

LECTURE 11

SORTING



Big-O Coding

Website: www.bigocoding.com

Giới thiệu

Độ phức tạp thuật toán (algorithmic complexity)

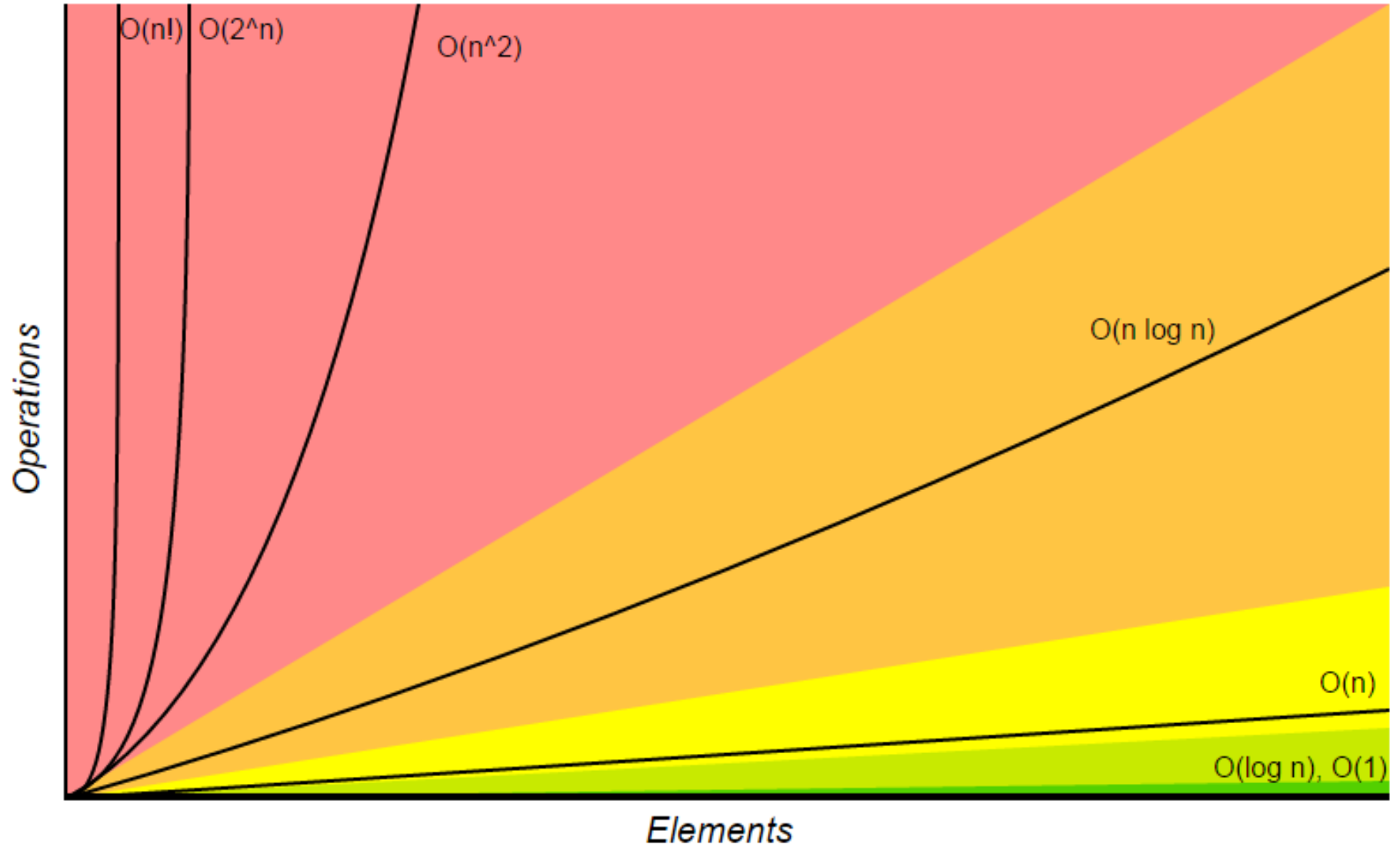
- Độ phức tạp thời gian (time complexity): thời gian chạy 1 thuật toán.
- Độ phức tạp không gian (space complexity): dung lượng bộ nhớ sử dụng khi chạy 1 thuật toán.

Thứ tự tăng dần các độ phức tạp thời gian

Dạng	Độ phức tạp	Ví dụ
Constant	$O(1)$	Truy cập mảng theo index
Logarithmic	$O(\log n)$	Tìm kiếm nhị phân
Linear	$O(n)$	Tìm kiếm tuần tự
Quasi-linear	$O(n \log n)$	Merge sort
Quadratic	$O(n^2)$	Duyệt mảng 2 chiều Insertion sort
Factorial	$O(n!)$	Bài toán người đưa thư

Big-O Complexity Chart

Horrible Bad Fair Good Excellent



Thời gian cho từng độ phức tạp

	n = 10	n = 20	n = 30	n = 40	n = 50	n = 60
$O(n)$	0.00001s	0.00002s	0.00003s	0.00004s	0.00005s	0.00006s
$O(n^2)$	0.0001s	0.0004s	0.0009s	0.0016s	0.0025s	0.0036s
$O(n^3)$	0.001s	0.008s	0.027s	0.064s	0.125s	0.216s
$O(n^5)$	0.1s	3.2s	24.3s	1.7min	5.2min	13.0min
$O(2^n)$	0.001s	1.0s	17.9min	12.7d	35.7y	366cent
$O(3^n)$	0.059s	58min	6.5y	3855cent	2e8cent	1.3e13cent

Số lượng phần tử n chấp nhận được

- Các máy chấm hiện tại, thường là 10^8 phép tính/s.

Length of Input (n)	Worst Accepted Algorithm
$\leq [10..11]$	$O(n!)$, $O(n^6)$
$\leq [15..18]$	$O(2^n * n^2)$
$\leq [18..22]$	$O(2^n * n)$
≤ 100	$O(n^4)$
≤ 400	$O(n^3)$
$\leq 2K$	$O(n^2 * \log n)$
$\leq 10K$	$O(n^2)$
$\leq 1M$	$O(n * \log n)$
$\leq 100M$	$O(n)$, $O(\log n)$, $O(1)$











Sắp xếp là gì?

5	7	2	5	9
---	---	---	---	---



2	5	5	7	9
---	---	---	---	---

- Có nhiều giải thuật sắp xếp khác nhau.
- Làm thế nào để sắp xếp hiệu quả nhất?

