

Algoritma dan Struktur Data

Bubble Sort

Umi Sa'adah

Tita Karlita

Entin Martiana Kusumaningtyas

Arna Fariza

2021



Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

Sorting Algorithms

1. Selection
2. Insertion
3. Bubble
4. Shell
5. Merge
6. Quick

Bubble Sort

- Metode gelembung (*bubble sort*) disebut dengan metode penukaran (*exchange sort*)
- *Bubble sort* adalah metode yang mengurutkan data dengan cara **membandingkan** masing-masing elemen, kemudian melakukan **penukaran** bila perlu.
- Metode ini mudah dipahami dan diprogram, tetapi bila dibandingkan dengan metode lain yang kita pelajari, metode ini merupakan metode yang **paling tidak efisien**.



Bubble Sort

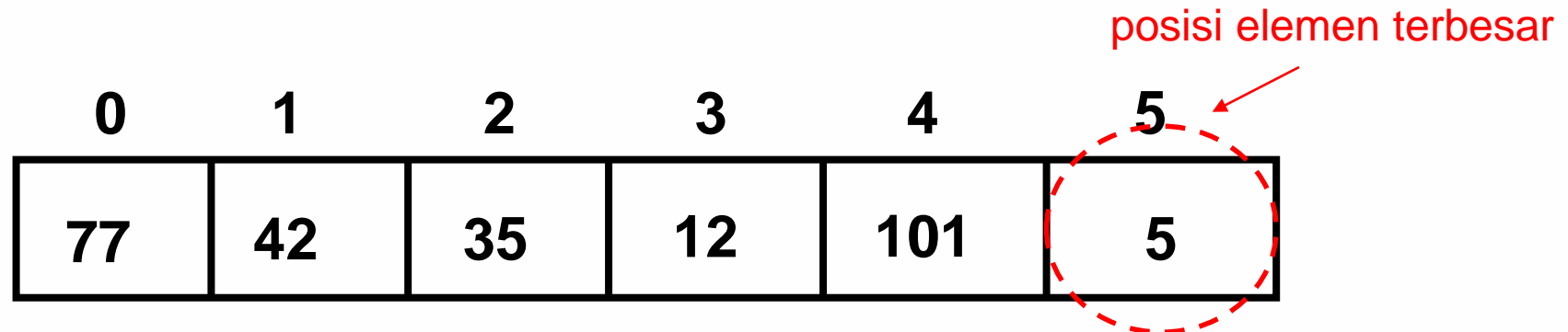
- Algoritma pengurutan dengan cara melakukan penukaran data dengan tepat disebelahnya secara terus menerus sampai bisa dipastikan dalam satu iterasi tertentu tidak ada lagi perubahan.
- Jika tidak ada perubahan berarti data sudah terurut.
- Disebut pengurutan gelembung karena masing-masing kunci akan dengan lambat menggelembung ke posisinya yang tepat.

"Bubbling Up" the Largest Element

- Memindahkan sekumpulan elemen
 - Memindahkan elemen dari posisi awal ke akhir (asc: posisi bagian kanan)
 - "Menggelembungkan" elemen terbesar ke bagian akhir menggunakan metode perbandingan sepasang (pair-wise comparisons) dan penukaran (swapping)

