

## Target Key

Scott

Compare Target Key with

List[0] Gordon

List[1] Andrew

List[2] Michael

List[3] Bill

List[4] Scott – SUCCESS!

5 comparisons required

# *Sequential Search - Sorted*

## The List

|         |
|---------|
| Adrian  |
| Andrew  |
| Bill    |
| Dennis  |
| Gordon  |
| Luke    |
| Michael |
| Scott   |

## Target Key

Jeff

Compare Target Key with

List[0] Adrian

List[1] Andrew

List[2] Bill

List[3] Dennis

List[4] Gordon

List[5] Luke – Not Found!

6 comparisons required



# *Sequential Search Analysis*

- ❖ What are the worst and average case running times for sequential search?
- ❖ We must determine the O-notation for the number of operations required in search.
- ❖ Jumlah operasi pencarian bergantung pada nilai  $n$ , yaitu jumlah elemen dalam list

## *Worst Case Time for Sequential Search*

- ❖ Untuk sebuah array dengan  $n$  elemen, maka worst case time untuk sequential search  $\rightarrow$  requires  $n$  array accesses:  $O(n)$ .
- ❖ Kondisi yang mengharuskan pengecekan terhadap semua elemen array ( $n$  record) adalah:
  - Record yang dicari berada pada posisi terakhir dari array
  - Setelah pengecekan seluruh elemen array, ternyata record yang dicari tidak berhasil ditemukan dalam array tersebut

# *Algoritma Sequential Search*

1.  $i \leftarrow 0$
2.  $\text{ketemu} \leftarrow \text{false}$
3. Selama (tidak ketemu) dan  $(i < N)$  kerjakan baris 4
4. Jika  $(\text{Data}[i] = \text{key})$  maka  
     $\text{ketemu} \leftarrow \text{true}$   
    jika tidak  
         $i \leftarrow i + 1$
5. Jika (ketemu) maka  
     $i$  adalah indeks dari data yang dicari  
    jika tidak  
        data tidak ditemukan

# *Binary Search*

- ❖ Define working range as entire list
- ❖ Repeat till done
  - Select the middle record
  - Compare the target key value with the key of the selected “record”
  - Comparison results:
    - $\text{Key} < \text{middle record}$  : Range = First half
    - $\text{Key} > \text{middle record}$  : Range = Last half
    - $\text{Key} = \text{middle record}$ : Success, Done
- ❖ Applies **only to Sorted Array List**

# *Binary Search*

The List

|         |
|---------|
| Adrian  |
| Andrew  |
| Bill    |
| Gordon  |
| Luke    |
| Michael |
| Scott   |

Target Key

Bill

Compare Target Key with  
List[3] Gordon (high)  
List[1] Andrew (low)  
List[2] Bill (match)

3 comparisons required



# *Binary Search*

Example: sorted array of integer keys. Target=7.

| [ 0 ] | [ 1 ] | [ 2 ] | [ 3 ] | [ 4 ] | [ 5 ] | [ 6 ] |
|-------|-------|-------|-------|-------|-------|-------|
| 3     | 6     | 7     | 11    | 32    | 33    | 53    |

# *Binary Search*

Example: sorted array of integer keys. Target=7.

| [ 0 ] | [ 1 ] | [ 2 ] | [ 3 ] | [ 4 ] | [ 5 ] | [ 6 ] |
|-------|-------|-------|-------|-------|-------|-------|
| 3     | 6     | 7     | 11    | 32    | 33    | 53    |



Find approximate midpoint

# *Binary Search*

Example: sorted array of integer keys. Target=7.

| [ 0 ] | [ 1 ] | [ 2 ] | [ 3 ] | [ 4 ] | [ 5 ] | [ 6 ] |
|-------|-------|-------|-------|-------|-------|-------|
| 3     | 6     | 7     | 11    | 32    | 33    | 53    |



Is 7 = midpoint key? NO.

# *Binary Search*

Example: sorted array of integer keys. Target=7.

| [ 0 ] | [ 1 ] | [ 2 ] | [ 3 ] | [ 4 ] | [ 5 ] | [ 6 ] |
|-------|-------|-------|-------|-------|-------|-------|
| 3     | 6     | 7     | 11    | 32    | 33    | 53    |




Is  $7 < \text{midpoint key}$ ? YES.