RANK	TEAM	SCORE	A	B	C	D	E	F	G	H	I	J	K	L
1	 Send Bobs to Alice National University of Singapore	10 1615	4/259	1/134	4	0	2/67	4/292	2/200	2/39	1/86	1/47	3/34	2/217
2	 map University of Engineering and Technolo	10 1806	6/268	1/80	5	0	1/38	13/293	2/273	3/72	1/108	1/25	6/61	1/88
3	 P_not_equal_NP Hanoi University of Science and Techn	9 1811	2/232	1/103	14	0	1/147	0	1/290	14/166	6/161	2/68	1/20	3/184
4	 HCMUS-Intimidate University of Science, VNU-HCM	8 1040	0	1/82	2/183	0	1/65	0	0	2/130	3/146	2/118	1/26	1/190
5	 HCMUS-Ascension University of Science, VNU-HCM	8 1114	8	1/104	3/209	0	1/126	0	1/291	2/76	1/94	2/81	1/53	0
6	 amazingbamboo_with_cococ Hanoi University of Science and Techn	8 1150	3	2/165	8	0	2/101	0	1/253	2/98	2/131	1/27	2/48	2/207
7	 trie University of Engineering and Technolo	8 1206	5	1/52	7	1	8/218	1	1/159	2/99	4/79	1/39	1/28	4/252
8	 THREE Hanoi University of Science	8 1441	1/206	2/151	3/251	0	1/145	0	0	4	1/210	1/96	3/30	2/232
9	 bitset University of Engineering and Technolo	8 1488	0	2/291	0	0	4/90	5	1/203	9/216	3/112	1/43	2/34	3/159
10	 ONE Hanoi University of Science	7 824	2/128	1/190	3	0	1/82	0	1	2/147	2/43	1/162	1/12	3



Samsung Galaxy J3 2016
2.650.000 đ



Samsung Galaxy J4 (2018)
2.650.000 đ



Samsung Galaxy J2 Pro (2018)
2.960.000 đ



Samsung Galaxy J4 Plus (2018)
3.050.000 đ



Samsung Galaxy J3 Pro (2017)
3.450.000 đ



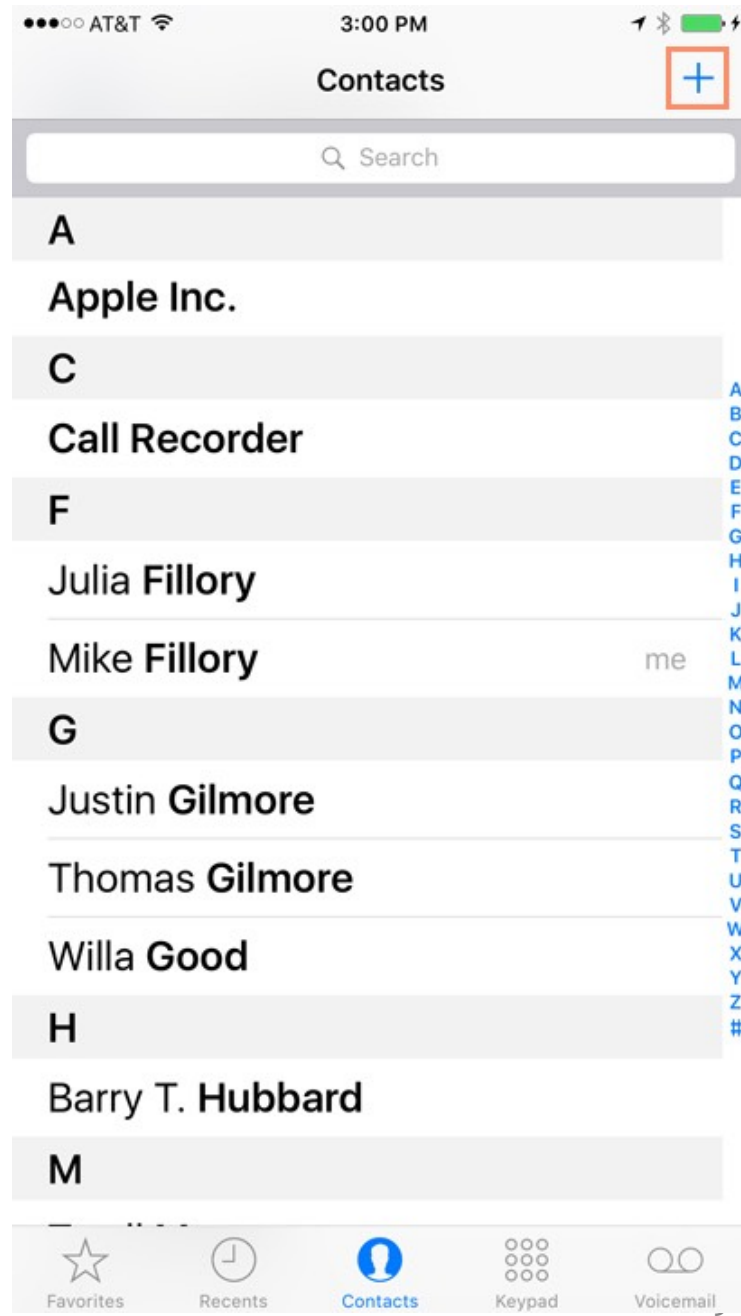
Samsung Galaxy J6 (2018)
3.890.000 đ



Samsung Galaxy J7 Prime
4.090.000 đ



Samsung Galaxy J6 Plus (2018)
4.090.000 đ



Các bài toán trên sorted list

- Tìm trung vị. (median)
- Tìm các cặp gần giống nhau. (closest pairs)
- Tìm kiếm nhị phân. (binary search)