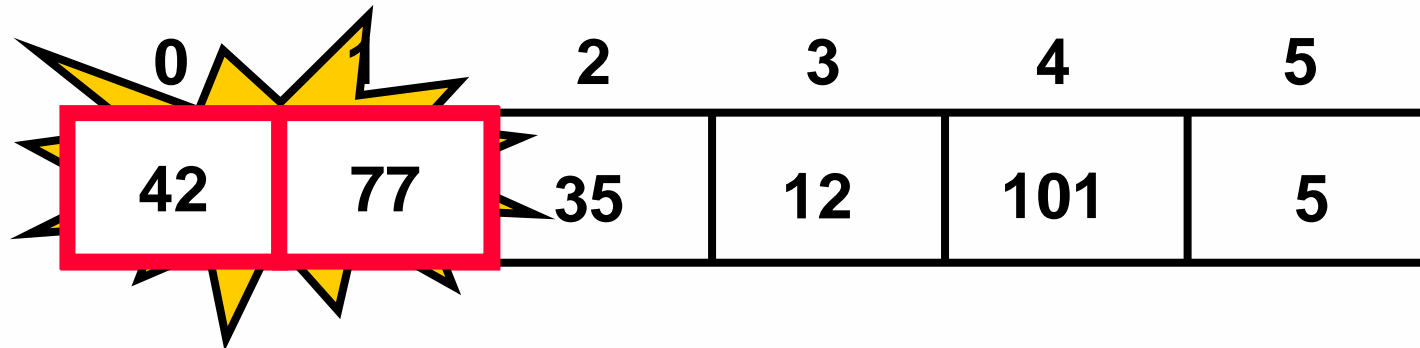
posisi elemen terbesar

0	1	2	3	4	5
77	42	35	12	101	5



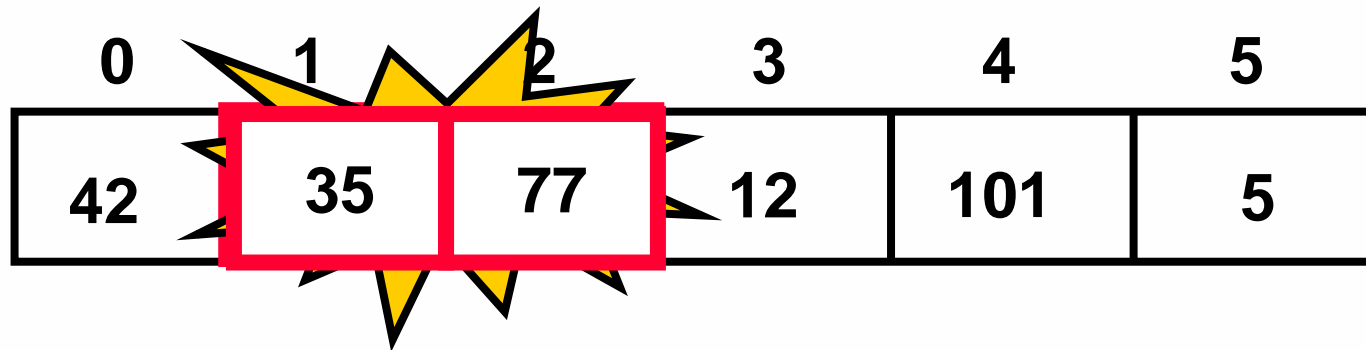
"Bubbling Up" the Largest Element

- Memindahkan sekumpulan elemen
 - Memindahkan elemen dari posisi awal ke akhir (asc: posisi bagian kanan)
 - "Menggelembungkan" elemen terbesar ke bagian akhir menggunakan metode perbandingan sepasang (pair-wise comparisons) dan penukaran (swapping)



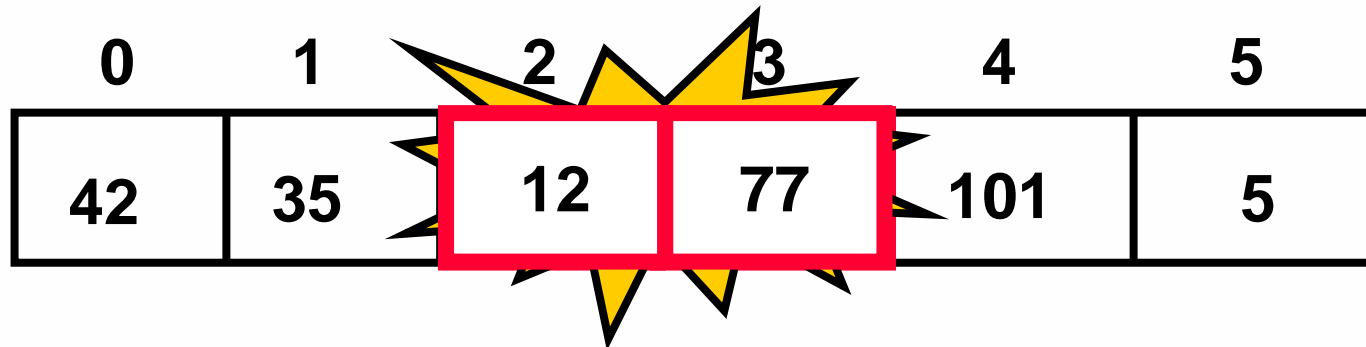
"Bubbling Up" the Largest Element

- Memindahkan sekumpulan elemen
 - Memindahkan elemen dari posisi awal ke akhir (asc: posisi bagian kanan)
 - "Menggelembungkan" elemen terbesar ke bagian akhir menggunakan metode perbandingan sepasang (pair-wise comparisons) dan penukaran (swapping)



