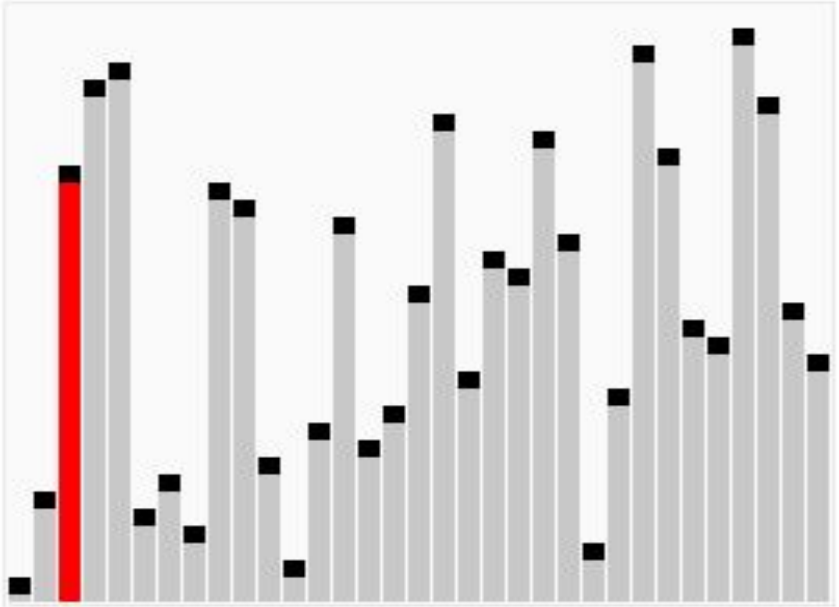
Sorting algorithm

dung.phamtrung@phenikaa-uni.edu.vn

Thuật toán sắp xếp

Thuật toán sắp xếp là một thuật toán đặt các phần tử của một danh sách theo thứ tự.

Các thứ tự được sử dụng thường xuyên nhất là thứ tự số và thứ tự từ vựng theo cả tăng dần hoặc giảm dần.



Bubble Sort

Bubble Sort

So sánh hai phần tử kề nhau, nếu chúng chưa đứng đúng thứ tự thì đổi chỗ

https://visualgo.net/en/sorting?slide=7

Bubble Sort

```
BubbleSort(Arr, n)
    for (i = n -1; i > 0; i--)
        for (j = 0; j < i; j++)
            if (Arr[j] > Arr[j + 1])
                swap(A[j], A[j + 1])
```

Bubble Sort

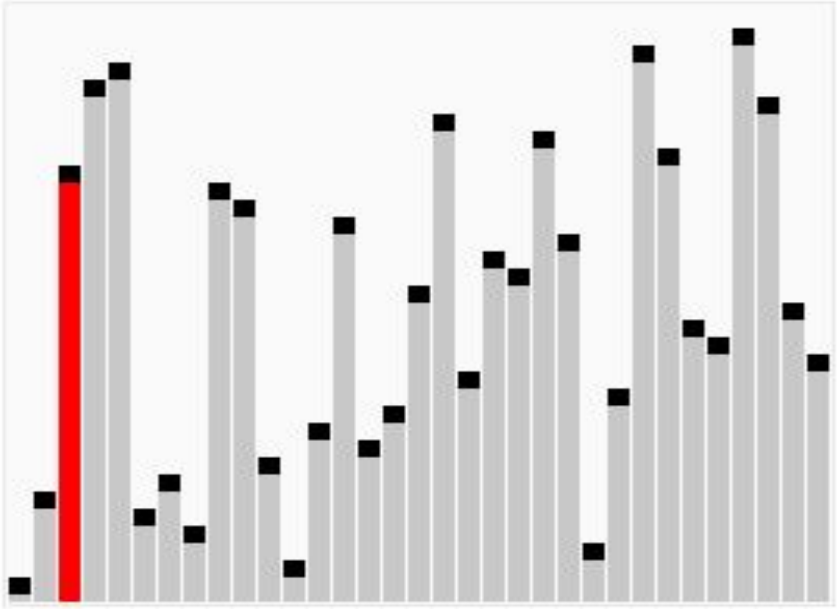
Độ phức tạp là $O(n^2)$

Có tính ổn định

Selection Sort

Seletion Sort

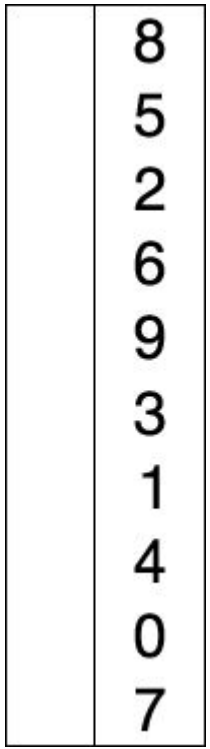
Tìm vị trí số nhỏ nhất (lớn nhất) rồi đổi chỗ



https://visualgo.net/en/sorting?slide=8

Seletion Sort

```
SeletionSort(Arr, n)
    for (i = 0; i < n - 1; i++)
        index_min = i
        for (j = i + 1; j < n; j++)
            if (Arr[j] < Arr[index_min])
                index_min = j
        swap(Arr[i], Arr[index_min])
```