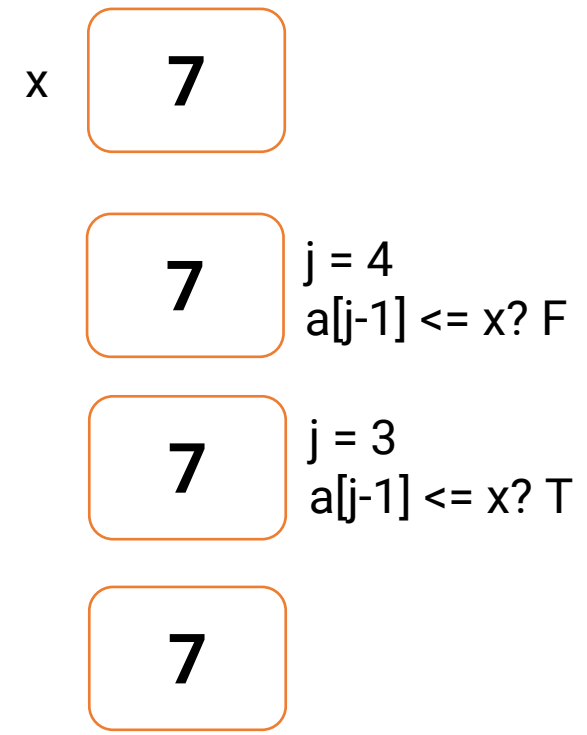
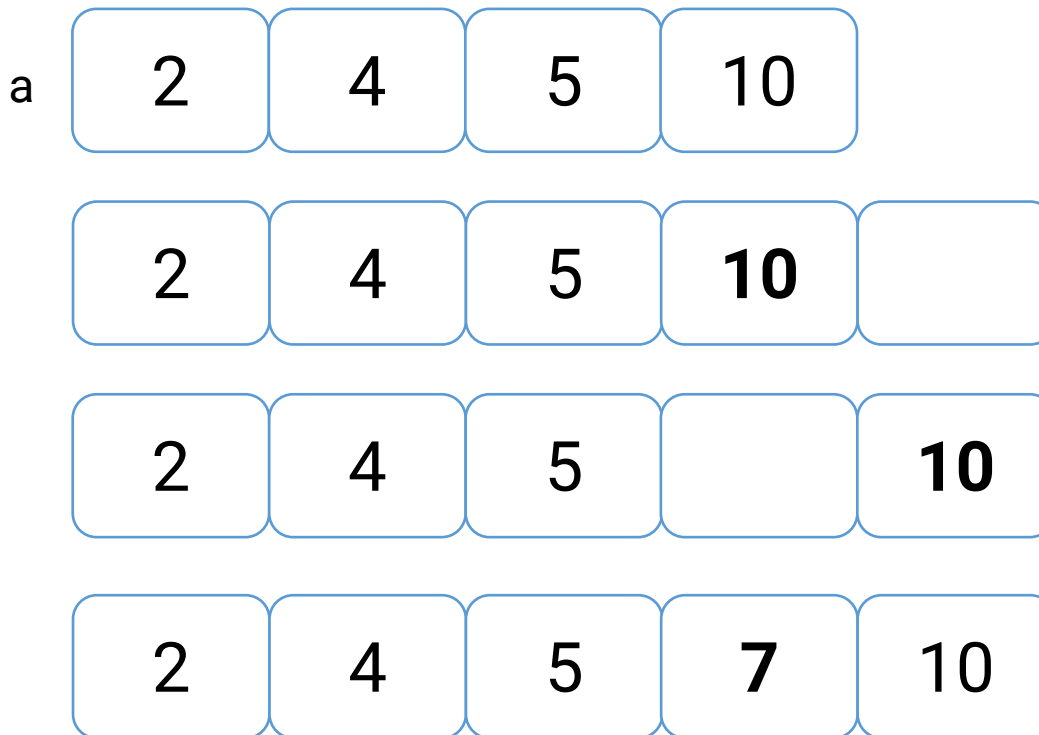
Insertion sort $\rightarrow O(n^2)$

Insertion sort – Đặt vấn đề

- Mảng a (k phần tử) đã tăng dần.
- Bỏ x vào vị trí nào trong mảng?
 - Để mảng a ($k+1$ phần tử) cũng tăng dần!



Ví dụ



Xét $a[j-1]$ và x để biết có nên thêm x vào $a[j]$ không

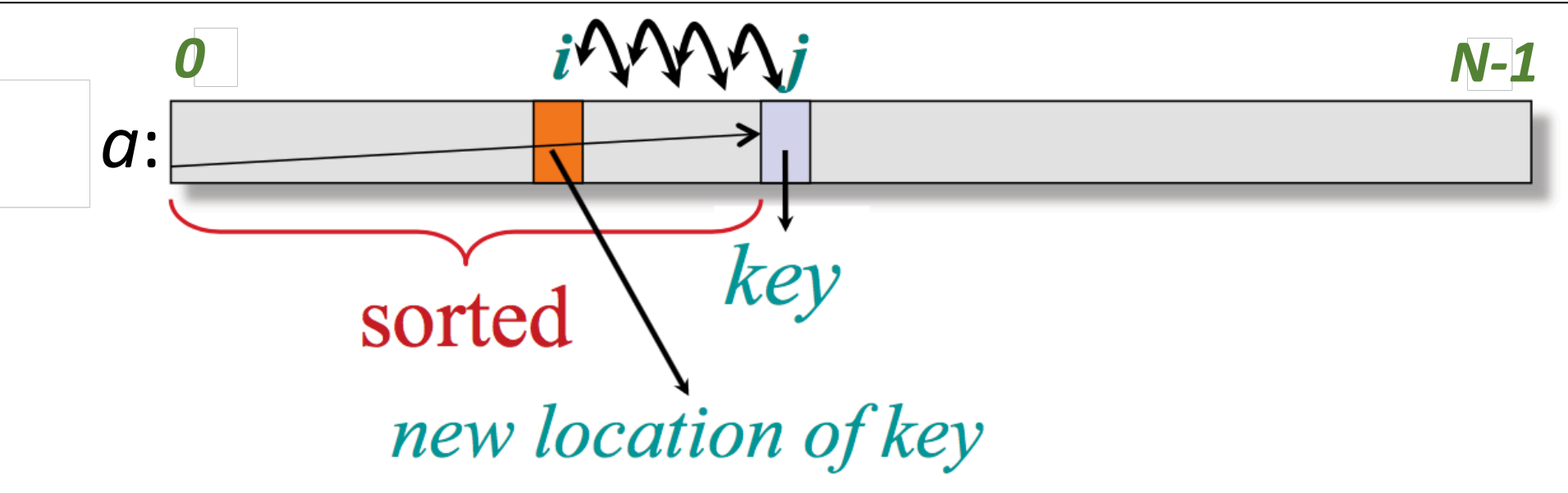
Giải thuật insertAsc(a, n, x)

1. Cho j chạy ngược, từ n về 0.
 - a. Nếu $j = 0$, thoát khỏi vòng lặp.
 - b. Nếu $a[j-1] \leq x$, thoát khỏi vòng lặp. Đây là vị trí j có thể thêm x vào.
 - c. Ngược lại, dời $a[j] = a[j-1]$.
2. Khi kết thúc vòng lặp, gán $a[j] = x$.