"Bubbling Up" the Largest Element

- Memindahkan sekumpulan elemen
 - Memindahkan elemen dari posisi awal ke akhir (asc: posisi bagian kanan)
 - "Menggelembungkan" elemen terbesar ke bagian akhir menggunakan metode perbandingan sepasang (pair-wise comparisons) dan penukaran (swapping)



"Bubbling Up" the Largest Element

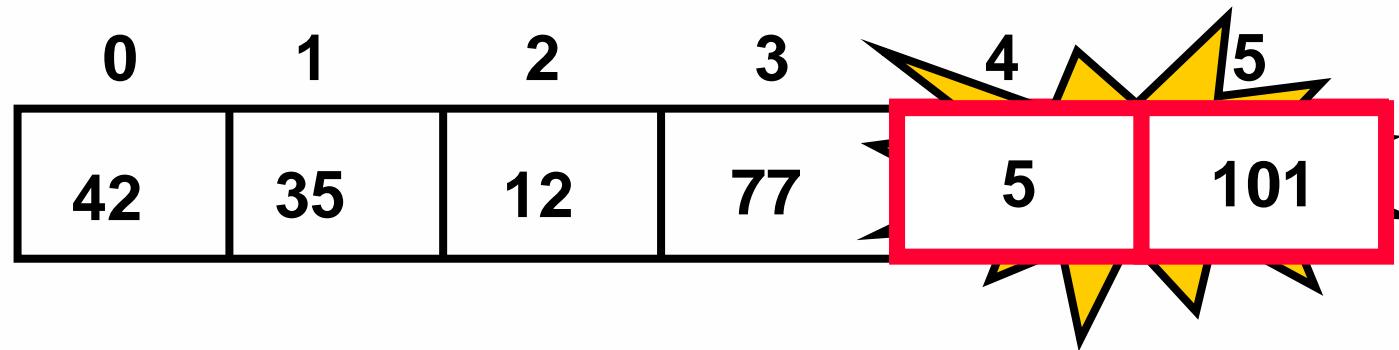
- Memindahkan sekumpulan elemen
 - Memindahkan elemen dari posisi awal ke akhir (asc: posisi bagian kanan)
 - "Menggelembungkan" elemen terbesar ke bagian akhir menggunakan metode perbandingan sepasang (pair-wise comparisons) dan penukaran (swapping)

0	1	2	3	4	5
42	35	12	77	101	5

No need to swap

"Bubbling Up" the Largest Element

- Memindahkan sekumpulan elemen
 - Memindahkan elemen dari posisi awal ke akhir (asc: posisi bagian kanan)
 - "Menggelembungkan" elemen terbesar ke bagian akhir menggunakan metode perbandingan sepasang (pair-wise comparisons) dan penukaran (swapping)



"Bubbling Up" the Largest Element

- Memindahkan sekumpulan elemen
 - Memindahkan elemen dari posisi awal ke akhir (asc: posisi bagian kanan)
 - "Menggelembungkan" elemen terbesar ke bagian akhir menggunakan metode perbandingan sepasang (pair-wise comparisons) dan penukaran (swapping)

0	1	2	3	4	5
42	35	12	77	5	101

Setelah 1x iterasi: elemen terbesar telah menempati posisinya.
Lakukan sejumlah iterasi lagi dengan cara yang sama untuk menempatkan elemen-elemen pada tempatnya.