# *Binary Search*

Example: sorted array of integer keys. Target=7.

| [ 0 ] | [ 1 ] | [ 2 ] | [ 3 ] | [ 4 ] | [ 5 ] | [ 6 ] |
|-------|-------|-------|-------|-------|-------|-------|
| 3     | 6     | 7     | 11    | 32    | 33    | 53    |



Search for the target in the area before midpoint.



# *Binary Search*

Example: sorted array of integer keys. Target=7.

| [ 0 ] | [ 1 ] | [ 2 ] | [ 3 ] | [ 4 ] | [ 5 ] | [ 6 ] |
|-------|-------|-------|-------|-------|-------|-------|
| 3     | 6     | 7     | 11    | 32    | 33    | 53    |



Find approximate midpoint

# *Binary Search*

Example: sorted array of integer keys. Target=7.

| [ 0 ] | [ 1 ] | [ 2 ] | [ 3 ] | [ 4 ] | [ 5 ] | [ 6 ] |
|-------|-------|-------|-------|-------|-------|-------|
| 3     | 6     | 7     | 11    | 32    | 33    | 53    |



Target = key of midpoint? NO.

# *Binary Search*

Example: sorted array of integer keys. Target=7.

| [ 0 ] | [ 1 ] | [ 2 ] | [ 3 ] | [ 4 ] | [ 5 ] | [ 6 ] |
|-------|-------|-------|-------|-------|-------|-------|
| 3     | 6     | 7     | 11    | 32    | 33    | 53    |



Target < key of midpoint? NO.

# *Binary Search*

Example: sorted array of integer keys. Target=7.

| [ 0 ] | [ 1 ] | [ 2 ] | [ 3 ] | [ 4 ] | [ 5 ] | [ 6 ] |
|-------|-------|-------|-------|-------|-------|-------|
| 3     | 6     | 7     | 11    | 32    | 33    | 53    |



Target > key of midpoint? YES.

# *Binary Search*

Example: sorted array of integer keys. Target=7.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] |
|-----|-----|-----|-----|-----|-----|-----|
| 3   | 6   | 7   | 11  | 32  | 33  | 53  |



Search for the target in the area after midpoint.



# *Binary Search*

Example: sorted array of integer keys. Target=7.

| [ 0 ] | [ 1 ] | [ 2 ] | [ 3 ] | [ 4 ] | [ 5 ] | [ 6 ] |
|-------|-------|-------|-------|-------|-------|-------|
| 3     | 6     | 7     | 11    | 32    | 33    | 53    |



Find approximate midpoint.

Is target = midpoint key? YES

# *Algoritma Binary Search*

1.  $L \leftarrow 0$
2.  $R \leftarrow N - 1$
3. ketemu  $\leftarrow$  false
4. Selama ( $L \leq R$ ) dan (tidak ketemu) kerjakan baris 5 sampai dengan 8
5.  $m \leftarrow (L + R) / 2$
6. Jika ( $\text{Data}[m] = \text{key}$ ) maka ketemu  $\leftarrow$  true
7. Jika ( $\text{key} < \text{Data}[m]$ ) maka  $R \leftarrow m - 1$
8. Jika ( $\text{key} > \text{Data}[m]$ ) maka  $L \leftarrow m + 1$
9. Jika (ketemu)

    maka m adalah indeks dari data yang dicari  
jika tidak  
    data tidak ditemukan

# *Performance Comparisons*

| Search Algorithm               | List Size:<br>100<br>elements             | List Size:<br>10,000<br>elements                 | List Size:<br>1,000,000<br>elements                 |
|--------------------------------|---|--|---|
| Sequential Search<br>(Average) | <b>50</b>                                 | <b>5,000</b>                                     | <b>500,000</b>                                      |
| Binary Search<br>(Maximum)     | <b>7</b><br><b><math>2^7 = 128</math></b> | <b>14</b><br><b><math>2^{14} = 16,384</math></b> | <b>20</b><br><b><math>2^{20} = 1,048,576</math></b> |