Insertion sort – Ý tưởng

Trước khi insert	Insert	Sau khi insert
$a[0]$ tăng dần	$a[1]$	$a[0]$, $a[1]$ tăng dần
$a[0]$, $a[1]$ tăng dần	$a[2]$	$a[0]$, $a[1]$, $a[2]$ tăng dần
...
$a[0]$, $a[1]$, ..., $a[n-4]$, $a[n-3]$ tăng dần	$a[n-2]$	$a[0]$, $a[1]$, ..., $a[n-4]$, $a[n-3]$, $a[n-2]$ tăng dần
$a[0]$, $a[1]$, ..., $a[n-3]$, $a[n-2]$ tăng dần	$a[n-1]$	$a[0]$, $a[1]$, ..., $a[n-3]$, $a[n-2]$, $a[n-1]$ tăng dần

Insertion sort – Minh họa



Giải thuật insertionSort(a, n)

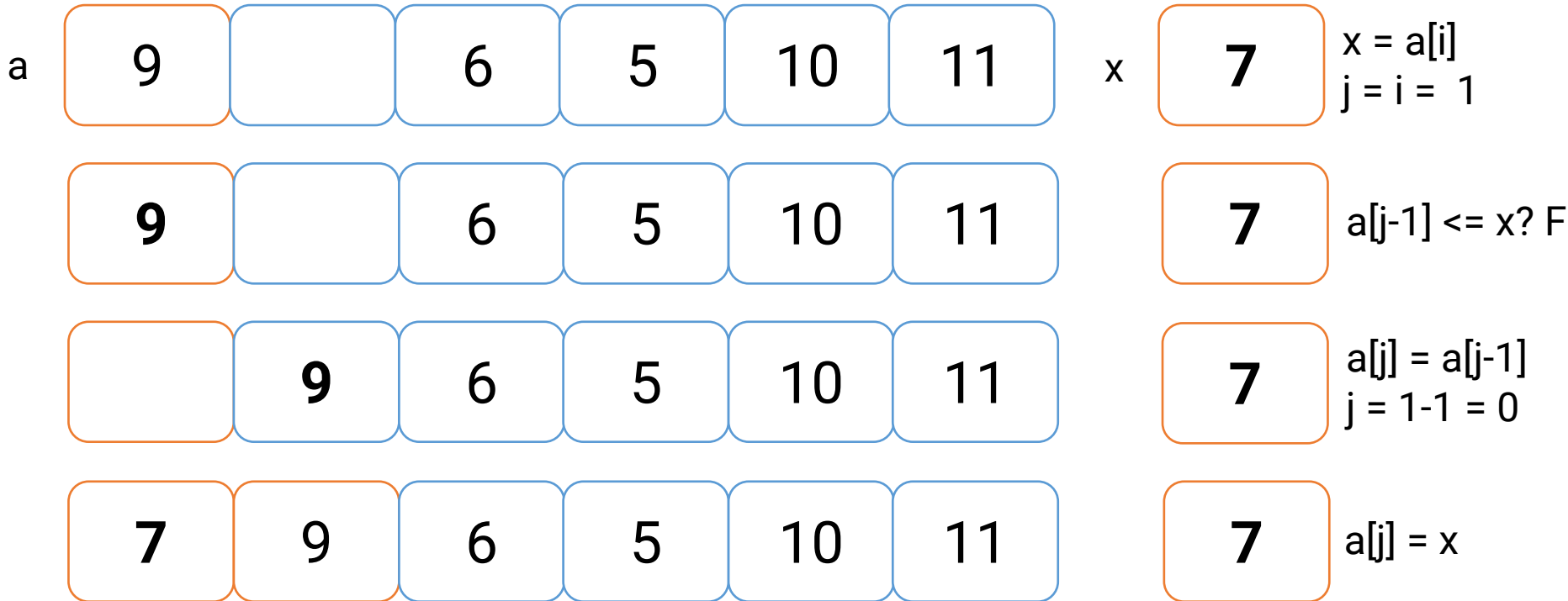
1. Cho i chạy từ 1 đến $n-1$.
 - a. Gán $x = a[i]$.
 - b. Gọi giải thuật insertAsc để insert x vào mảng a , có i phần tử.

Insertion sort – Ví dụ

- Sắp xếp dãy số sau tăng dần từ $a[0]$ đến $a[5]$.