Seletion Sort

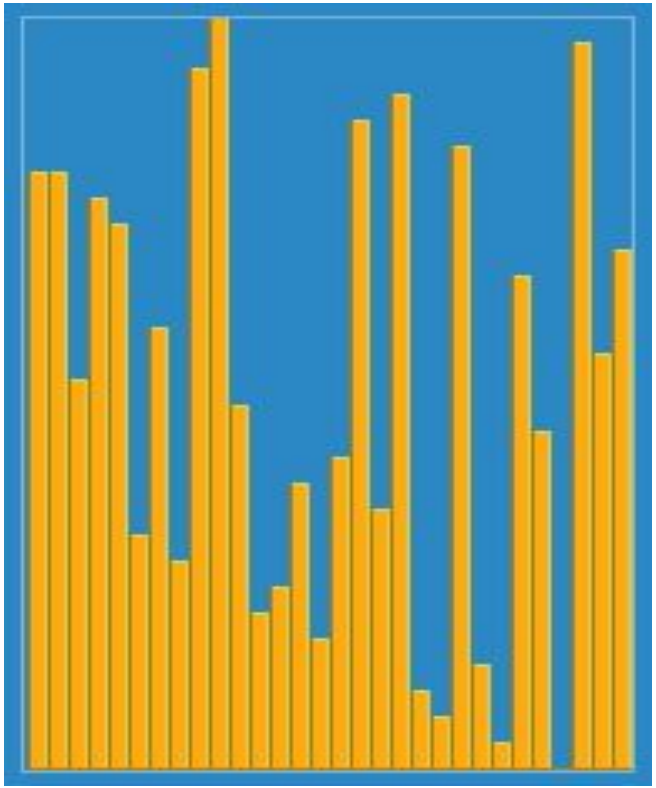
Độ phức tạp là $O(n^2)$

Có tính ổn định

Insertion Sort

Insertion Sort

Chèn một số vào một dãy đã sắp xếp



https://visualgo.net/en/sorting?slide=9

Insertion Sort

```
InsertionSort(Arr, n)
    for (i = 1; i < n; i++)
        tmp = Arr[i - 1]
        j = i - 1
        while (j >= 0 && Arr[j] > tmp)
            Arr[j + 1] = Arr[j]
            j--
        Arr[j + 1] = tmp
```