Algoritma Bubble Sort

(untuk satu kali iterasi)

1. $\text{index} \leftarrow 0$
2. $\text{pos_akhir} \leftarrow n-2$
3. selama $\text{index} \leq \text{pos_akhir}$ do baris 4 sampai dengan 7
4. Jika $A[\text{index}] > A[\text{index}+1]$
5. $\text{Swap}(A[\text{index}], A[\text{index}+1])$
6. $\text{index} \leftarrow \text{index} + 1$
7. End baris 4

Pseudocode Bubble Sort

(untuk satu kali iterasi)

1. **PROCEDURE** bubbleSort(A : array data, N: jml data)
2. **REPEAT**
3. **FOR** j = 0 TO N - 1 **INCLUSIVE DO:**
4. **IF** A[j] > A[j+1] **THEN**
5. swap(A[j], A[j+1])
6. **END IF**
7. **END FOR**
8. **END PROCEDURE**

Yang perlu diperhatikan....

- Perhatikan bahwa dalam satu kali iterasi hanya satu elemen terbesar yang sudah menempati posisinya.
- Sebagian elemen yang lain masih belum terurutkan.
- Sehingga kita perlu **mengulang proses ini**.

0	1	2	3	4	5
42	35	12	77	5	101

Nilai terbesar telah menempati posisinya

Repeat “Bubble Up” How Many Times?

- Jika kita punya N elemen...
- Dan jika setiap kali kita menggelembung kan sebuah elemen, kita menempatkannya pada posisi yang tepat...
- Berarti, kita mengulang proses “bubble up” sebanyak $N - 1$ kali.
- Hal ini menjamin kita akan menempatkan seluruh N elemen secara tepat.

“Bubbling” All the Elements

1-5

0	1	2	3	4	5
42	35	12	77	5	101

0	1	2	3	4	5
35	12	42	5	77	101

0	1	2	3	4	5
12	35	5	42	77	101

0	1	2	3	4	5
12	5	35	42	77	101

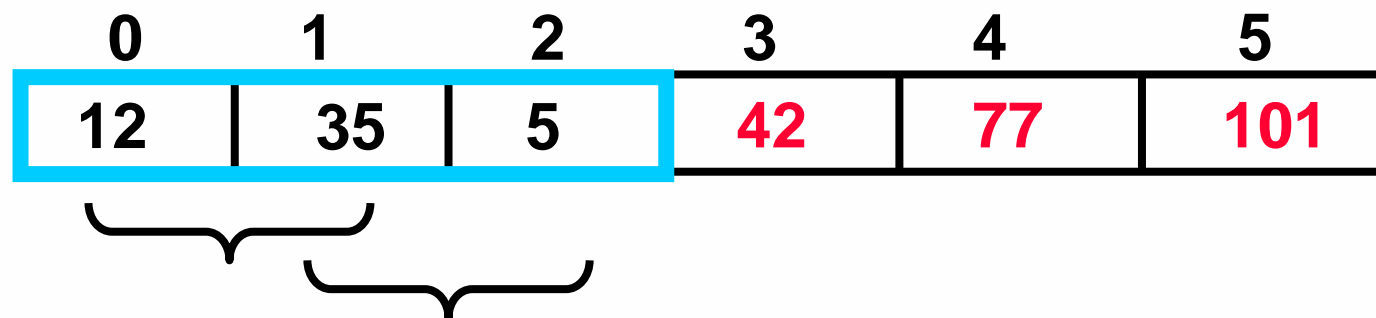
0	1	2	3	4	5
5	12	35	42	77	101

Pengurangan Jumlah Perbandingan

0	1	2	3	4	5
77	42	35	12	101	5
0	1	2	3	4	5
42	35	12	77	5	101
0	1	2	3	4	5
35	12	42	5	77	101
0	1	2	3	4	5
12	35	5	42	77	101
0	1	2	3	4	5
12	5	35	42	77	101

Pengurangan Jumlah Pembandingan

- Pada proses “bubble up” ke-P, kita hanya butuh untuk melakukan sebanyak $MAX-P$ pembandingan.
- MAX = jumlah data
- Contoh:
 - Ini adalah proses “bubble up” ke-4
 - MAX adalah 6
 - Jadi, kita punya 2 ($6-4$) pembandingan yang harus dilakukan



Putting It All Together

Pseudocode Bubble Sort

(untuk semua iterasi)

```
PROCEDURE bubbleSort( A : array data, N: jml data )  
  REPEAT  
    FOR i=0 to N-1 INCLUSIVE DO:  
      FOR j = 0 TO N- i-1 INCLUSIVE DO:  
        IF A[j] > A[j+1] THEN  
          swap( A[j], A[j+1] )  
        END IF  
      END FOR  
    END PROCEDURE
```



Apakah seluruh elemen telah terurut?