9	7	6	5	10	11
---	---	---	---	----	----

Xét $i = 1$, $a[1] = 7$, $\text{insertAsc}(a, 1, 7)$



Xét $i = 2$, $a[2] = 6$, $\text{insertAsc}(a, 2, 6)$



Xét $i = 3$, $a[3] = 5$, $\text{insertAsc}(a, 3, 5)$

a

6	7	9		10	11
---	---	---	--	----	----

x

5

$x = a[i]$
 $j = i = 3$

6	7	9		10	11
---	---	---	--	----	----

5

$a[j-1] \leq x$? F

6	7		9	10	11
---	---	--	---	----	----

5

$a[j] = a[j-1]$
 $j = 3-1 = 2$

6	7		9	10	11
---	---	--	---	----	----

5

$a[j-1] \leq x$? F

6		7	9	10	11
---	--	---	---	----	----

5

$a[j] = a[j-1]$
 $j = 2-1 = 1$

6		7	9	10	11
---	--	---	---	----	----

5

$a[j-1] \leq x$? F

	6	7	9	10	11
--	---	---	---	----	----

5

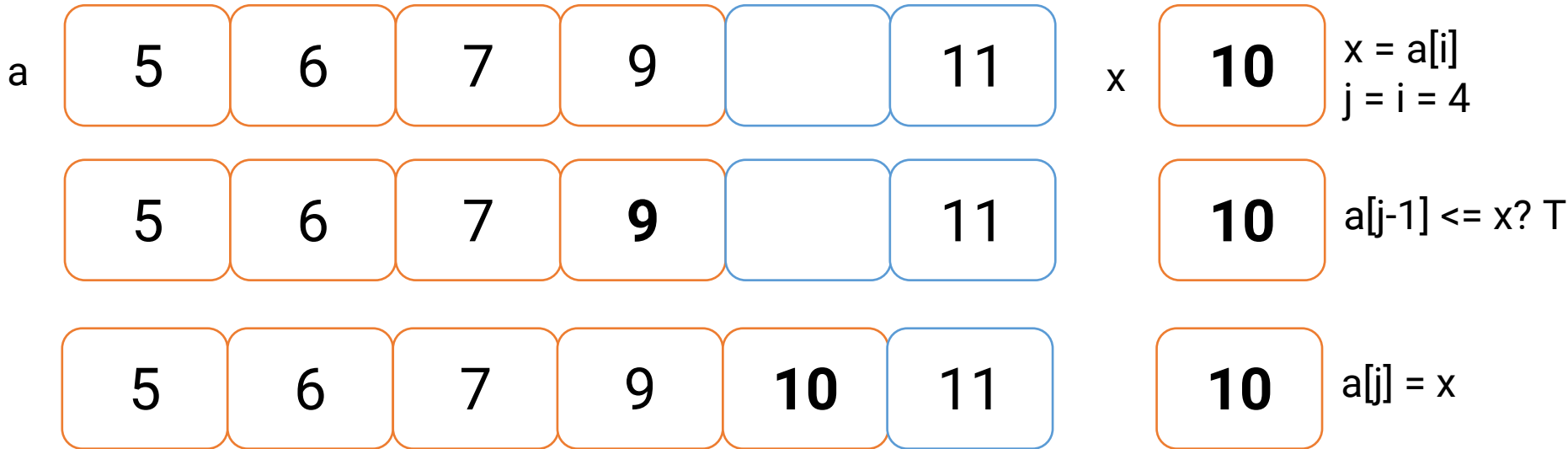
$a[j] = a[j-1]$
 $j = 1-1 = 0$

5	6	7	9	10	11
---	---	---	---	----	----

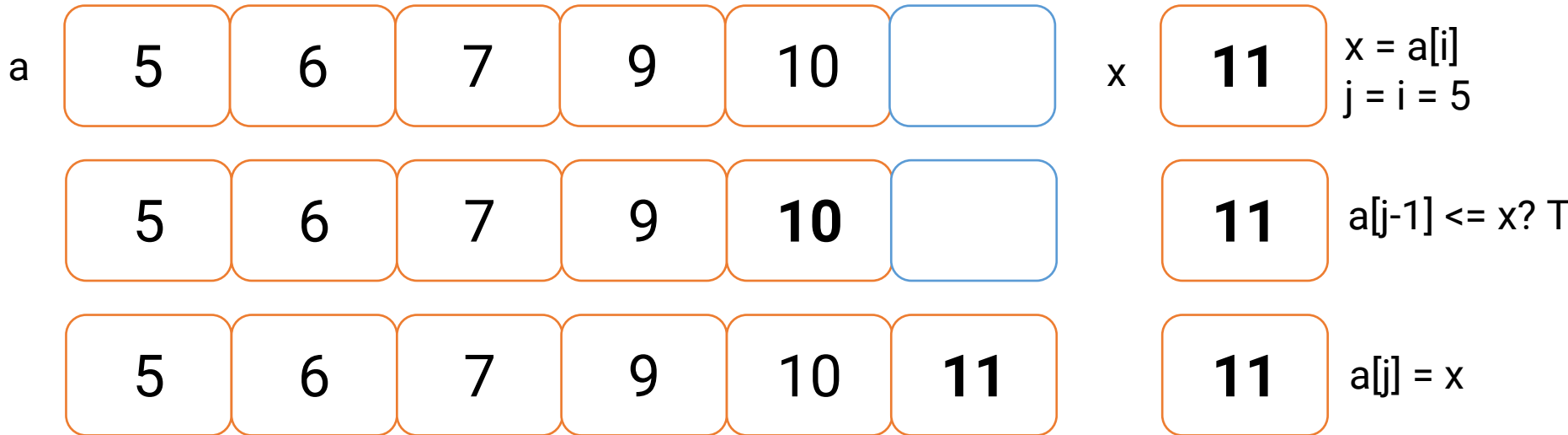
5

$a[j] = x$

Xét $i = 4$, $a[4] = 10$, $\text{insertAsc}(a, 4, 10)$



Xét $i = 5$, $a[5] = 11$, $\text{insertAsc}(a, 5, 11)$



BT1 – SẮP XẾP TĂNG DẦN

- Cho mảng một chiều các số nguyên. Hãy sắp xếp mảng tăng dần.



BT1 – Gợi ý – Xử lí chính

1. Đọc vào mảng a.
2. Gọi giải thuật insertion sort.
3. In mảng a (sau khi đã sắp xếp).



BT1 – Gợi ý hàm `insertSort(a, n)`

1. Cho i chạy từ 1 đến $n-1$.
 - a. Gán $x = a[i]$.
 - b. Gọi giải thuật `insertAsc` để insert x vào mảng a , có i phần tử.



BT2 – SẮP XẾP GIẢM DẦN

- Cho mảng một chiều các số nguyên. Hãy sắp xếp mảng giảm dần.



BT2 – Gợi ý

- Làm tương tự như BT1, nhưng thay đổi phép so sánh $a[j-1]$ và x .



BT2 – Insertion sort - TLE

Length of Input (n)	Worst Accepted Algorithm
$\leq [10..11]$	$O(n!), O(n^6)$
$\leq [15..18]$	$O(2^n * n^2)$
$\leq [18..22]$	$O(2^n * n)$
≤ 100	$O(n^4)$
≤ 400	$O(n^3)$
$\leq 2K$	$O(n^2 * \log n)$
$\leq 10K$	$O(n^2)$
$\leq 1M$	$O(n * \log n)$
$\leq 100M$	$O(n), O(\log n), O(1)$

Merge sort $\rightarrow O(n \log n)$

Merge sort $\rightarrow O(n \log n)$

Merge sort – Đặt vấn đề

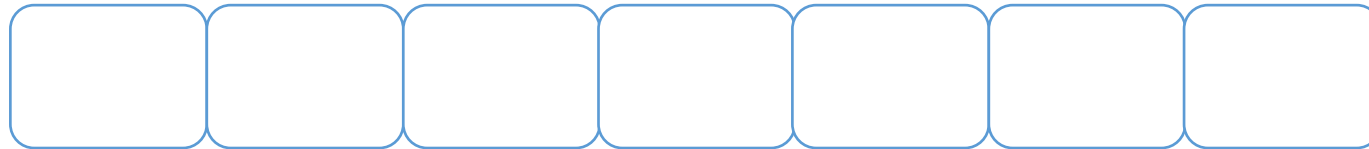
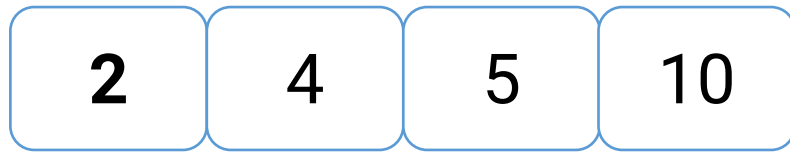
- Mảng L, có n_L phần tử đã tăng dần.
- Mảng R, có n_R phần tử đã tăng dần.
- Làm sao trộn L + R thành mảng a, có $n_L + n_R$ phần tử, cũng tăng dần?
- Time complexity: $O(n_L + n_R)$.

Gợi ý giải thuật merge

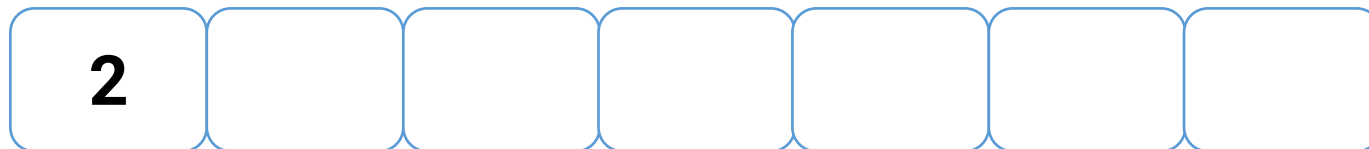
1. i, j, k bắt đầu từ 0. (i cho L , j cho R và k cho a)
2. Vòng lặp với điều kiện $i < nL$ và $j < nR$:
 - a. Nếu $L[i] < R[j]$
 - i. Đưa $L[i]$ vào vị trí k trong mảng a
 - ii. Tăng i
 - b. Ngược lại
 - i. Đưa $R[j]$ vào vị trí k trong mảng a
 - ii. Tăng j
 - c. Tăng k
3. Duyệt qua các phần tử còn lại của mảng L , $a[k] = L[i]$.
4. Duyệt qua các phần tử còn lại của mảng R , $a[k] = R[j]$.