Insertion Sort

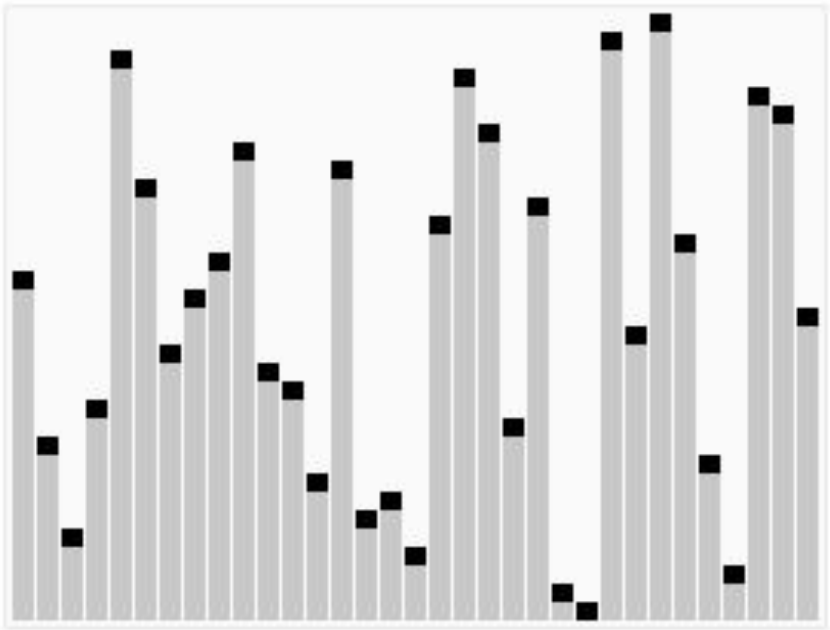
Độ phức tạp là $O(n^2)$

Có tính ổn định

Quick Sort

Quick Sort

Chia dãy số thành hai phần rồi sắp xếp hai phần đó

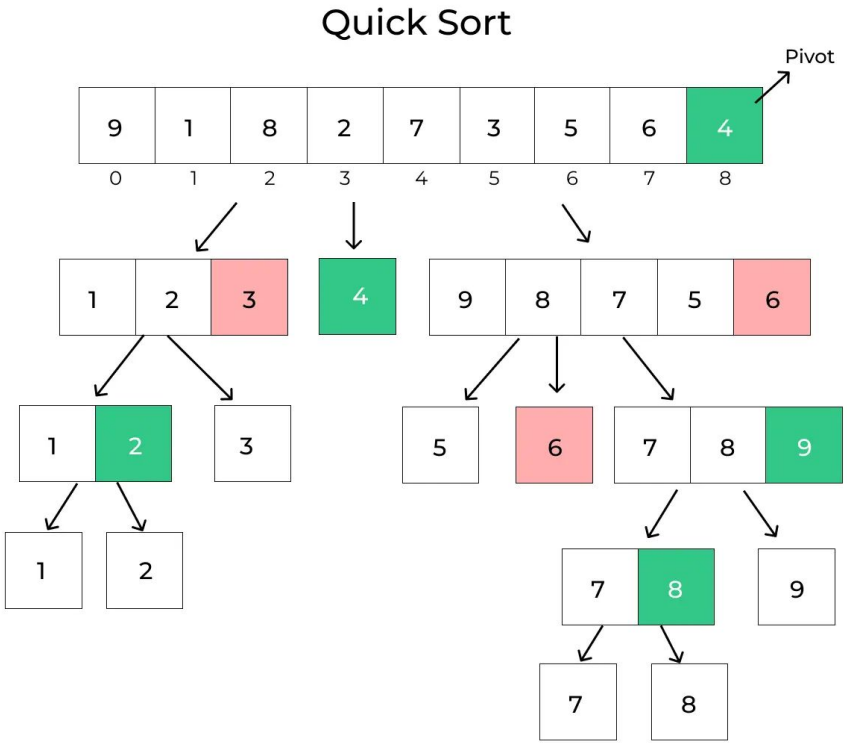


Quick Sort

Chọn một giá trị khóa

Chia dãy số thành hai phần: một là nhỏ hơn khóa, hai là lớn hơn hoặc bằng khóa

Sắp xếp hai dãy số con



Quick Sort

```
QuickSort(Arr, L, H)
    if (L < H)
        k = Arr[ramdon(L,H)]
        i = L; j = H;
        Lặp lại
            while  (A[i] < k) i++
            while  (A[j] > k) j--
            if (i <= j)
                if (i < j) Swap(A[i],A[j])
                ++i; --j;
        khi (i < j)
        QuickSort( L, j)
        QuickSort( i, H)
```