- Bagaimana jika seluruh elemen telah terurut?
- Bagaimana jika hanya sedikit elemen yang tidak pada posisinya, dan setelah beberapa operasi “bubble up,” seluruh elemen telah terurut?
- Kita menginginkan untuk bisa mendeteksi kondisi ini dan “stop early”!

0	1	2	3	4	5
5	12	35	42	77	101

0	1	2	3	4	5
10	1	3	4	7	9

Gunakan sebuah “Flag” Boolean

- Kita bisa menggunakan sebuah variabel boolean untuk menentukan apakah terjadi operasi swapping selama proses “bubble up.”
- Jika tidak terjadi, maka kita akan mengetahui bahwa seluruh elemen telah terurut!
- “flag” boolean ini perlu di-RESET setiap kali selesai satu kali operasi “bubble up.”

FOR dgn flag

```
1.  PROCEDURE bubbleSort( A : array data, N: jml data )
2.    did_swap isotype Boolean
3.    did_swap <- true
4.    FOR i=0 TO N-1 INCLUSIVE DO:
5.        IF(DID_SWAP == true)
6.            did_swap <- false
7.            FOR j = 0 TO N- i-1 INCLUSIVE DO:
8.                if A[j] > A[j+1] then
9.                    swap( A[j], A[j+1] )
10.                   did_swap <- true
11.                END IF
12.            END FOR
13.        END IF
14.    END PROCEDURE
```

Ringkasan

- Algoritma “Bubble Up” akan memindahkan elemen terbesar ke posisinya yang tepat (di sebelah kanan)
- Ulangi proses “Bubble Up” sampai seluruh elemen telah menempati posisinya yang tepat:
 - MAXIMUM sebanyak $N-1$ kali
 - Bisa berakhir lebih cepat jika tidak lagi terjadi swapping (penukaran)
- Jumlah perbandingan berkurang 1x setiap kali satu elemen berhasil diletakkan pada posisinya yang tepat.



Bubble Sort → Analysis

- BEST CASE:
 - Array sudah dalam keadaan terurut naik
 - Jumlah perbandingan key (C) : $n-1$
 - Jumlah swap = 0
 - Jumlah pergeseran (M) : 0
- WORST CASE
 - Array dalam urutan kebalikannya
 - Jumlah perbandingan key (C) = $(n-1) + (n-2) + \dots + 1 = n * (n-1) / 2$
 - Jumlah swap = $(n-1) + (n-2) + \dots + 1 = n * (n-1) / 2$
 - Jumlah pergeseran (M) = $3 * n * (n-1) / 2$

Kompleksitas Bubble Sort

Perhatikan pada hubungan antara 2 loop yang ada:

- Inner loop bersarang di dalam outer loop
- Inner loop akan dieksekusi untuk setiap iterasi dari outer loop