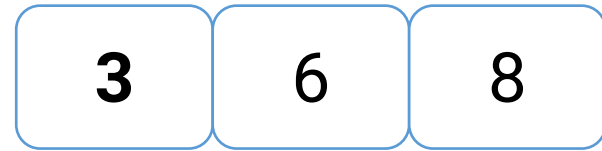
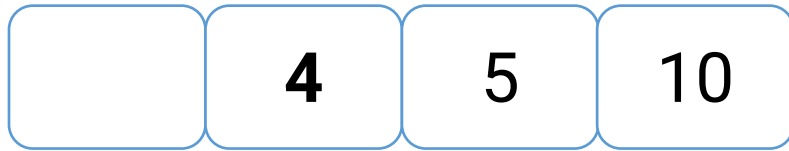
Minh họa



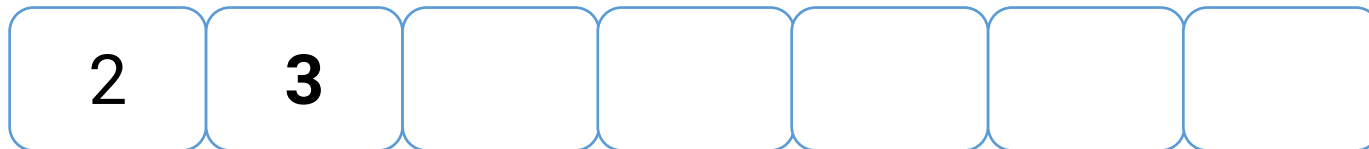
$i = 0$
 $j = 0$
 $L[i] < R[j]?$



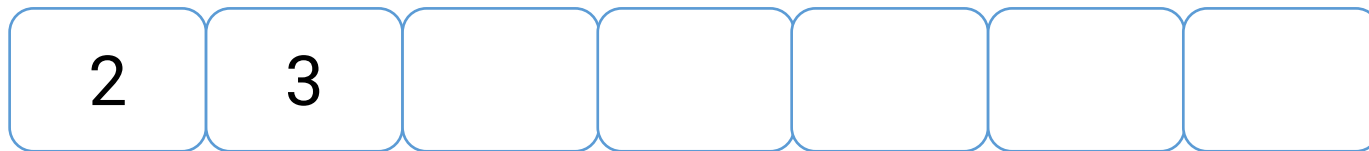
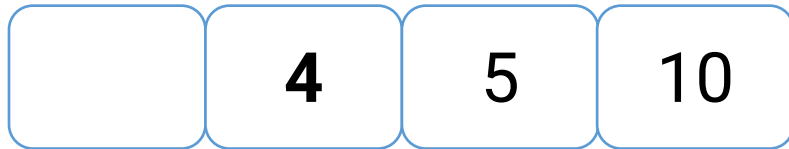
Minh họa



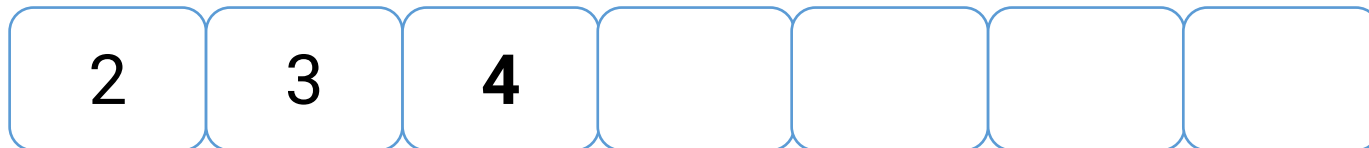
$i = 1$
 $j = 0$
 $L[i] < R[j]?$



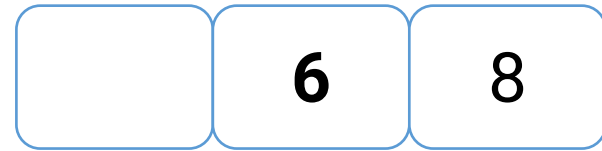
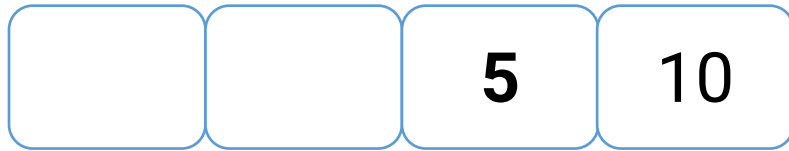
Minh họa