Quick Sort

Độ phức tạp

- Trường hợp tốt nhất: O(nlogn)
- Trường hợp xấu nhất: $O(n^2)$

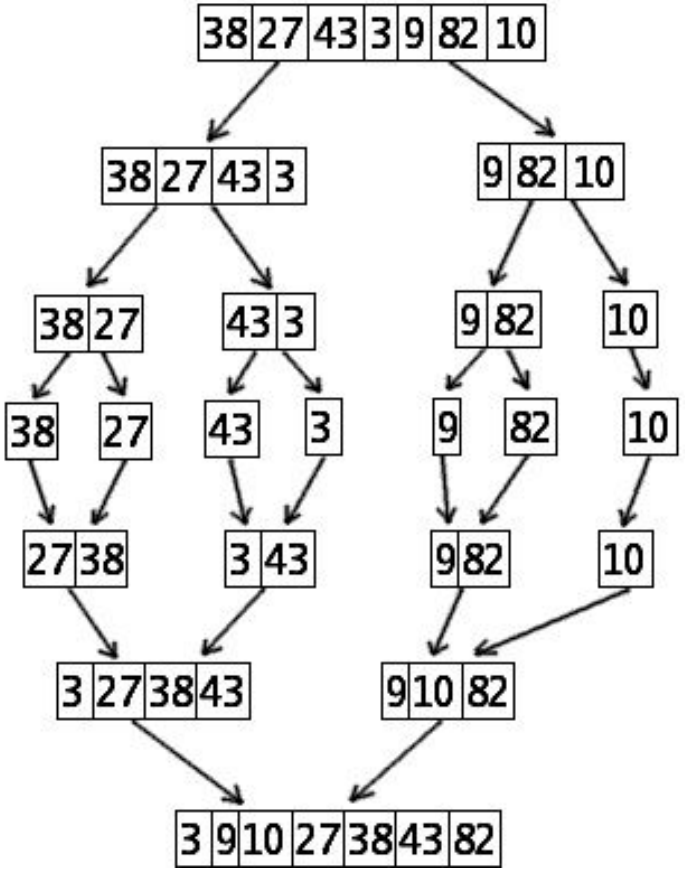
Không có tính ổn định

Là thuật toán có vận tốc trung bình là nhanh nhất

Merge Sort

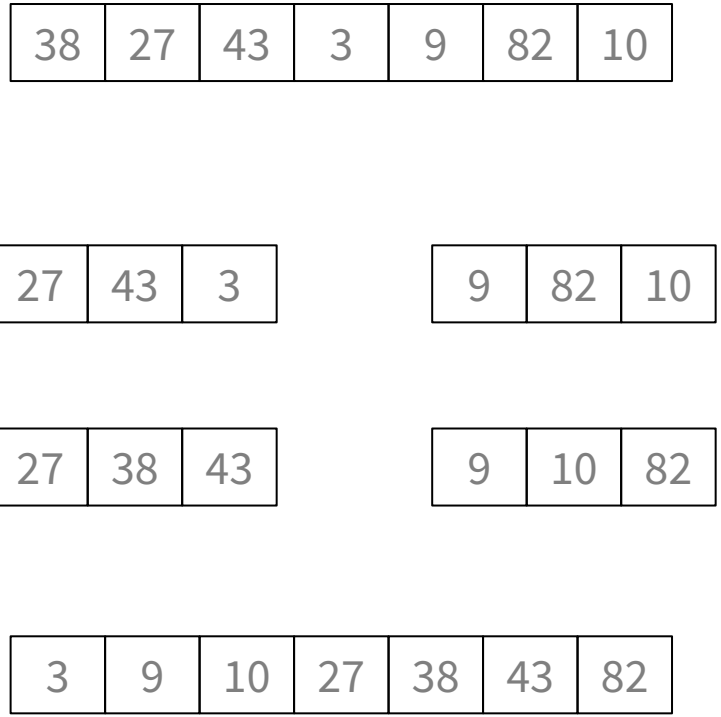
Merge Sort

Sắp xếp hai dãy đã được sắp xếp thành một dãy số được sắp xếp



Merge Sort

- Tách dãy số ra làm hai phần
- Sắp xếp hai dãy số đó
- Trộn hai dãy số lại thành dãy số sắp xếp



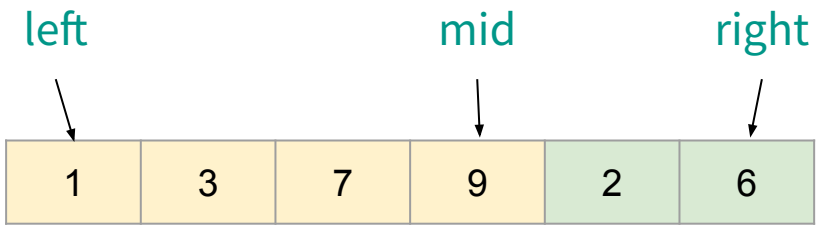
Megre

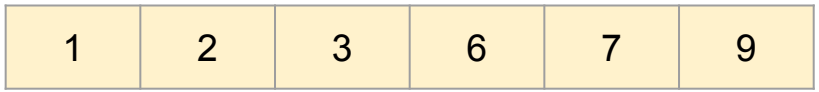
Cho hai dãy số đã được sắp xếp

- Dãy số A gồm n phần tử
- Dãy số B gồm m phần tử

Hãy trộn hai dãy số trên thành một dãy số C cũng được sắp xếp có n + m phần tử

A	B		C
1, 3, 7, 9	2, 6	1 < 2	1
3, 7, 9	2, 6	3 > 2	1, 2
3, 7, 9	6	3 < 6	1, 2, 3
7, 9	6	7 > 6	1, 2, 3, 6
7, 9			1, 2, 3, 6
			1, 2, 3, 6, 7, 9





Megre

```
Megre( Arr, left, mid, right)
        i = left;
        j = mid + 1
        v = left
        while (i <= mid && j <= right)
                if (Arr[i] < Arr[j])
                        result[v] = Arr[i]
                        i++
                else
                        result[v] = Arr[j]
                        j++
                v++
        while (i <= mid)
                result[v] = Arr[i]
                i++
                v++
        while (j <= right)
                result[v] = Arr[j]
                j++
                v++
        return result
```

Merge Sort

```
MergeSort(Arr, left, right)
    if (left < right)
        mid = (left + right) / 2
        MergeSort(Arr, left, mid)
        MergeSort(Arr, mid + 1, right)
        Merge(Arr, left, mid, right)
```

Merge Sort

Độ phức tạp O(nlogn)

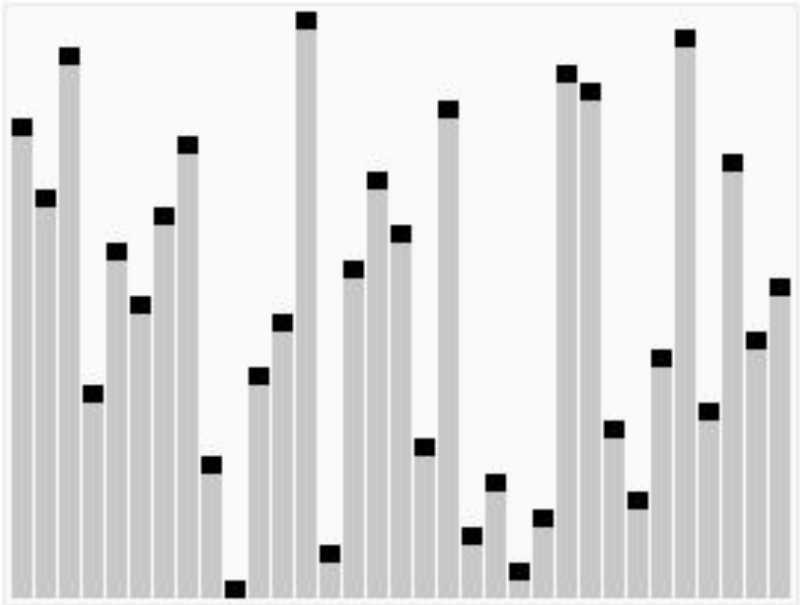
Có tính ổn định

Heap Sort

Heap Sort

Dựa trên cấu trúc Heap để tìm giá trị lớn nhất

Nó là một cải tiến của Selection Sort



Selection Sort

```
SelectionSort(Arr, n)
    for (i = n - 1; i > 0; i--)
        index_max = Arr[i]
        for (j = i - 1; j >= 0; j--)
            if (Arr[index_max] < Arr[j])
                index_max = j
        swap(Arr[index_max], Arr[i])
```