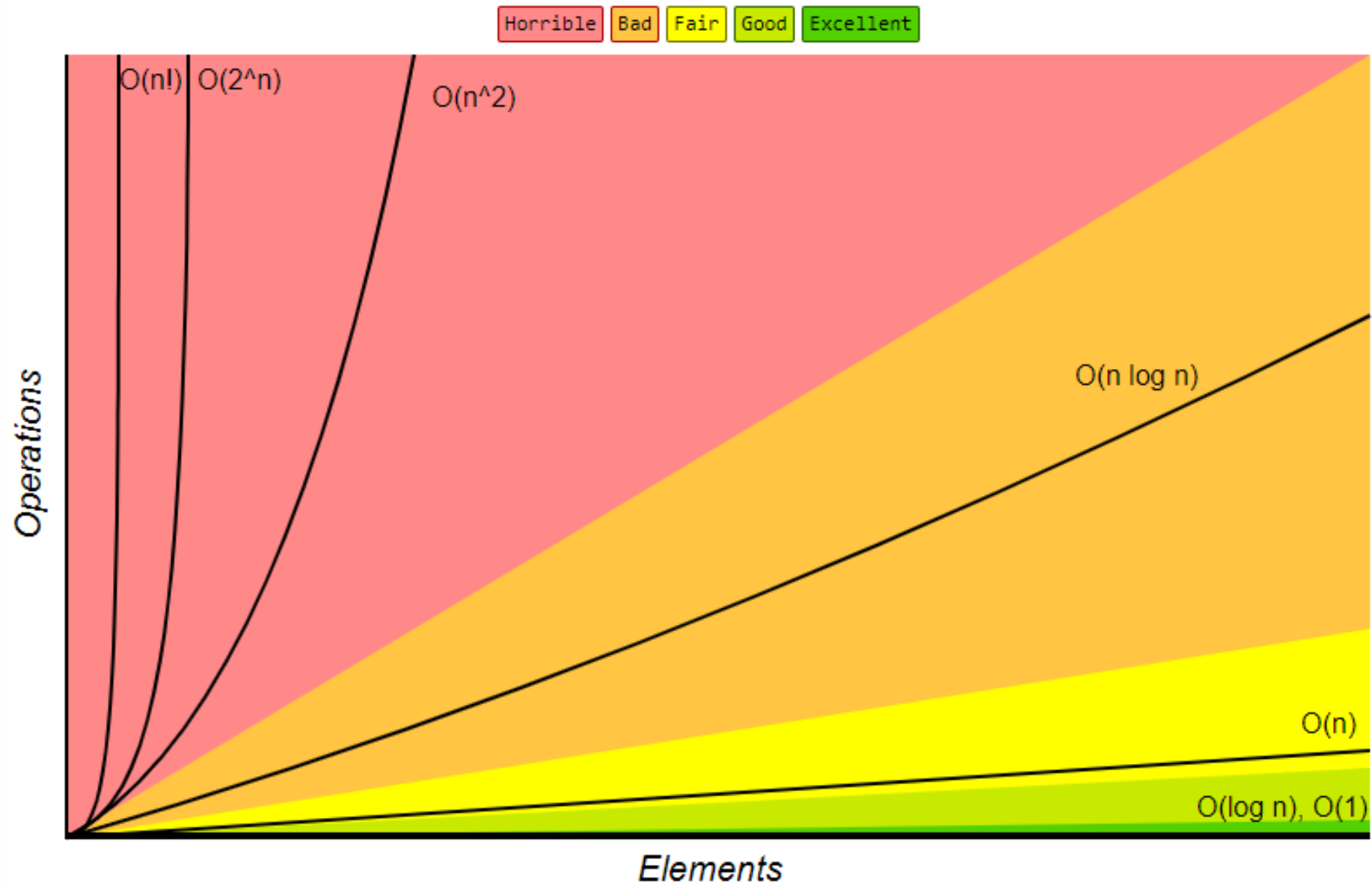
Bubble Sort

- Mirip dengan Selection, setiap kali proses “Bubble Up” akan memilih nilai maksimum dari elemen yang ada pada sisi unsorted.

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
<u>Quicksort</u>	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(\log(n))$
<u>Mergesort</u>	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
<u>Timsort</u>	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
<u>Heapsort</u>	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(1)$
<u>Bubble Sort</u>	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
<u>Insertion Sort</u>	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
<u>Selection Sort</u>	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
<u>Tree Sort</u>	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(n)$
<u>Shell Sort</u>	$\Omega(n \log(n))$	$\Theta(n(\log(n))^2)$	$O(n(\log(n))^2)$	$O(1)$
<u>Bucket Sort</u>	$\Omega(n+k)$	$\Theta(n+k)$	$O(n^2)$	$O(n)$
<u>Radix Sort</u>	$\Omega(nk)$	$\Theta(nk)$	$O(nk)$	$O(n+k)$
<u>Counting Sort</u>	$\Omega(n+k)$	$\Theta(n+k)$	$O(n+k)$	$O(k)$
<u>Cubesort</u>	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$

Big-O Complexity Chart



Kesimpulan

- Algoritma Bubble Sort adalah algoritma pengurutan dengan cara melakukan penukaran data dengan tepat disebelahnya secara terus menerus sampai bisa dipastikan dalam satu iterasi tertentu tidak ada lagi perubahan.
- Jika tidak ada perubahan berarti data sudah terurut.