$i = 1$
 $j = 1$
 $L[i] < R[j]?$



Minh họa



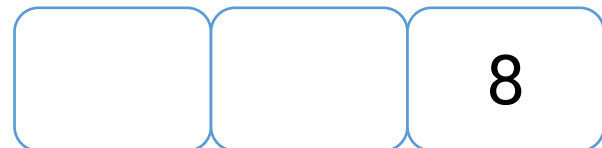
$i = 2$
 $j = 1$
 $L[i] < R[j]?$



Minh họa



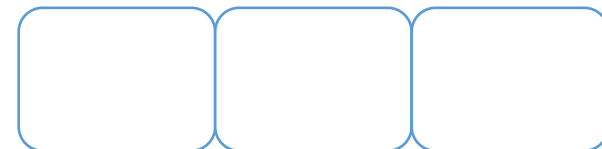
$i = 3$
 $j = 1$
 $L[i] < R[j]?$



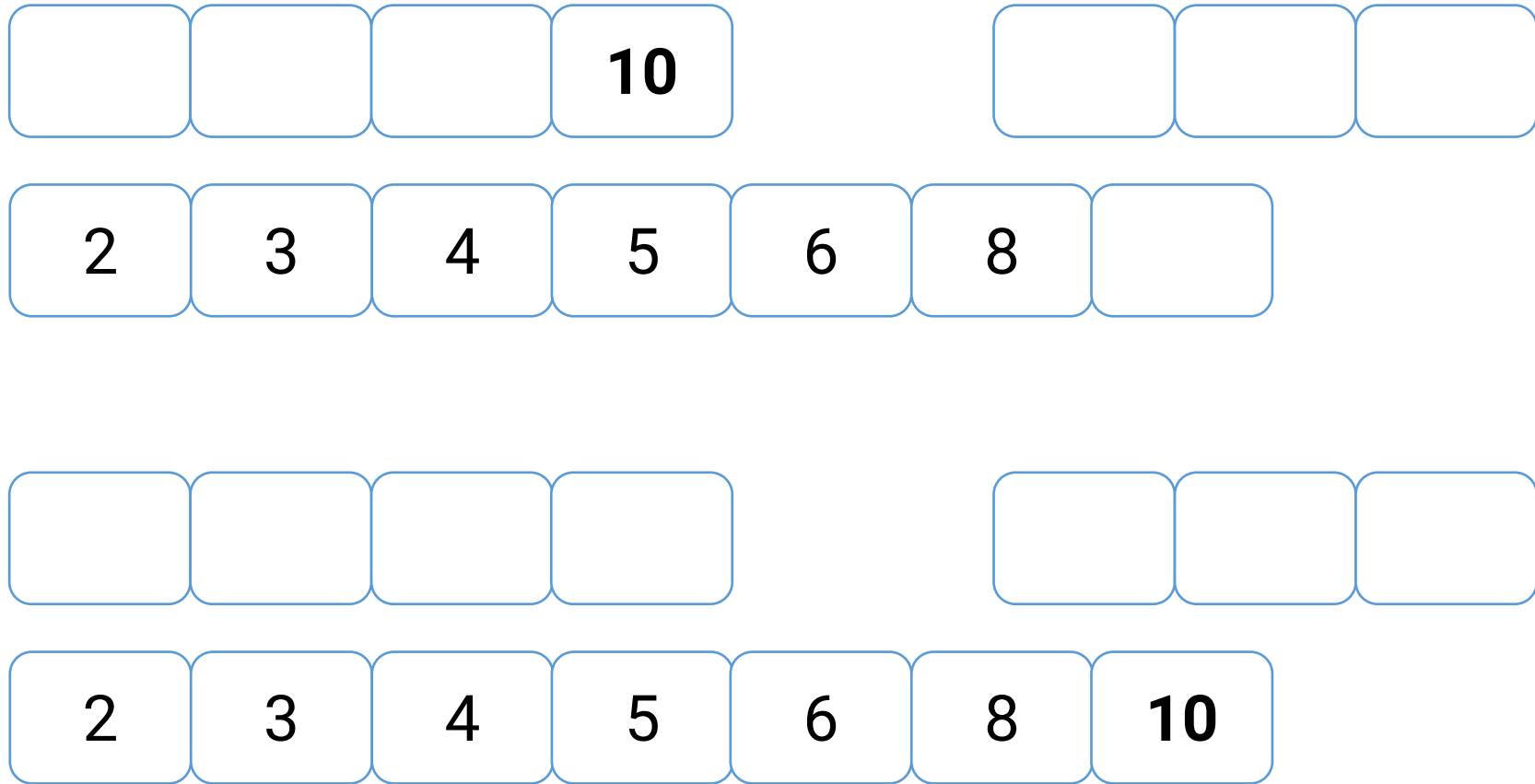
Minh họa



$i = 3$
 $j = 2$
 $L[i] < R[j]?$



Minh họa



Merge sort – Ý tưởng

- Sử dụng chiến lược chia để trị. (divide-and-conquer approach)
 - Nếu mảng có 1 phần tử, mảng đã sắp xếp tăng dần, ko cần làm gì cả.
 - Ngược lại, thực hiện Merge sort trên 2 mảng con $a[0 \dots n/2 - 1]$ và $a[n/2 \dots n - 1]$.
 - Thực hiện thao tác Merge trên 2 mảng đã sắp xếp, để được 1 mảng đã sắp xếp.

Merge sort – Minh họa

- Sắp xếp dãy số sau tăng dần từ $a[0]$ đến $a[5]$.



Merge sort – Minh họa



Merge sort – Minh họa

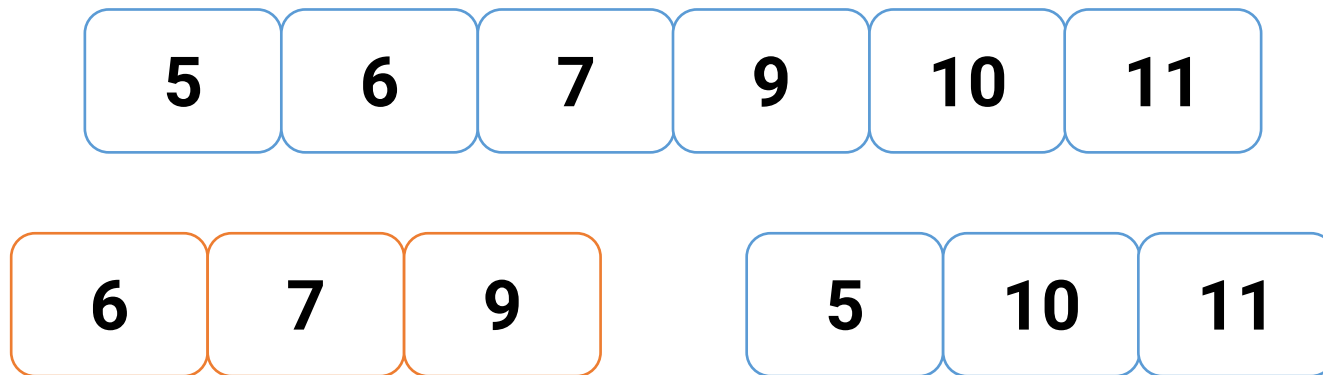
6 7 9

5 10 11

7 9 6

5 10 11

Merge sort – Minh họa



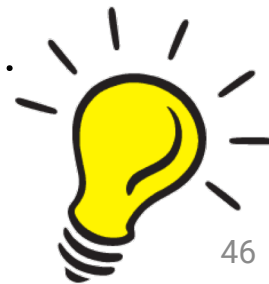
BT2 – Gợi ý – Xử lí chính

1. Đọc vào mảng a.
2. Gọi giải thuật merge sort.
3. In mảng sau khi sắp xếp.



BT2 – Gợi ý hàm mergeSort()

- Nếu $n \leq 1$:
 - Mảng 1 phần tử đã được sắp xếp.
- $nL = n / 2$.
- $nR = n - nL$.
- L: lấy từ $a[0]$ đến $a[nL-1]$ (slicing).
- R: lấy từ $a[nL]$ đến $a[n-1]$ (slicing).
- Gọi đệ qui mergeSort(L, nL) cho mảng L.
- Gọi đệ qui mergeSort(R, nR) cho mảng R.
- Gọi hàm merge(L, nL, R, nR, a, n) để merge L, R vào a ban đầu.



BT2 – Gợi ý hàm merge()

1. i, j, k bắt đầu từ 0. (i cho L , j cho R và k cho a)
2. Vòng lặp với điều kiện $i < nL$ và $j < nR$:
 - a. Nếu $L[i] > R[j]$
 - i. Đưa $L[i]$ vào vị trí k trong mảng a
 - ii. Tăng i
 - b. Ngược lại
 - i. Đưa $R[j]$ vào vị trí k trong mảng a
 - ii. Tăng j
 - c. Tăng k
3. Duyệt qua các phần tử còn lại của mảng L , $a[k] = L[i]$.
4. Duyệt qua các phần tử còn lại của mảng R , $a[k] = R[j]$.



Slicing (Python)

```
a = [10, 20, 30, 40, 50, 60]
```

```
# begin = 1, end = 3
```

```
a12 = a[1:3]
```

```
# begin = 0, end = 4
```

```
a03 = a[:4]
```

```
# begin = 3
```

```
a35 = a[3:]
```

```
# all
```

```
a05 = a[:]
```

- Python hỗ trợ toán tử slicing, cho phép lấy nhanh một dãy các phần tử liên tiếp trong mảng.
- Sau khi slicing, thay đổi giá trị trên mảng mới không ảnh hưởng đến mảng cũ.

Slicing (C++)

```
int a[] = {10, 20, 30, 40, 50, 60};  
  
int begin = 1;  
int end = 3;  
int na02 = end - begin;  
int a02[100];  
for(int i = 0; i < na02; i++)  
    a02[i] = a[begin + i];
```

- C++ không hỗ trợ toán tử slicing.
- Phải dùng vòng lặp, copy từng giá trị từ mảng cũ qua mảng mới.

Slicing (Java)

```
int []a = {10, 20, 30, 40, 50, 60};  
  
int begin = 1;  
int end = 3;  
int na02 = end - begin;  
int []a02 = new int[na02];  
for(int i = 0; i < na02; i++)  
    a02[i] = a[begin + i];
```

- Java cũng không hỗ trợ toán tử slicing.
- Sử dụng toán tử new để tạo mảng mới.
- Dùng vòng lặp, copy từng giá trị từ mảng cũ qua mảng mới.

Thêm phần tử vào vị trí cuối trong mảng (Python)

```
a = [10, 20, 30]
# 10, 20, 30

a.append(40)
# 10, 20, 30, 40

a.append(50)
# 10, 20, 30, 40, 50
```

- Sử dụng hàm `append()`

Thêm phần tử vào vị trí cuối trong mảng (C++)

```
int a[100] = {10, 20, 30};  
int n = 3;  
// 10, 20, 30  
  
a[n] = 40;  
n++;  
// 10, 20, 30, 40  
  
a[n] = 50;  
n++;  
// 10, 20, 30, 40, 50
```

- Nếu $n < \text{số phần tử tối đa}$, gán $a[n] = x$ và tăng n .

Thêm phần tử vào vị trí cuối trong mảng (Java)

```
int []a = {10, 20, 30};

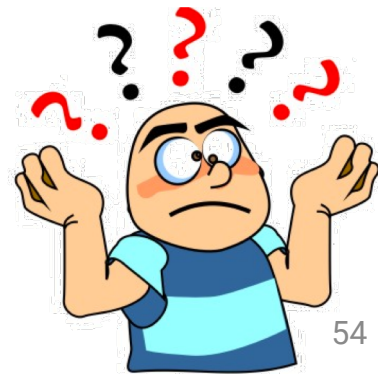
int []b;
b = new int[a.length + 1];
for(int i = 0; i < a.length; i++)
    b[i] = a[i];
b[b.length-1] = 40;

a = b;
```

- Tạo mảng tạm b → copy giá trị từ a qua b → thêm x vào cuối b → gán a = b.

BT3 – SẮP XẾP NGUYÊN TỐ

- Sắp xếp mảng tăng dần các số nguyên, nhưng các số nguyên tố vẫn giữ nguyên vị trí.



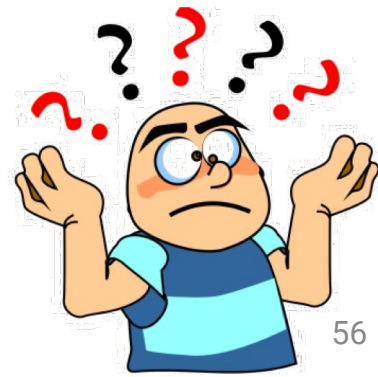
BT3 – Gợi ý – Xử lí chính

- Với giới hạn tối đa 1000 phần tử, bài này có thể sử dụng insertion sort hoặc merge sort
1. Đọc vào mảng a.
 2. Duyệt qua các phần tử trong mảng a, tách các số không phải nguyên tố đưa vào mảng b.
 3. Sử dụng giải thuật insertion sort hoặc merge sort trên mảng b.
 4. Khi in ra: $j = 0$ và duyệt for i trên mảng a
 1. Nếu $a[i]$ không là số nguyên tố, in $b[j]$, $j += 1$.
 2. Ngược lại, in $a[i]$.



BT4 – TÌM TRUNG VỊ MEDIAN

- Cho số nguyên dương N và mảng một chiều gồm N số nguyên. Hãy tìm giá trị trung vị của mảng đó.

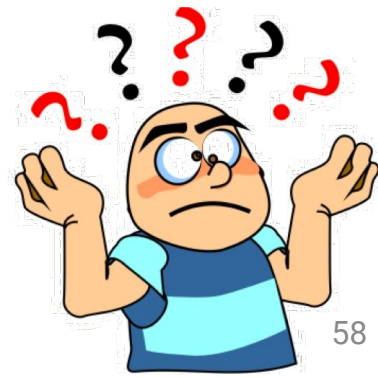


BT4 – Gợi ý

- Bài cũng có thể tùy chọn `insertionSort()` hoặc `mergeSort()`.
- Đọc vào mảng a.
- Sử dụng giải thuật insertion sort hoặc merge sort trên mảng a.
- In ra trung vị.

BT5 – SẮP XẾP CHẴN LẺ

- Cho mảng một chiều sắp xếp sao cho các vị trí có giá trị chẵn tạo thành dãy tăng dần, các vị trí có giá trị lẻ tạo thành dãy giảm dần



BT5 – Gợi ý

- Bài cũng có thể tùy chọn insertion sort hoặc merge sort.
- Tách các số chẵn, số lẻ trong mảng a thành 2 mảng even, odd riêng.
- Gọi hàm sắp xếp tăng 2 mảng even và odd.
- Khi in:
 - Nếu $a[i]$ là chẵn thì lấy trong even[ie] ra in.
 - Nếu $a[i]$ là lẻ thì lấy trong odd[io] ra in.
 - Lưu ý: ie tăng, io giảm.



Hỏi đáp