Tìm vị số lớn nhất
từ vị trí 0 đến vị trí i

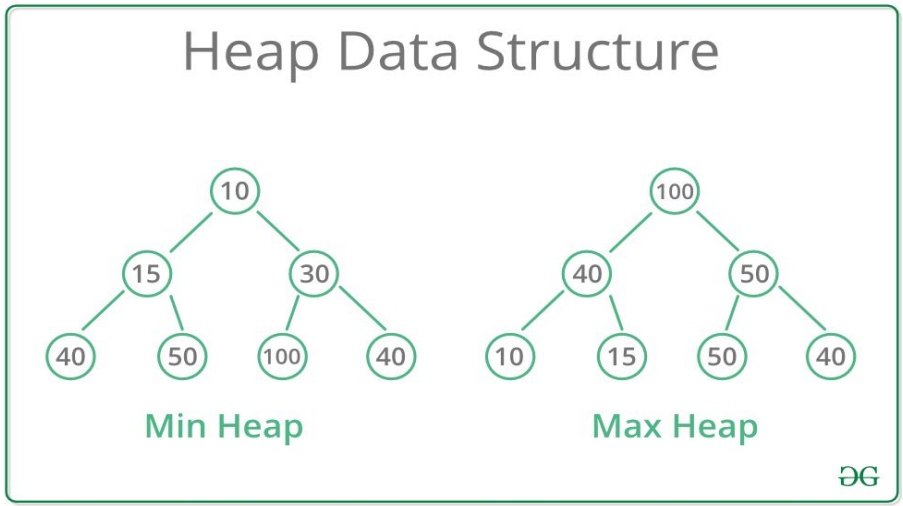
Cấu trúc Heap

Một mảng A có thể coi là một câu nhị phân với công thức

$$iLeftChild(i) = 2 \cdot i + 1$$
$$iRightChild(i) = 2 \cdot i + 2$$
$$iParent(i) = floor((i - 1) / 2)$$

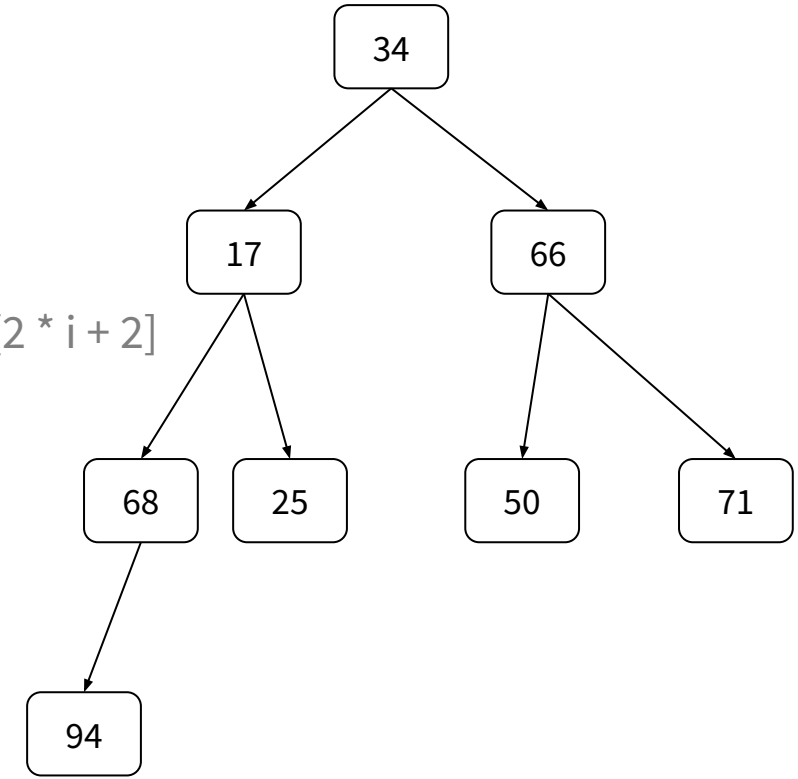
Mảng A có cấu trúc Heap cực đại nếu

$$a[i] >= a[2 * i + 1]$$
$$a[i] >= a[2 * i + 2]$$



PushDown

```
PushDown(Arr, root, max_index)
        i = root
        while (2*i + 1 <= max_index)
                tmp = 2 * i + 1
                if (2 * i + 2 <= max_index && Arr[tmp] < Arr[2 * i + 2]
                        tmp = tmp + 1
                if (Arr[i] < Arr[tmp])
                        swap(Arr[i], Arr[tmp])
                        i = tmp
                else
                        break
```

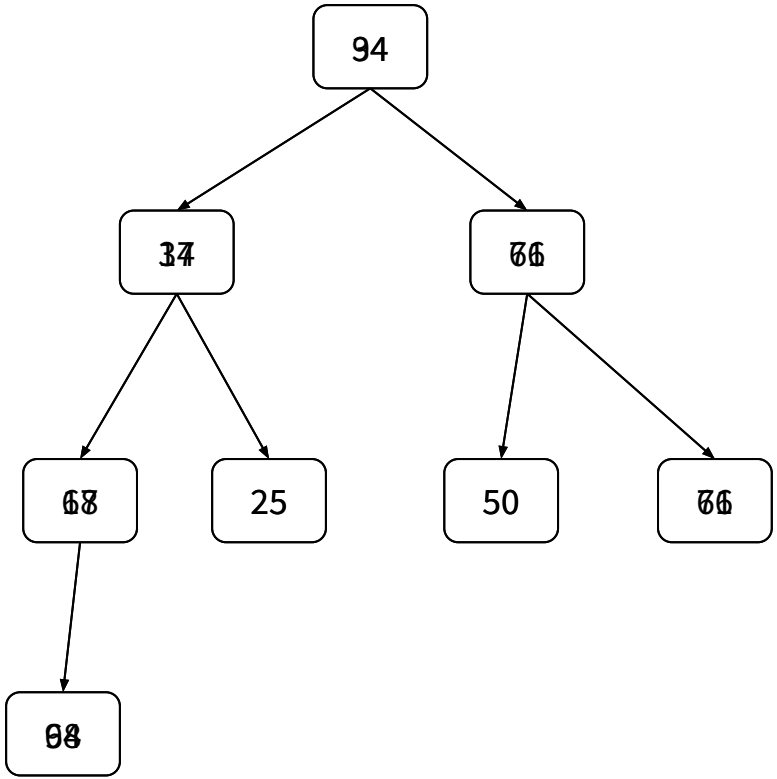


HeapSort

```
HeapSort(Arr, n)
    par = (n - 2) / 2;
    for (i = par; i >= 0; i--)
        pushDown(Arr, i, n - 1)
    for (i = n - 1; i > 0; i--)
        swap(Arr[0], Arr[i])
        pushDown(Arr, 0, i - 1)
```

PushUP

```
PushUP(Arr, max_index)
    par_index = (max_index - 1) / 2
    for (i = par_index; i >= 0; i--)
        tmp = 2*i + 1
        if (2*i + 2 <=max_index && Arr[tmp] < Arr[2*i + 2])
            tmp = 2*i + 2
        if (Arr[tmp] > Arr[i])
            swap(Arr[tmp], Arr[i])
```



Heap Sort

```
SelectionSort(Arr, n)
    for (i = n - 1; i > 0; i--)
        index_max = Arr[i]
        for (j = i - 1; j >= 0; j--)
            if (Arr[index_max] < Arr[j])
                index_max = j
        swap(Arr[index_max], Arr[i])
```

```
HeapSort(Arr, n)
    for (i = n - 1; i > 0; i--)
        PushUP(Arr, i)
        swap(Arr[0], Arr[i])
```

Heap Sort

Độ phức tạp O(nlogn)

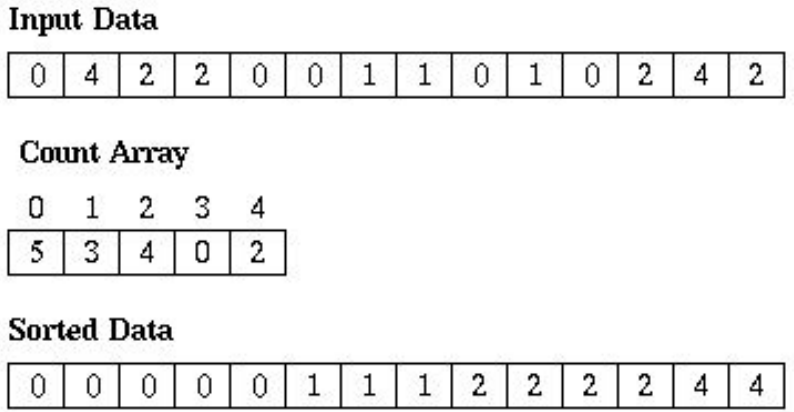
Không có tính ổn định

Couting Sort

Couting Sort

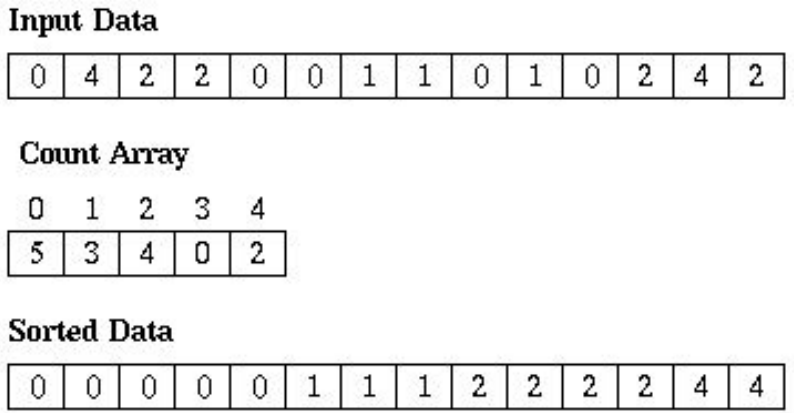
Đếm số lần xuất hiện của mỗi phần tử

Là thuật toán sắp xếp không so sánh



Couting Sort

- Đếm số lần xuất hiện của các phần tử trong danh sách
- In lại danh sách theo số đã đếm được



Couting Sort

```
CoutingSort(input, k)
    for (i = 0; i < length(input); i++)
        count[input[i]]++
    index = 0
    for (i = 0; i <= k; i++)
        for (j = 0; j < cout[i]; j++)
            output[index] = i
            index++
    return output
```

Input Data

0	4	2	2	0	0	1	1	0	1	0	2	4	2

Count Array

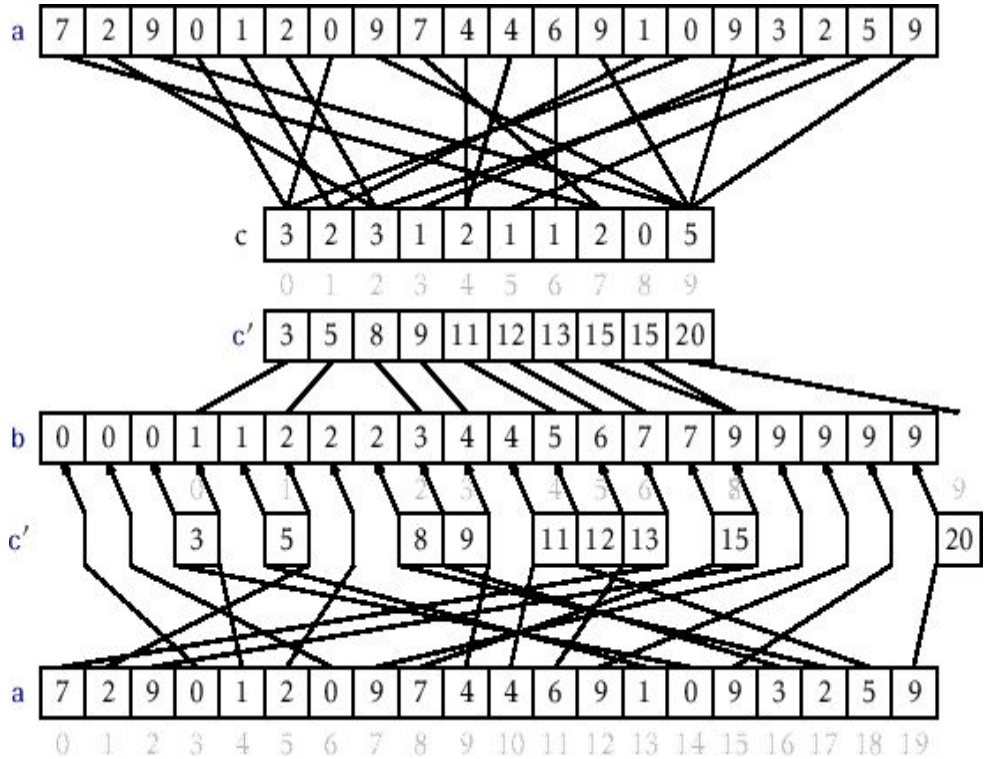
0	1	2	3	4
5	3	4	0	2

Sorted Data

0	0	0	0	0	1	1	1	2	2	2	2	4	4

Couting Sort

```
CountingSort(input, k)
    for (i = 0; i < length(input); i++)
        j = input[i]
        count[j]++
    for (i = 1; i <= k; i++)
        count[i] = count[i] + count[i - 1]
    for (i = length(input) - 1; i >= 0; i--)
        j = input[i]
        count[j] --
        output[count[j]] = j
    return output
```



Couting Sort

Độ phức tạp là O(n+k), với k là khoảng từ min đến max giá trị

Không có độ ổn định

Phù hợp với dữ liệu có k nhỏ

Project objective: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do

Understanding the market

Market trends

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor

Client Implications:

- Incididunt ut labore et dolore
- Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor

Client Implications:

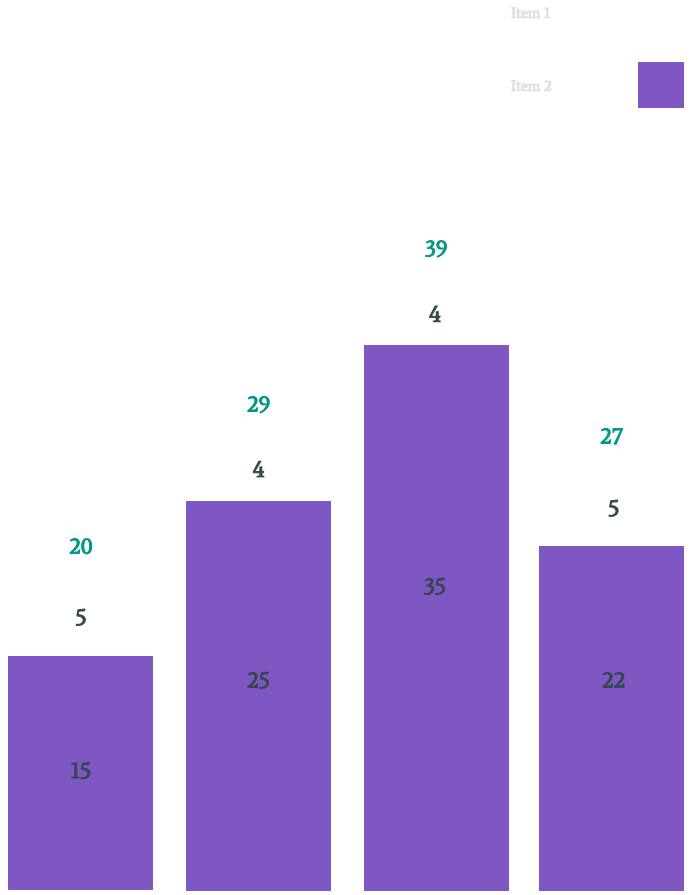
- Incididunt ut labore et dolore
- Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore

Trend analysis

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor

Client Implications:

- Incididunt ut labore et dolore
- Consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore



Proposed deliverables

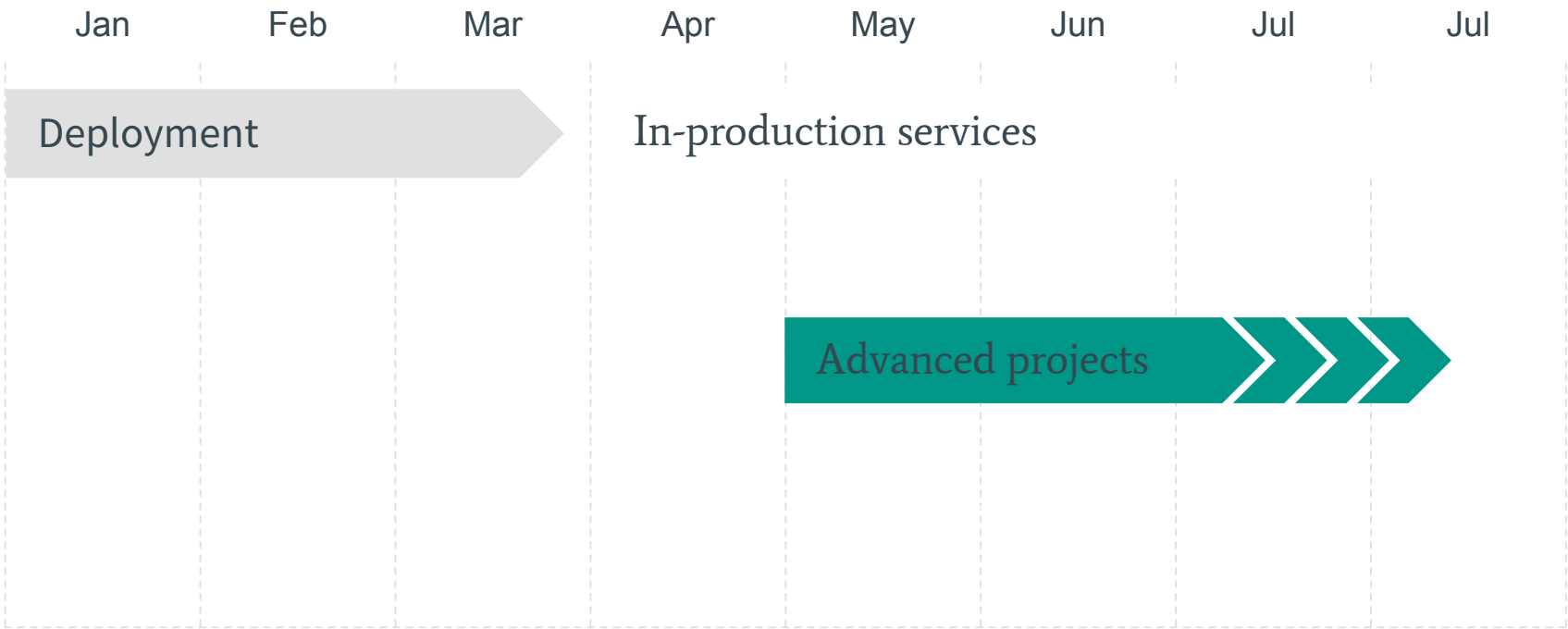
- Lorem ipsum dolor sit amet
- Sed do eiusmod tempor incididunt ut labore

- Lorem ipsum dolor sit amet
- Sed do eiusmod tempor incididunt ut labore

- Lorem ipsum dolor sit amet
- Sed do eiusmod tempor incididunt ut labore

- Lorem ipsum dolor sit amet
- Sed do eiusmod tempor incididunt ut labore

Timeline


Jan
Feb
Mar
Apr
May
Jun
Jul
Jul
Deployment
In-production services
Advanced projects

The Team

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur

Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum