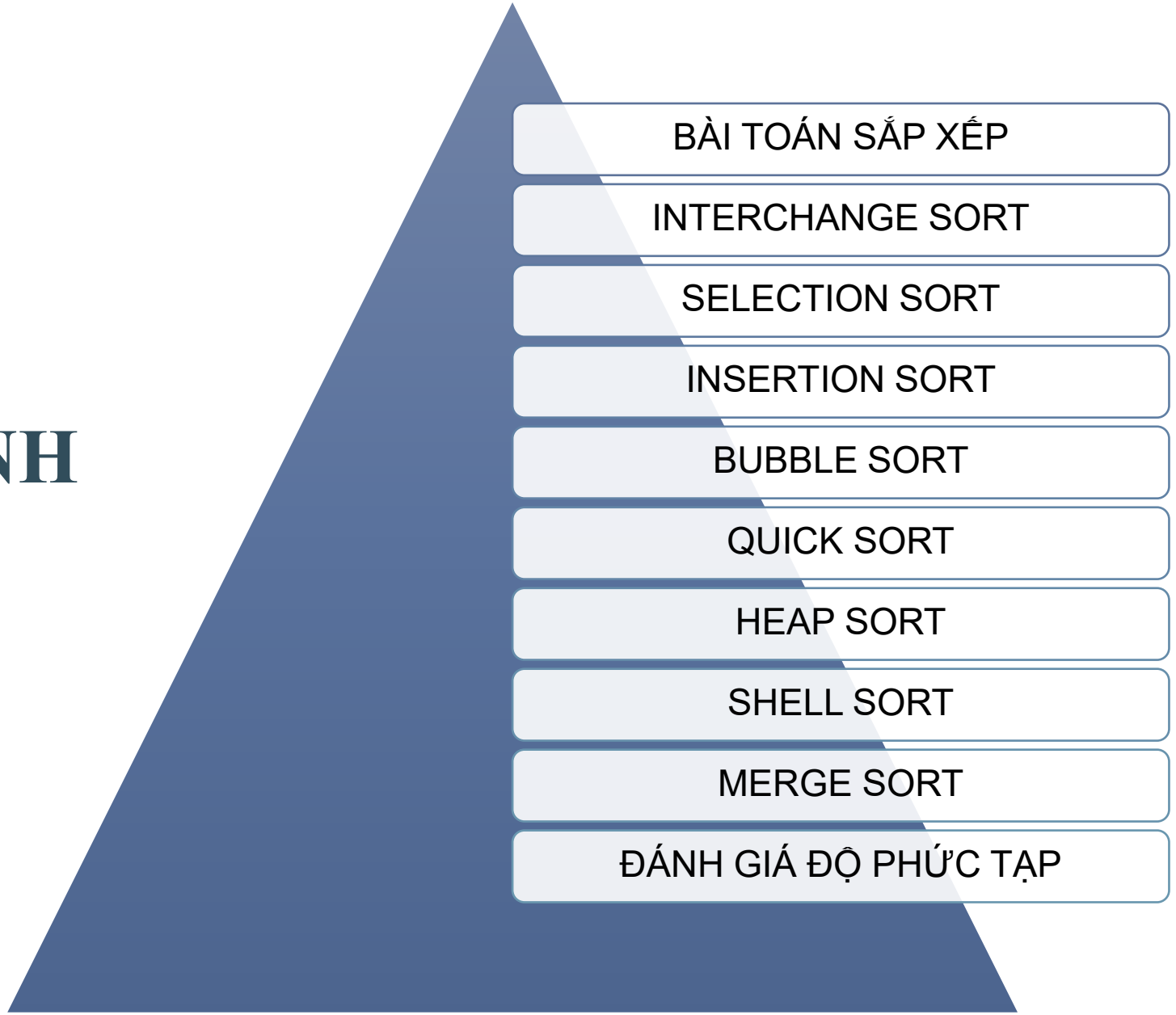
CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

CHƯƠNG 3

CÁC GIẢI THUẬT SẮP XẾP

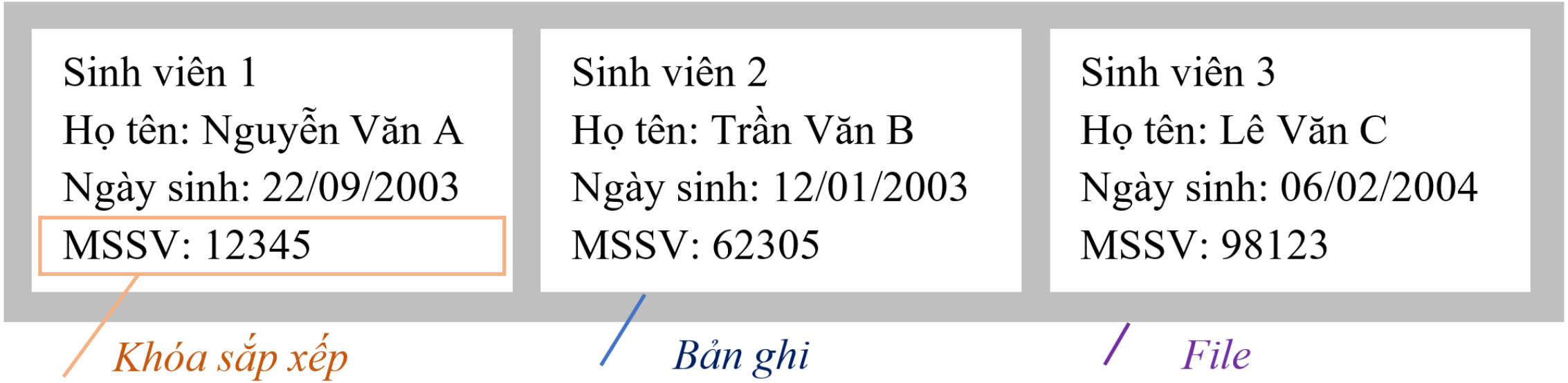
GV: Bùi Văn Thượng
Email: thuong.bui@ut.edu.vn

NỘI DUNG CHÍNH



1 BÀI TOÁN SẮP XẾP

Sắp xếp là quá trình xếp đặt các bản ghi của một file theo một thứ tự nào đó. Việc xếp đặt này được thực hiện dựa trên một hay nhiều trường nào đó của bản ghi và các thông tin này được gọi là khóa sắp xếp (key). Thứ tự các bản ghi được xác định dựa trên các khóa khác nhau và việc sắp xếp được thực hiện đối với mỗi khoá theo các thứ tự khác nhau. Hầu hết các thuật toán sắp xếp sử dụng hai thao tác cơ bản là so sánh và đổi chỗ các phần tử cần sắp xếp



1 BÀI TOÁN SẮP XẾP

Phân loại bài toán sắp xếp

- Sắp xếp trong (internal sorting): Dữ liệu sắp xếp được lưu đầy đủ trong bộ nhớ trong.
- Sắp xếp ngoài (external sorting): Dữ liệu cần sắp xếp có kích thước quá lớn và không thể lưu vào bộ nhớ trong để sắp xếp, các thao tác truy cập dữ liệu cũng mất nhiều thời gian hơn.
- Sắp xếp gián tiếp: Tạo ra file mới chứa các trường khóa của file ban đầu, sắp xếp trên file mới này với các bản ghi có kích thước nhỏ và sau đó truy cập vào các bản ghi trong file ban đầu thông qua các con trỏ hoặc chỉ số.

1 BÀI TOÁN SẮP XẾP

Các tiêu chuẩn đánh giá một thuật toán sắp xếp:

- Thời gian thực hiện: Số phép so sánh và hoán đổi.
- Bộ nhớ sử dụng: Phần bộ nhớ dùng cho việc thực hiện thuật toán.
- Tính ổn định: Không làm thay đổi thứ tự của các khóa bằng nhau.

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ý tưởng của giải thuật: Xuất phát từ đầu dãy, tìm tất cả các nghịch thế chứa phần tử này (hai phần tử gọi là **nghịch thế** nếu chúng có thứ tự ngược lại thứ tự cần sắp xếp), triệt tiêu chúng bằng cách đổi chỗ 2 phần tử trong cặp nghịch thế. Lặp lại quá trình xử lý trên với phần tử kế tiếp trong dãy cho tới khi hết phần tử để xét.

Các bước thực hiện:

- **Bước 1:** i = 0; // bắt đầu từ đầu dãy
- **Bước 2:** j = i+1; //tìm các nghịch thế với a[i]
- **Bước 3:** Trong khi j < n thực hiện lặp lại quá trình sau:
  - *Nếu a[j]<a[i] thì đổi chỗ a[i], a[j]*
  - *j = j+1;*
- **Bước 4:** i = i+1; Nếu i < n-1 quay lại bước 2. Ngược lại thì dừng.

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

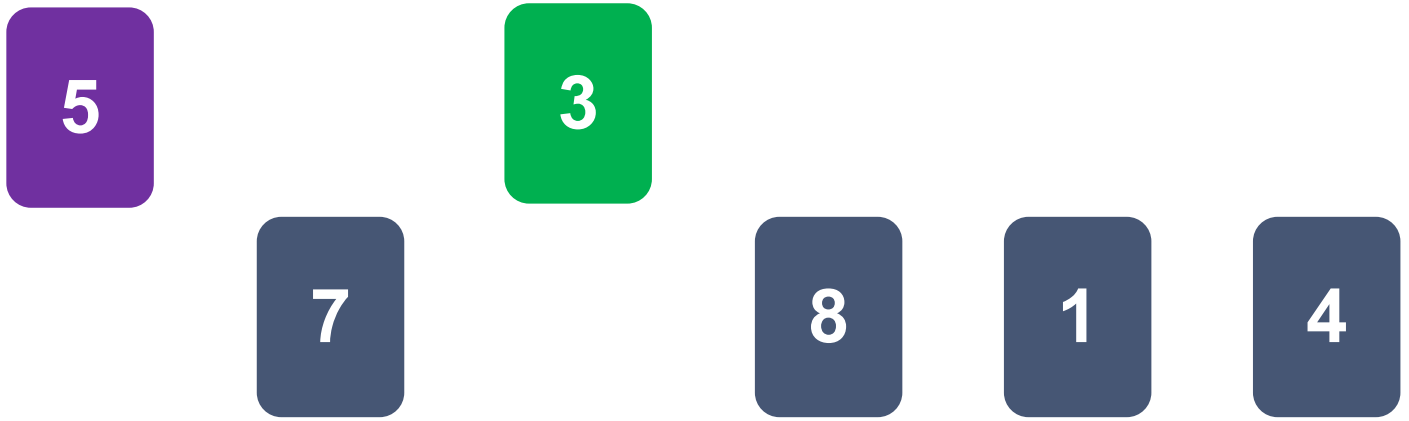
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

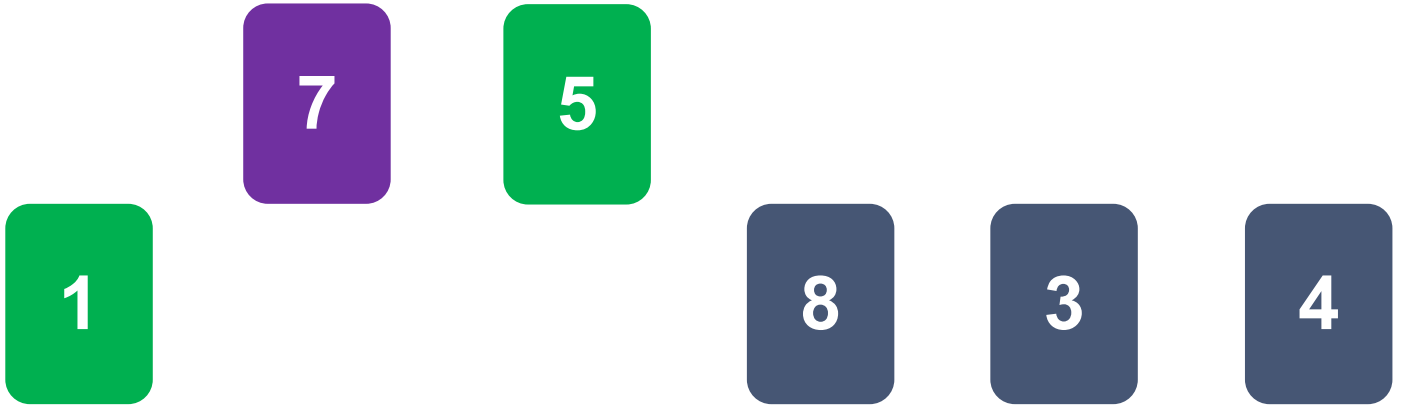
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

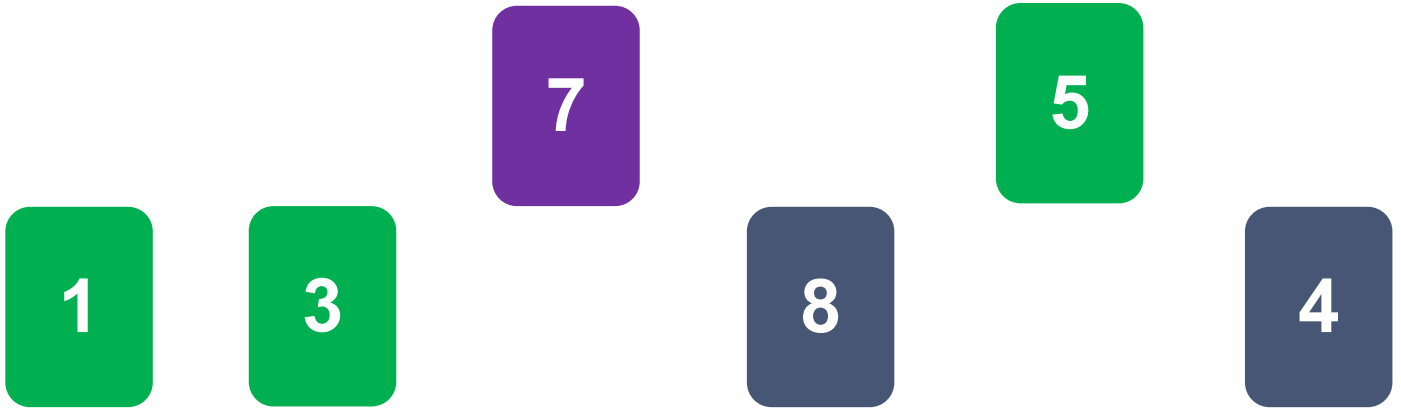
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

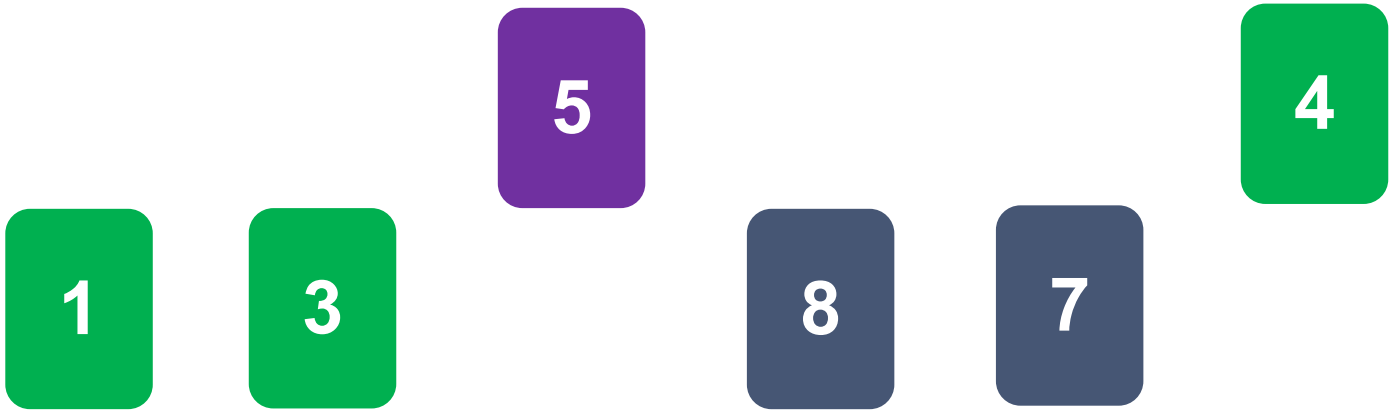


2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

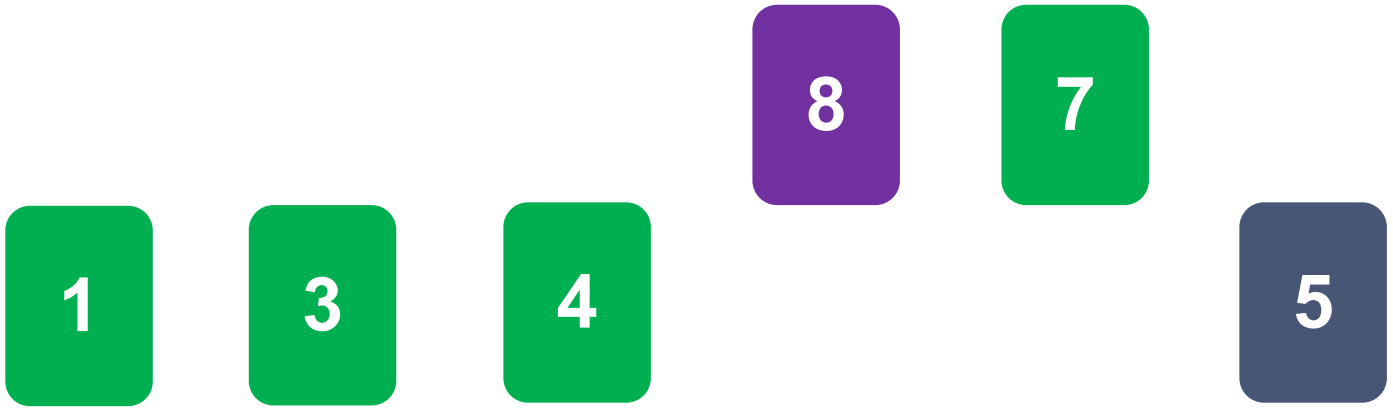
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

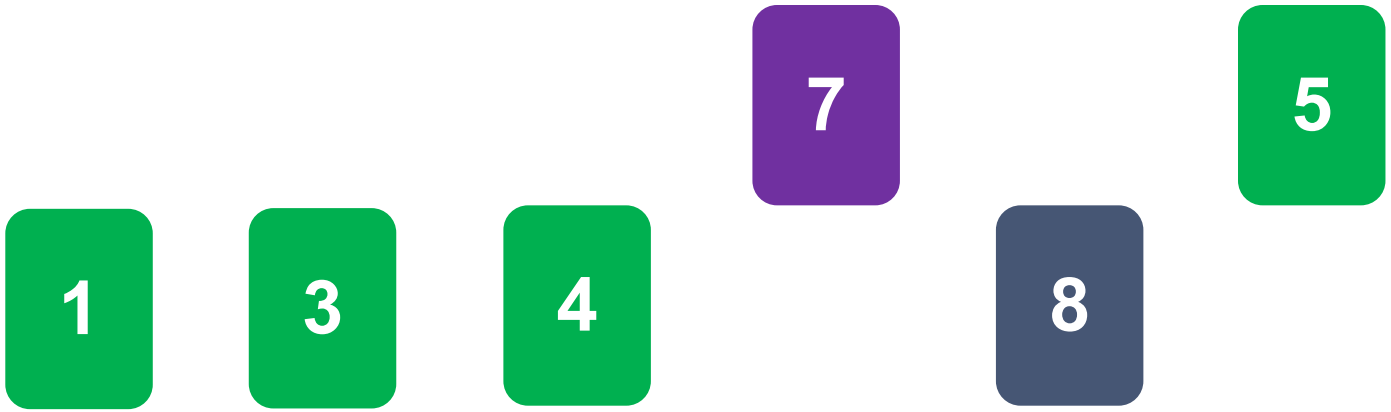


2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

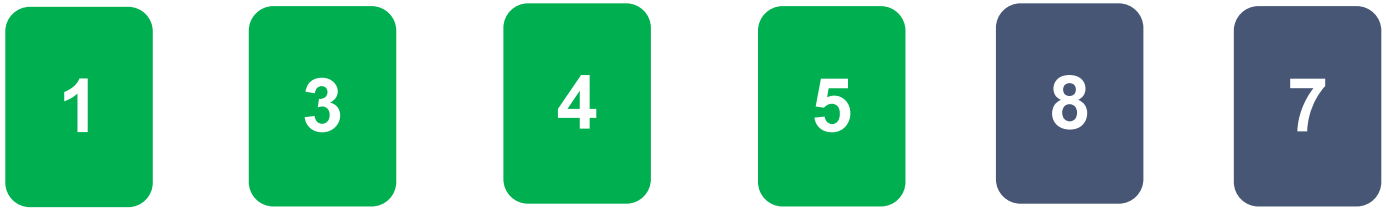
2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



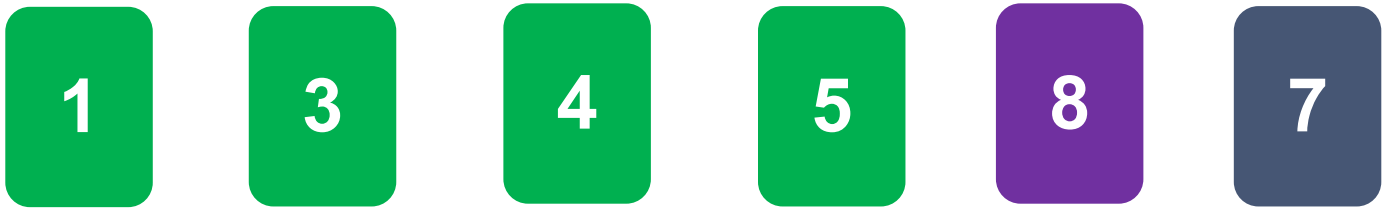
2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



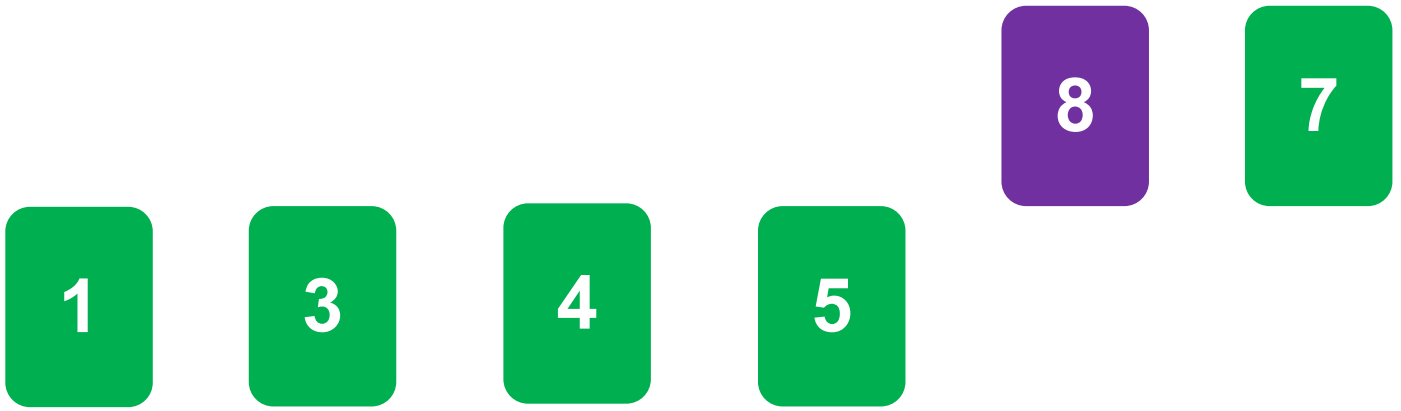
2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



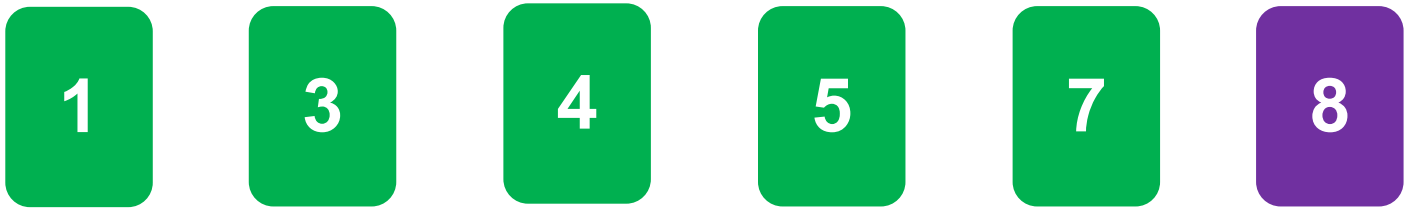
2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



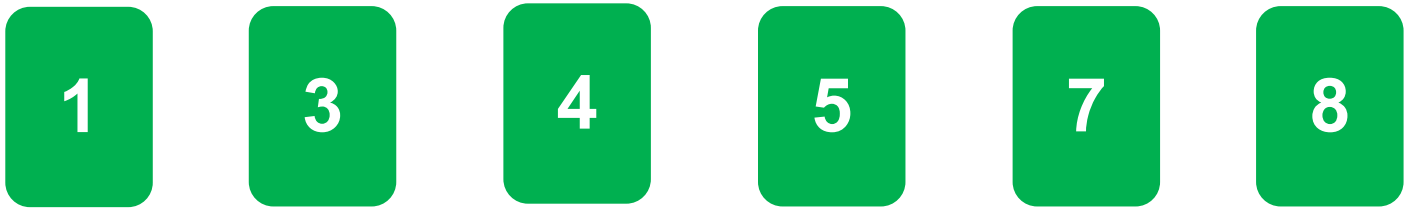
2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



2 ĐỔI CHỖ TRỰC TIẾP – INTERCHANGE SORT

Hàm minh họa thuật toán đổi chỗ trực tiếp:

```
void InterchangeSort (int a[], int n) {
        int i, j;
        for (i = 0; i<n-1; i++)
                for (j = i+1; j<n; j++)
                        if (a[j]<a[i]) {//Đổi chỗ a[i] và a[j]
                                int t = a[i];
                                a[i] = a[j];
                                a[j]= t;
                        }
}
```

3 CHỌN TRỰC TIẾP – SELECTION SORT

Ý tưởng của giải thuật: Tìm và đổi vị trí phần tử nhỏ nhất với phần tử đầu tiên trong dãy. Lặp lại tương tự quá trình trên đối với các phần tử thứ 2, 3, ..., n-1.

Các bước thực hiện:

Bước 1: i = 0.

Bước 2: Tìm phần tử a[min] nhỏ nhất trong dãy hiện hành từ a[i] đến a[n-1].

Bước 3: Đổi chỗ a[min] và a[i].

Bước 4: Nếu i < n-1 thì i = i+1 và lặp lại từ bước 2. Ngược lại thì dừng thuật toán.

3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

3 CHỌN TRỰC TIẾP – SELECTION SORT

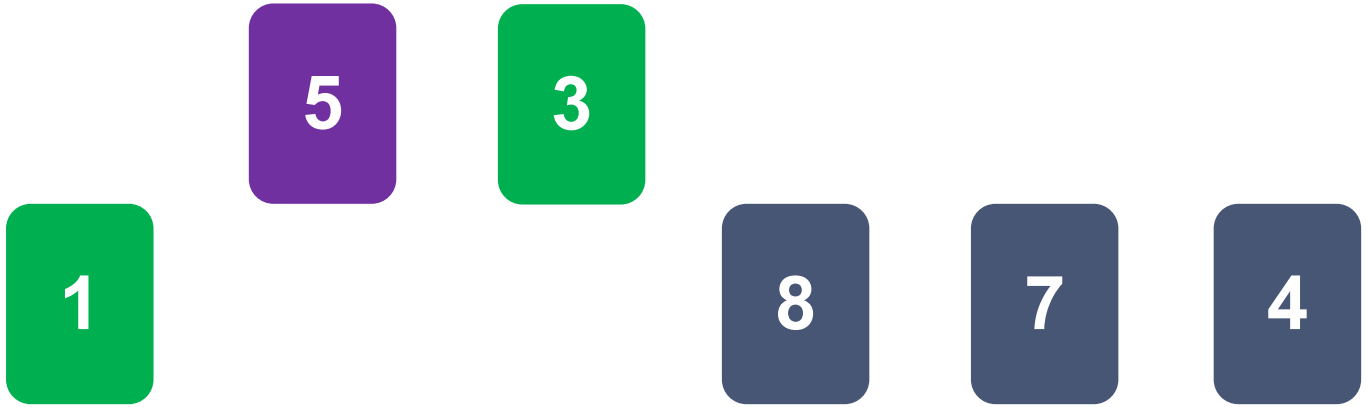
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

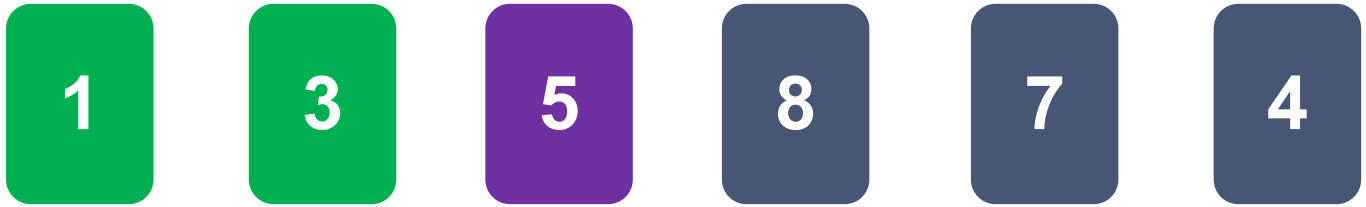
3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



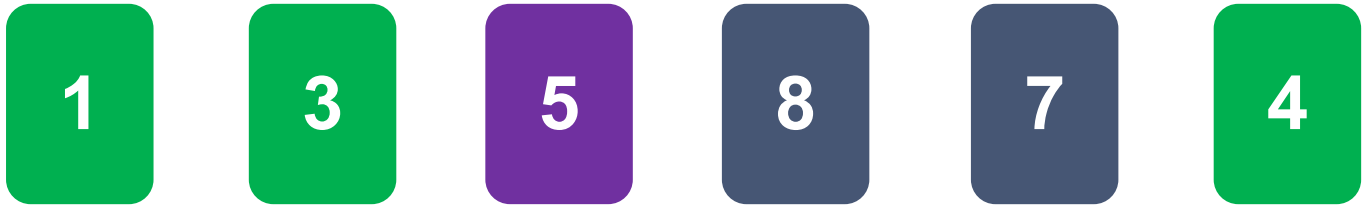
3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



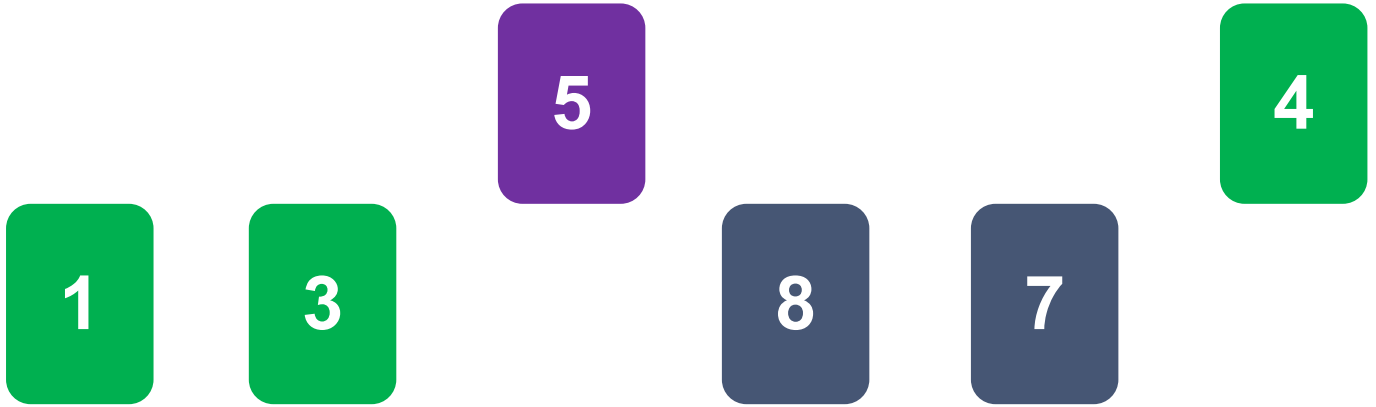
3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



3 CHỌN TRỰC TIẾP – SELECTION SORT

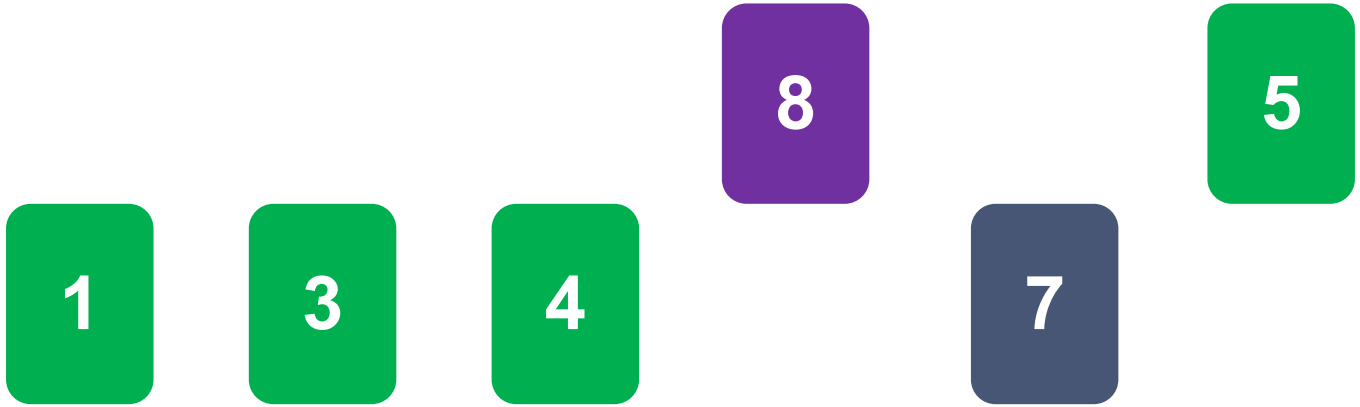
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

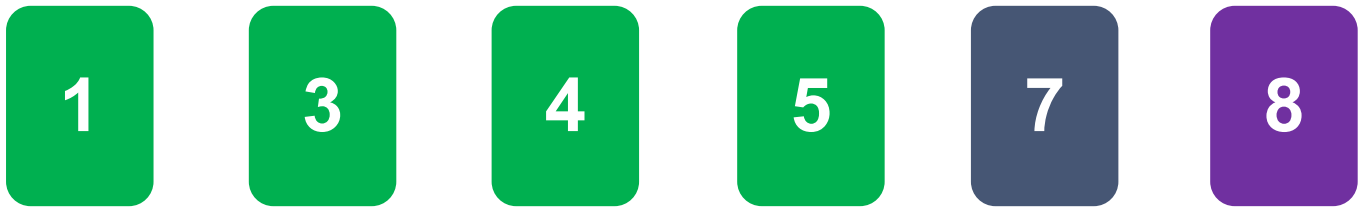
3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



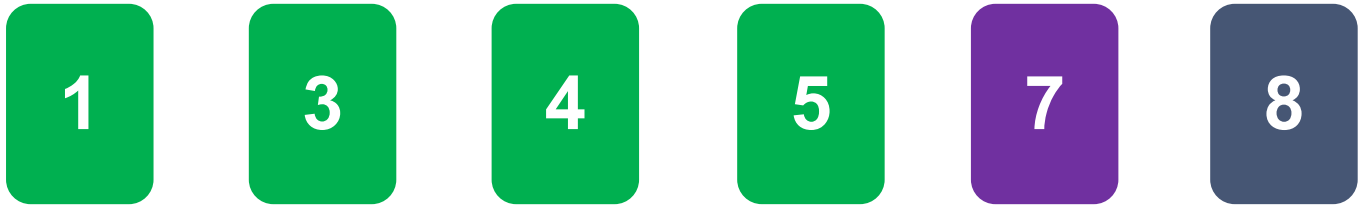
3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



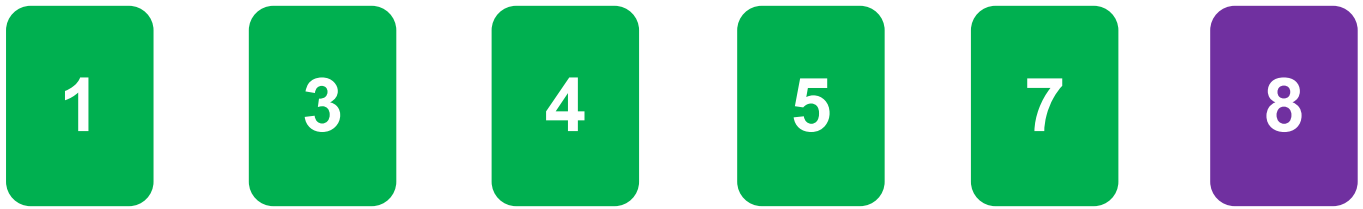
3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



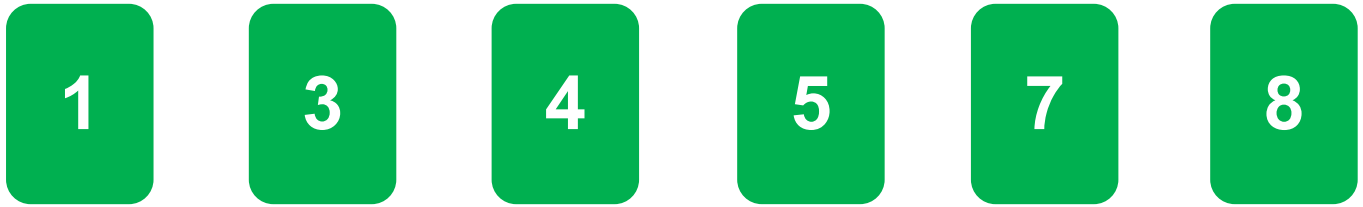
3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



3 CHỌN TRỰC TIẾP – SELECTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



3 CHỌN TRỰC TIẾP – SELECTION SORT

Hàm minh họa thuật toán chọn trực tiếp:

```
void SelectionSort (int a[], int n){
        int min, i, j; //min dùng để lưu vị trí của phần tử nhỏ
nhất
        for (i = 0; i<n -1; i++) {
                min = i;
                for (j = i+1; j <n; j++)
                        if (a[j] < a[min])  min = j;
                int t = a[i]; //Đổi chỗ a[i] và a[min] cho nhau
                a[i]= a[min];
                a[min]= t;
        }
}
```

4 CHÈN TRỰC TIẾP – INSERTION SORT

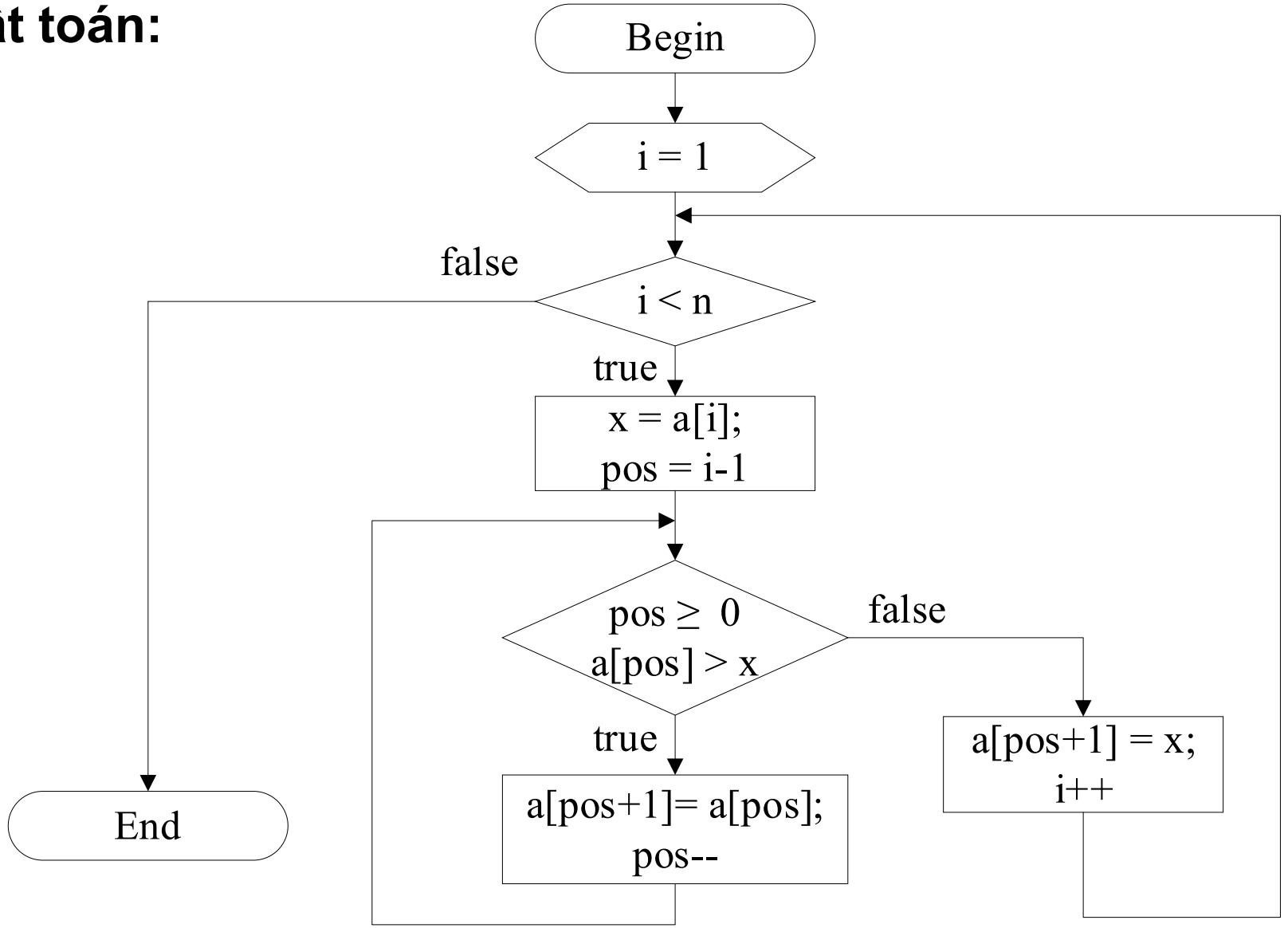
Ý tưởng của giải thuật: Ban đầu coi như phần tử đầu tiên đã được sắp xếp. Sau đó so sánh phần tử thứ 2 với phần tử đầu tiên để đổi chỗ nếu cần, để sao cho 2 phần tử này được sắp xếp theo thứ tự chỉ định.Tiếp theo, tìm vị trí để chèn phần tử thứ 3 của dãy hiện hành vào dãy 2 phần tử đầu đã được sắp xếp ở trên sao cho dãy vẫn được sắp xếp đúng thứ tự. Quá trình trên lặp lại liên tục cho tới khi hết dãy.

Các bước thực hiện:

- **Bước 1:** i = 1; //Giả sử có đoạn a[0] đã được sắp.
- **Bước 2:** x = a[i]; position = i-1; //Tìm vị trí position thích hợp để chèn a[i] vào.
- **Bước 3:** Tìm vị trí position thích hợp để chèn a[i]. Sau đó dời chỗ các phần tử từ a[position] đến a[i-1] sang phải 1 vị trí.
- **Bước 4:** a[position] = x; //Chèn a[i] vào vị trí tìm được.
- **Bước 5:** i = i+1; Nếu i < n thì lặp lại bước 2, ngược lại dừng thuật toán.

4 CHÈN TRỰC TIẾP – INSERTION SORT

Lưu đồ thuật toán:



4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

4 CHÈN TRỰC TIẾP – INSERTION SORT

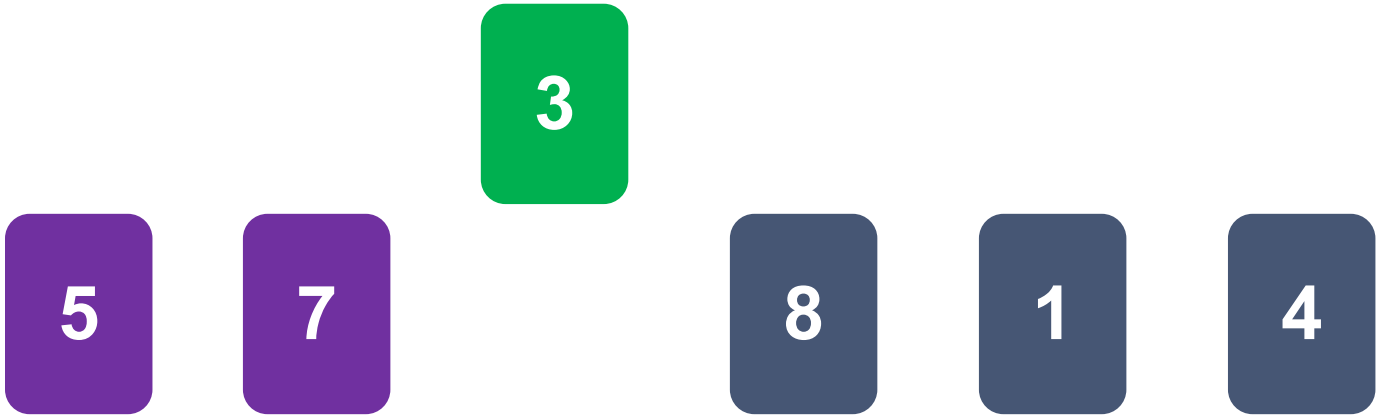
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

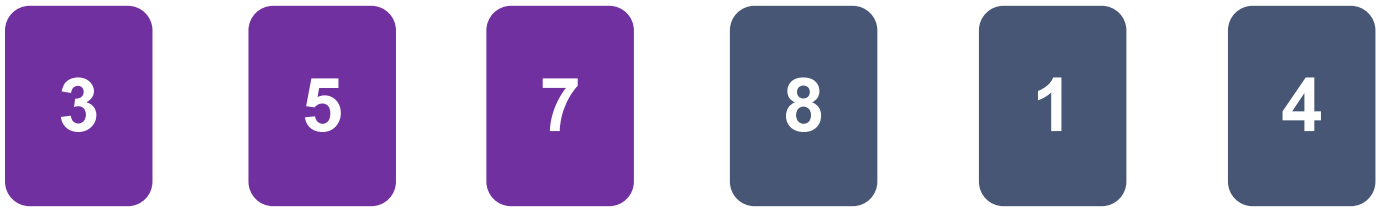


4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

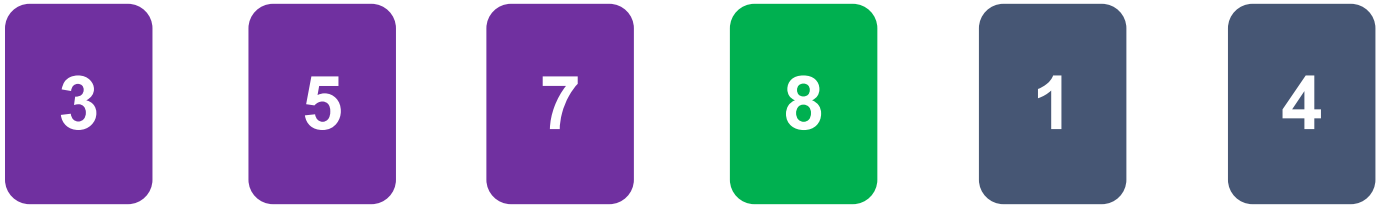
4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



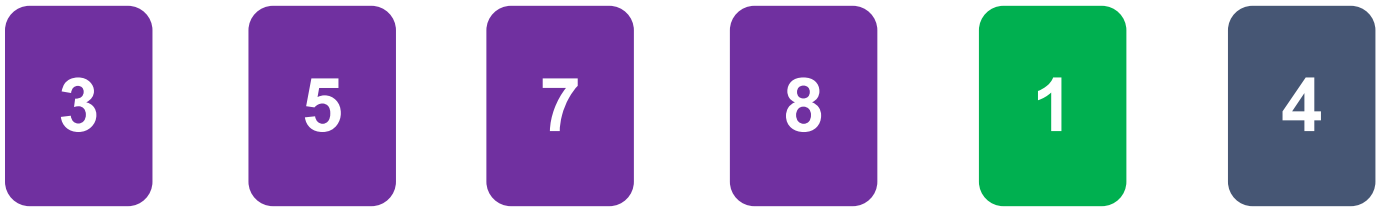
4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

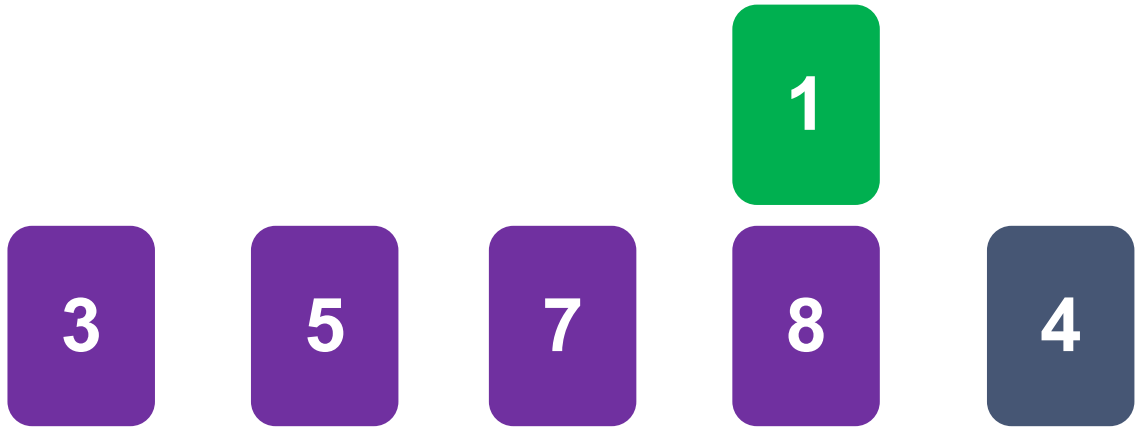


4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

4 CHÈN TRỰC TIẾP – INSERTION SORT

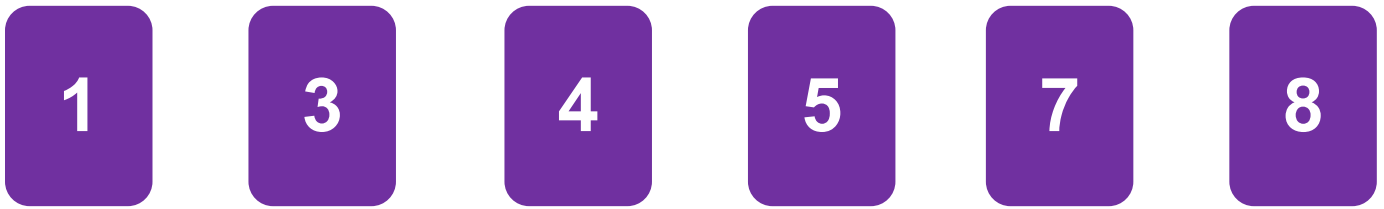
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

4 CHÈN TRỰC TIẾP – INSERTION SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



4 CHÈN TRỰC TIẾP – INSERTION SORT

Hàm minh họa giải thuật sắp xếp chèn trực tiếp:

```
void InsertionSort (int a[], int n) {
        int pos, i, x;
        for (i=1; i<n; i++) {
                x = a[i];   pos = i - 1;
                while (pos >= 0 && a[pos] > x) {
                        a[pos+1] = a[pos];  pos--;
                }
                a[pos+1] = x;
        }
}
```

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ý tưởng của giải thuật:

- Xuất phát từ cuối dãy, ta so sánh 2 phần tử kế cận nhau. Nếu 2 phần tử đó có giá trị ngược với thứ tự cần sắp xếp thì ta tiến hành đổi chỗ của chúng cho nhau.
- Tiếp tục so sánh và đổi chỗ các phần tử cận kề cho tới đầu dãy, khi đó phần tử đầu dãy tìm được đã nằm đúng vị trí theo thứ tự cần sắp xếp.
- Lặp lại các bước trên với những phần từ còn lại chưa nằm đúng vị trí cho tới khi không còn phần tử để xét. Khi đó dãy đã được sắp xếp như ý.

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Các bước thực hiện:

Bước 1: i = 0; // Lần xử lý đầu tiên.

Bước 2: j = n-1; //Duyệt từ cuối dãy ngược về vị trí i.

Trong khi j > i thực hiện:

- *Nếu a[j]<a[j-1] thì tiến hành đổi chỗ a[j], a[j-1] cho nhau.*
- *j = j-1;*

Bước 3: i = i+1; // Lần xử lý kế tiếp

- *Nếu i = n -1 thì giải thuật dừng.*
- *Ngược lại ta lặp lại bước 2.*

Ghi chú:

Giải thuật sắp xếp Bubble Sort có thể cài đặt theo hướng duyệt từ đầu dãy tới cuối dãy hoặc ngược lại.

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

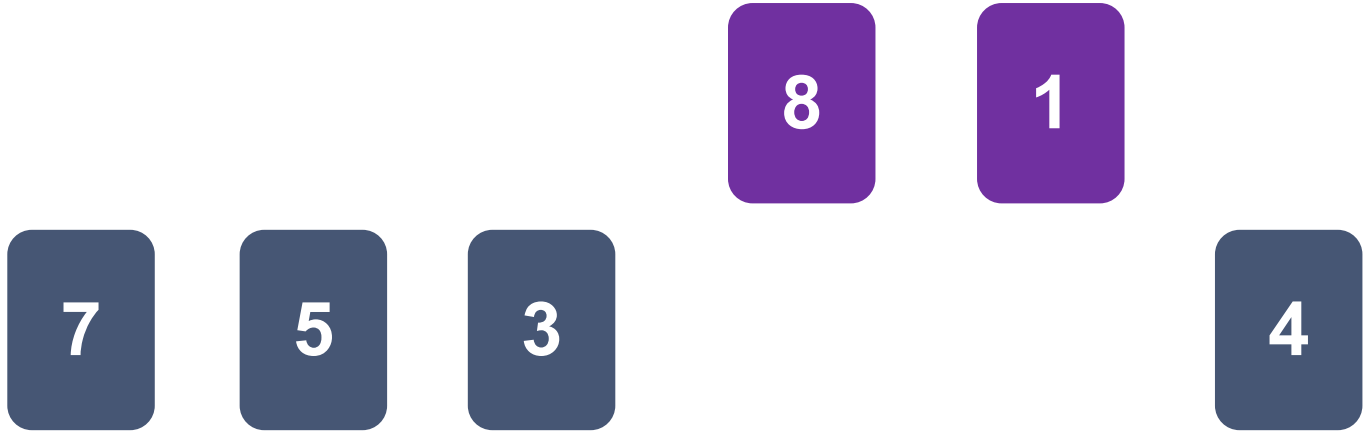
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

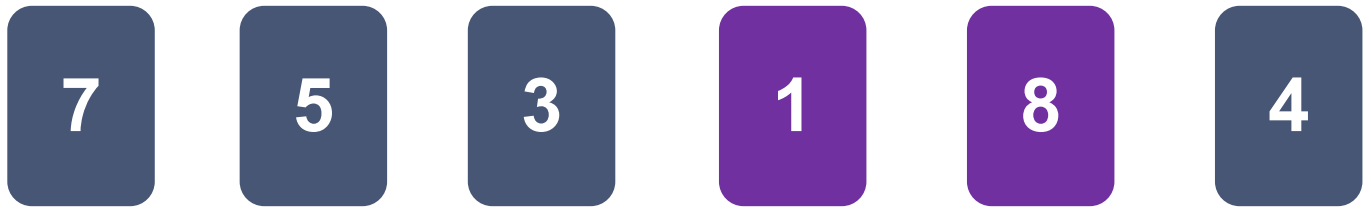
5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



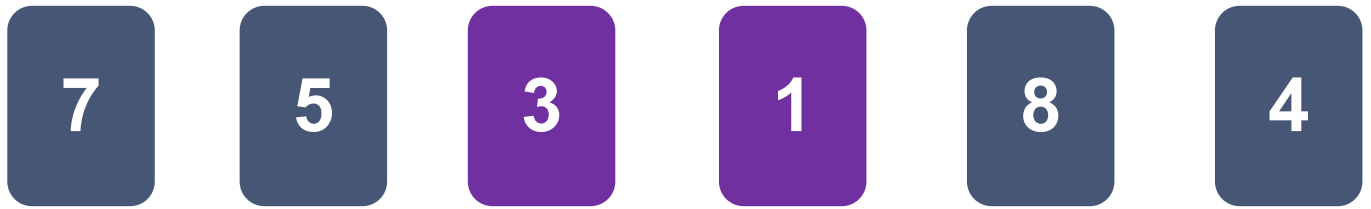
5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



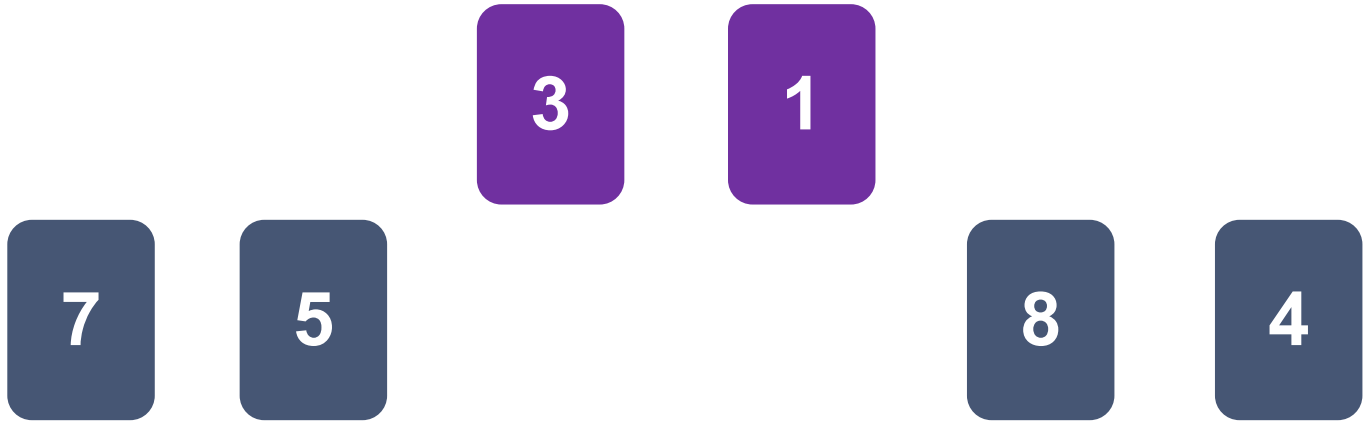
5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



5 SẮP XẾP NỔI BỌT – BUBBLE SORT

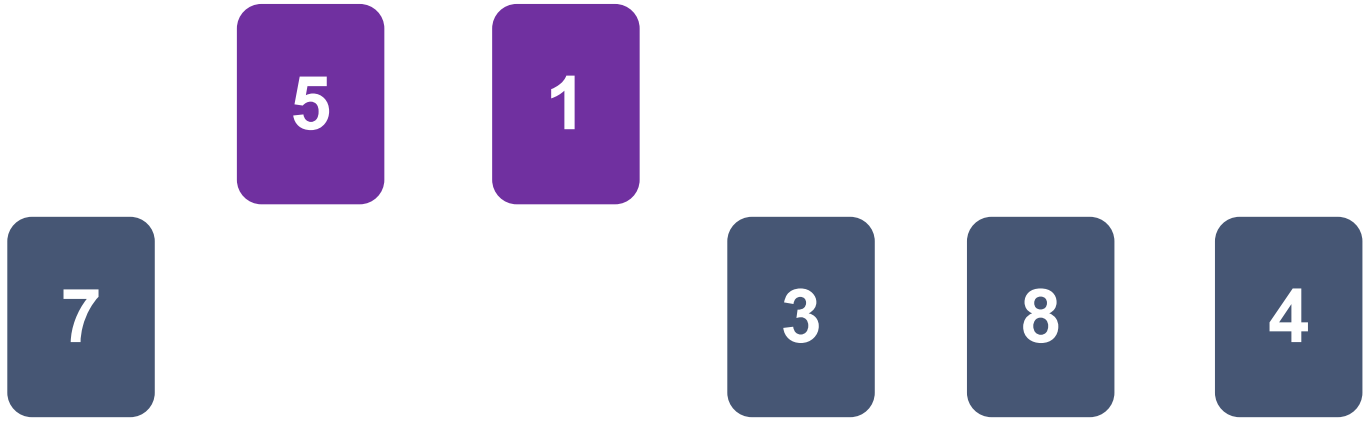
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



5 SẮP XẾP NỔI BỌT – BUBBLE SORT

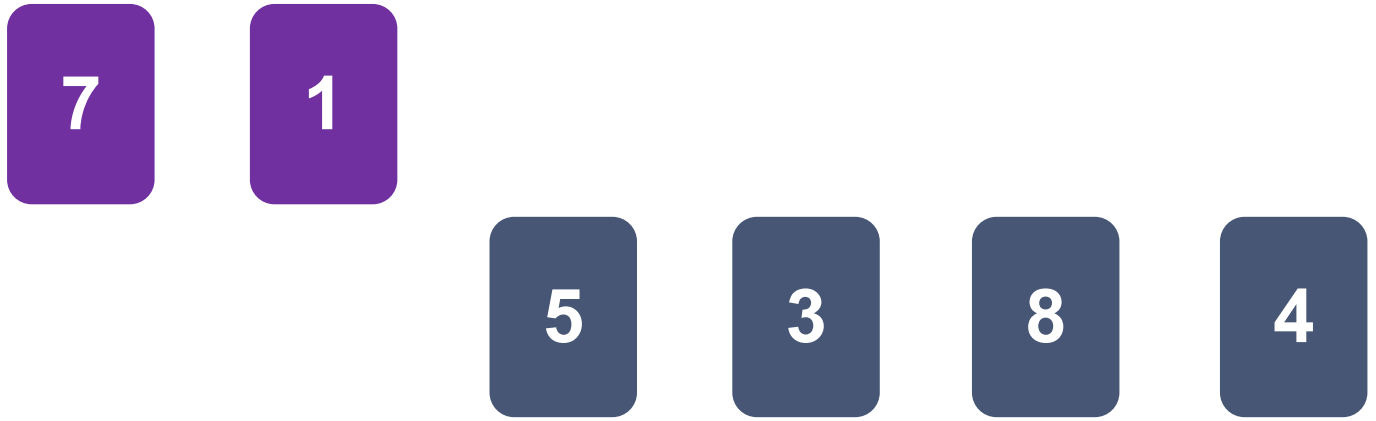
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



5 SẮP XẾP NỔI BỌT – BUBBLE SORT

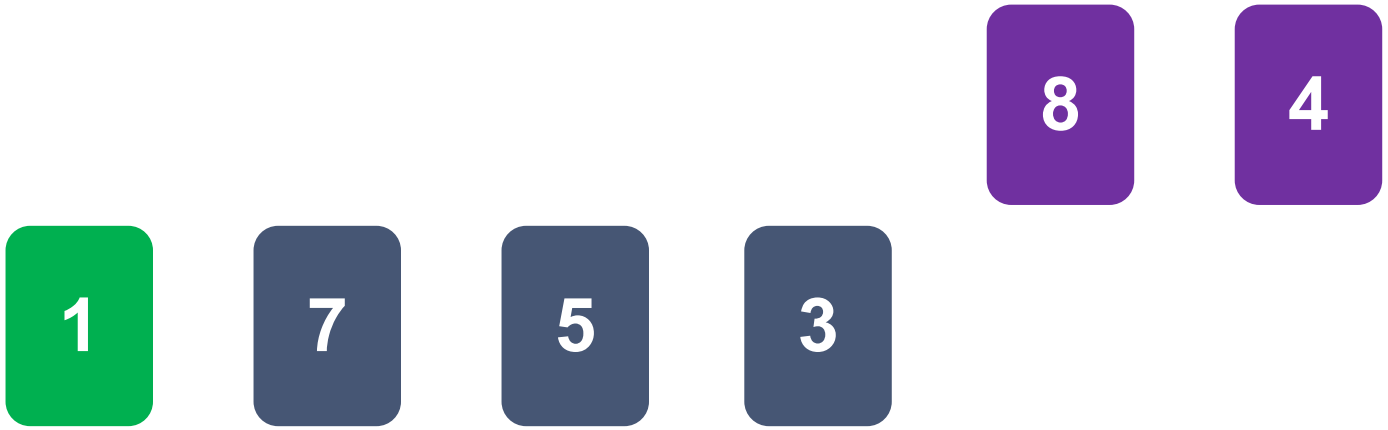
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



5 SẮP XẾP NỔI BỌT – BUBBLE SORT

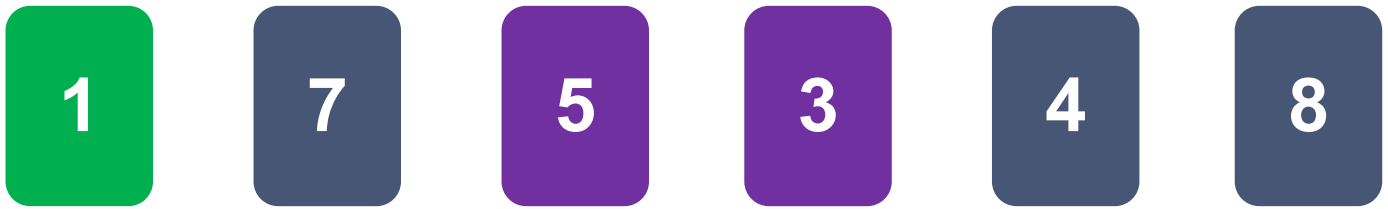
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

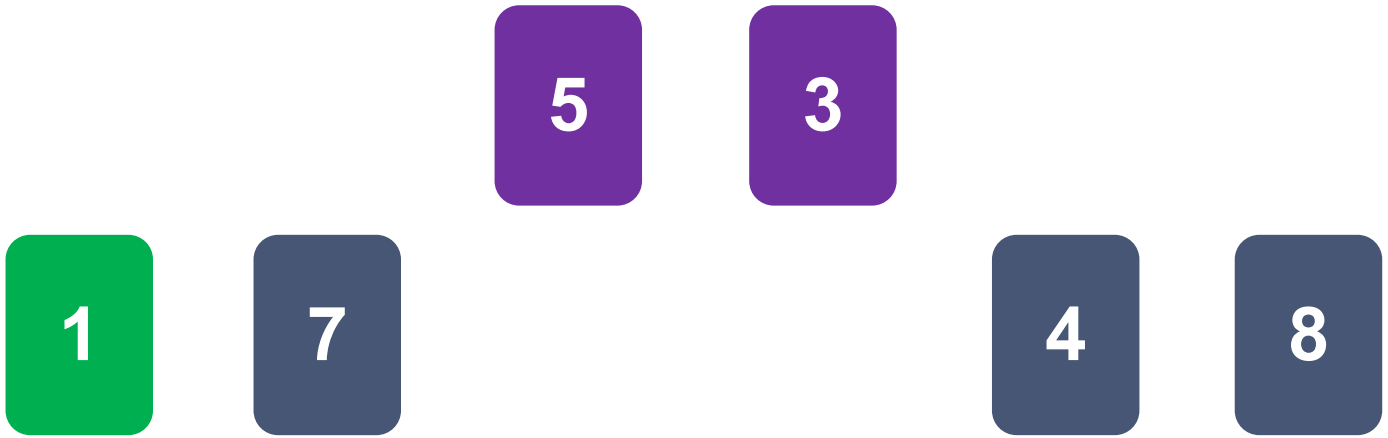
5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



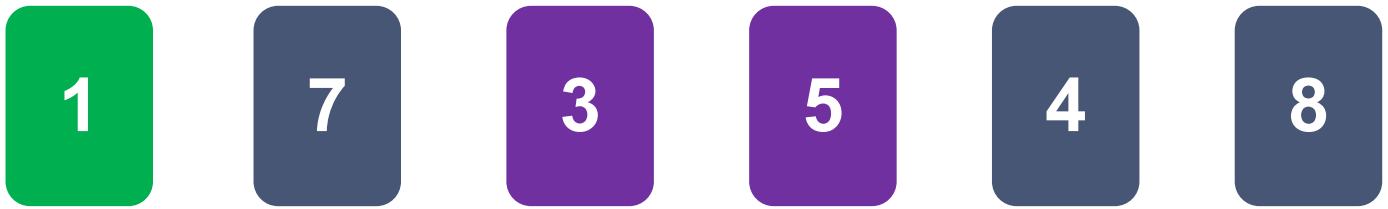
5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



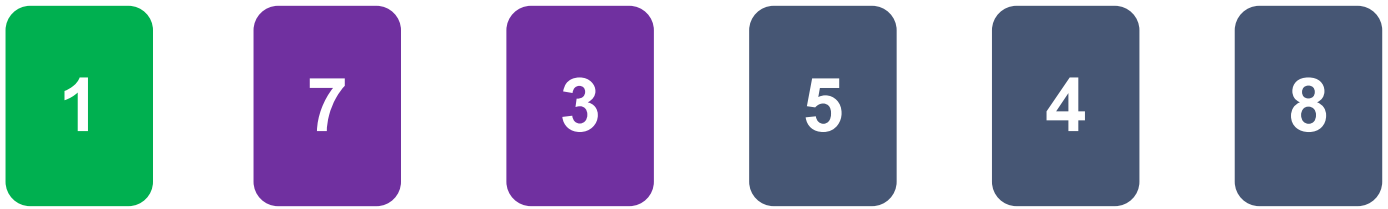
5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



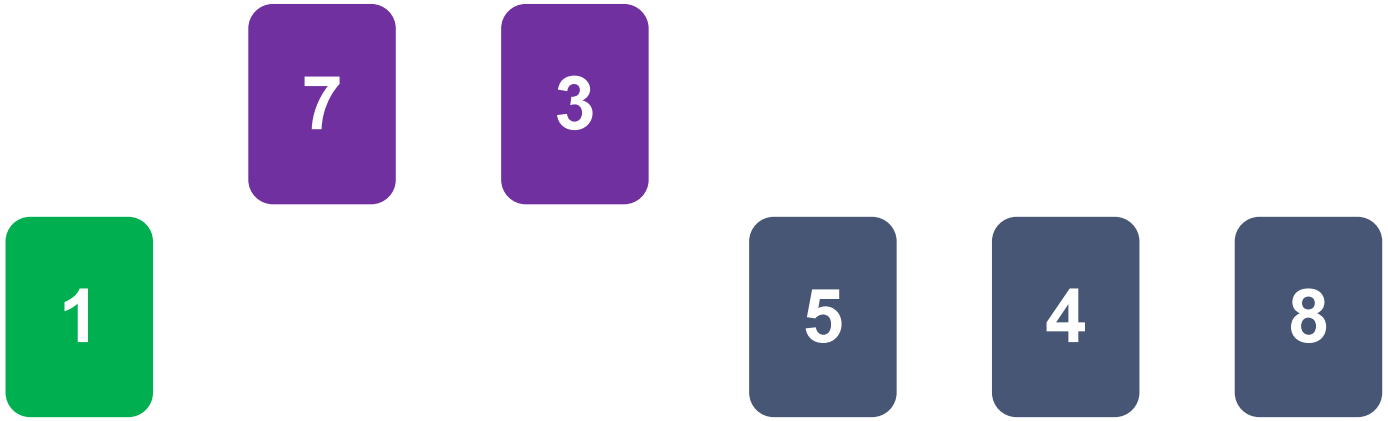
5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

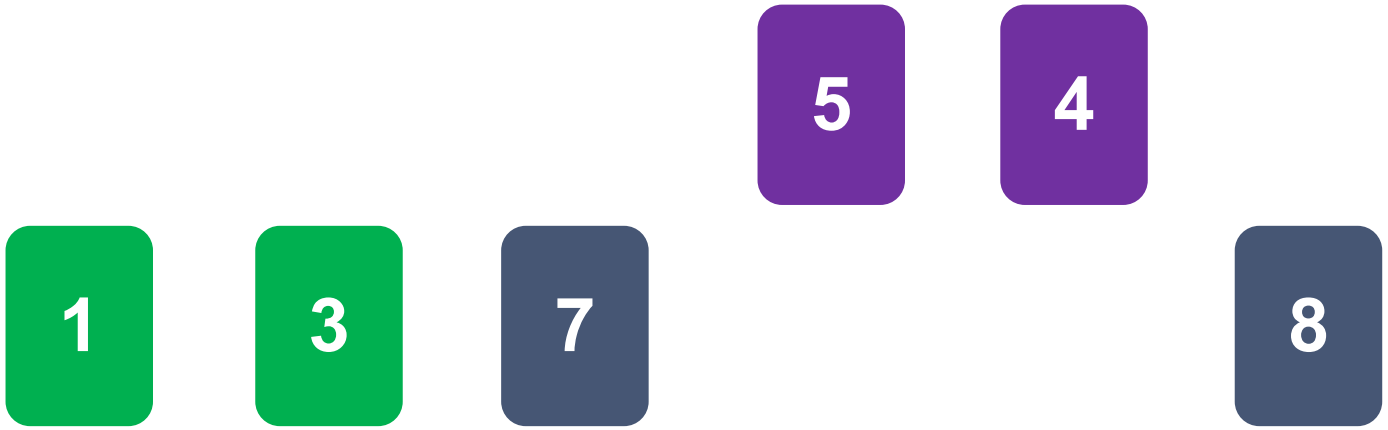
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

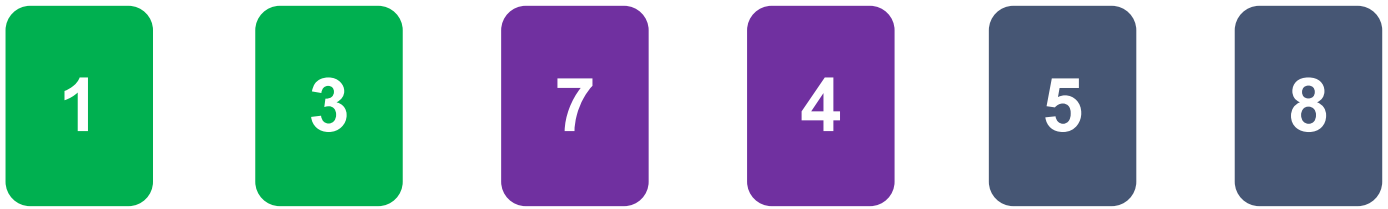


5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

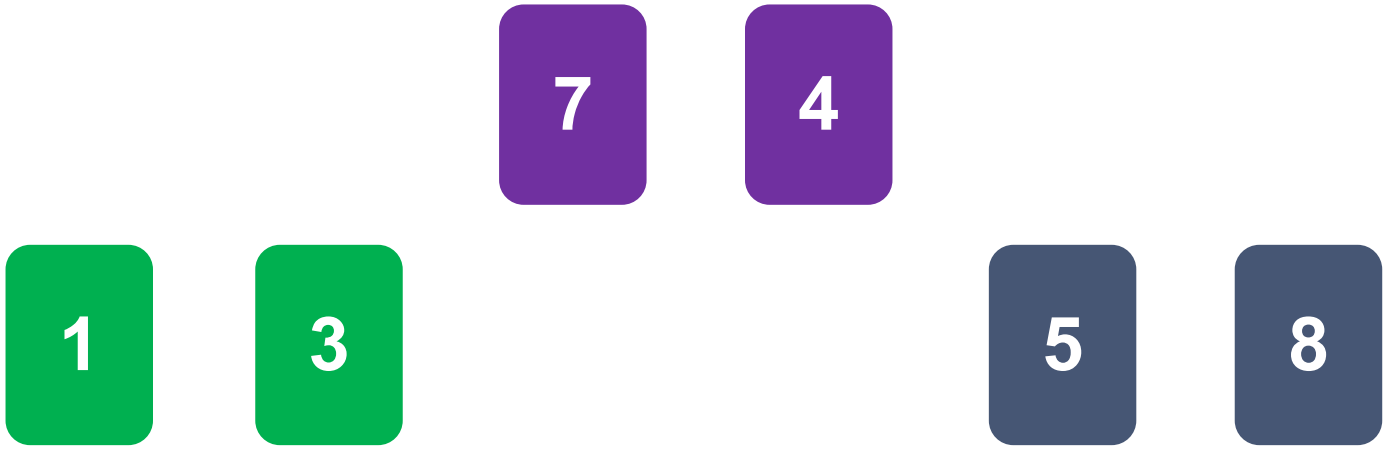
5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



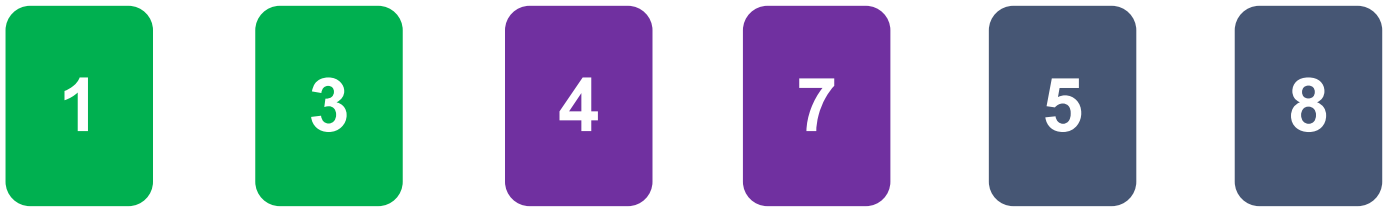
5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



5 SẮP XẾP NỔI BỌT – BUBBLE SORT

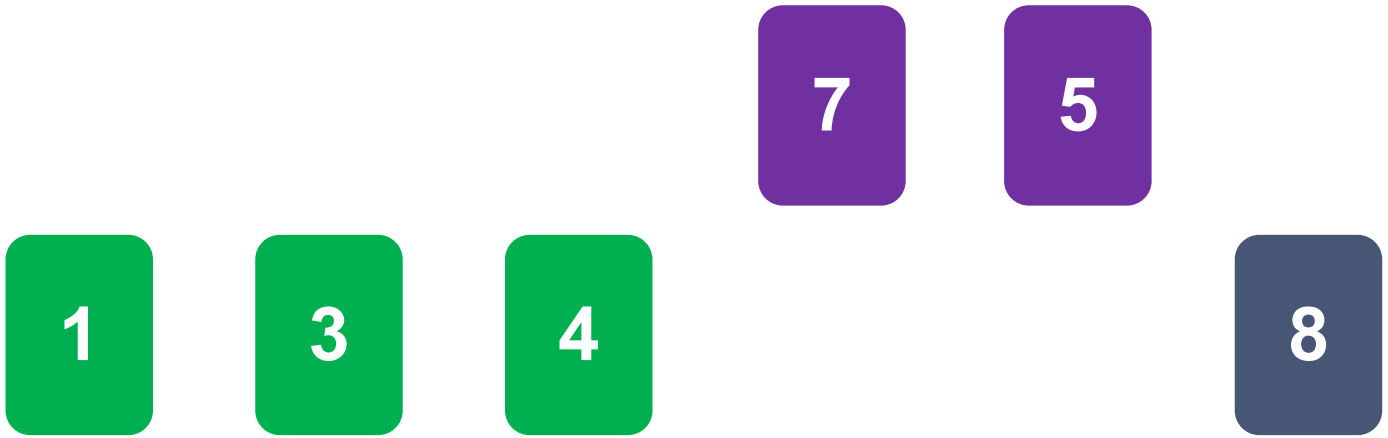
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

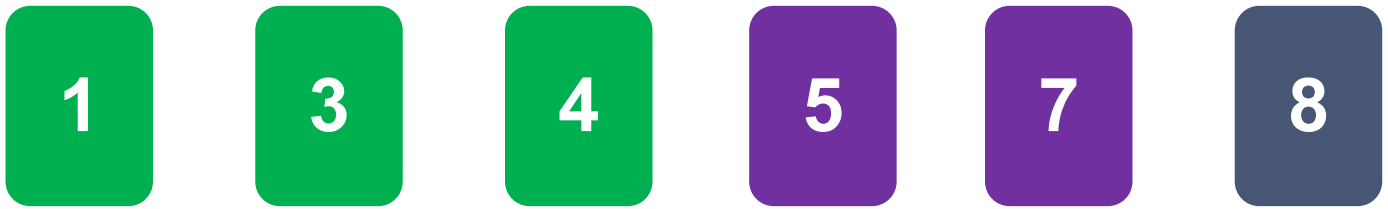
5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



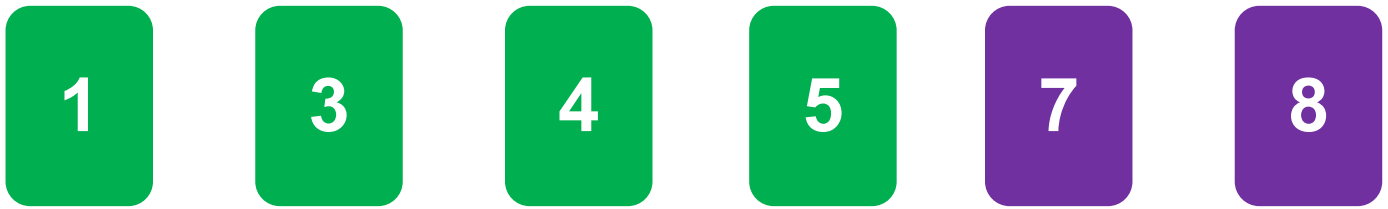
5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



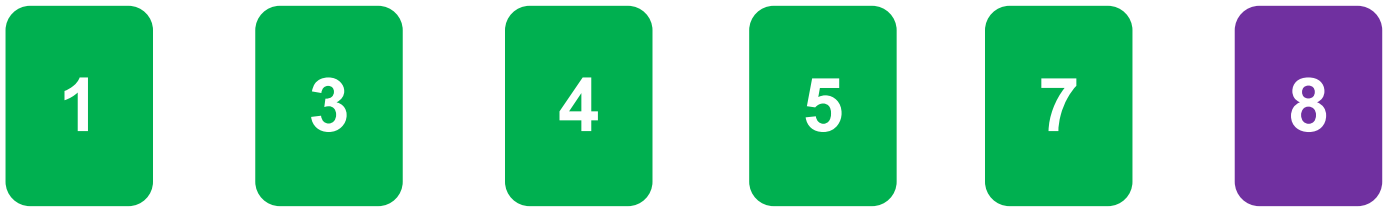
5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



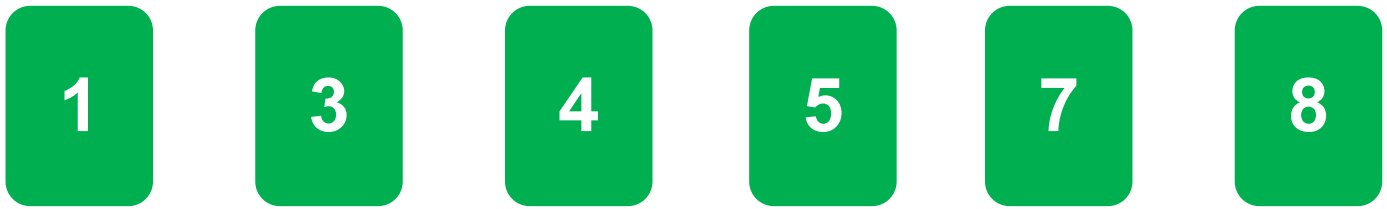
5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



5 SẮP XẾP NỔI BỌT – BUBBLE SORT

Hàm mình họa giải thuật sắp xếp nổi bọt

```
void BubbleSort (int a[], int n) {
        int i, j;
        for (i = 0; i<n-1; i++)
                for (j = n-1; j >i; j --)
                        if(a[j]< a[j-1]) {
                                int t = a[j];
                                a[j] = a[j-1];
                                a[j-1] = t;
                        }
}
```

6 SẮP XẾP NHANH – QUICK SORT

Quick Sort là một trong những thuật toán sắp xếp được xem là nhanh nhất trong các giải thuật sắp xếp. Thuật toán Quick Sort có thể được hiểu bằng cách chia để trị (Divide and Conquer).

Ý tưởng của giải thuật sắp xếp nhanh - Quick Sort:

- *Quick Sort chia mảng thành hai danh sách bằng cách so sánh từng phần tử của danh sách với một phần tử được chọn được gọi là phần tử chốt.*
- *Những phần tử nhỏ hơn phần tử chốt được đưa về phía trước và nằm trong danh sách con thứ nhất, các phần tử lớn hơn hoặc bằng phần tử chốt được đưa về phía sau và thuộc danh sách con thứ hai.*
- *Sau đó, thuật toán được áp dụng đệ quy cho cả hai danh sách con. Cứ tiếp tục chia các danh sách con như vậy tới khi các danh sách con đều có độ dài bằng 1.*

6 SẮP XẾP NHANH – QUICK SORT

Các bước thực hiện:

Bước 1: Lấy 1 phần tử bất kì của dãy làm phần tử chốt X (giả sử lấy phần tử đầu tiên hoặc phần tử giữa của dãy, …).

Bước 2: Tiến hành duyệt từ bên trái dãy cho đến khi gặp phần tử lớn hơn hoặc bằng X thì dừng. Đồng thời duyệt từ bên phải dãy cho đến khi gặp phần tử nhỏ hơn hoặc bằng X thì dừng.

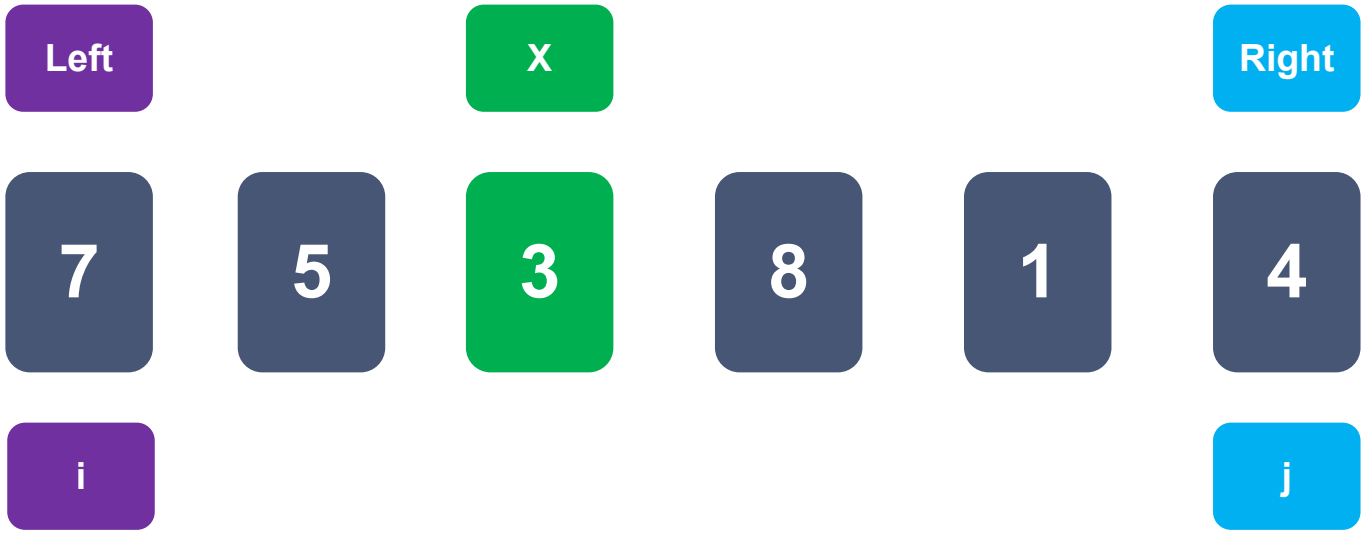
Bước 3: Đổi chỗ 2 phần tử tìm được ở bước 2.

Bước 4: Tiếp tục duyệt cho đến khi 2 biến duyệt từ 2 chiều gặp nhau. Khi đó dãy ban đầu đã phân thành 2 phần: phần trái gồm những phần tử nhỏ hơn X, phần phải gồm những phần tử lớn hơn hoặc bằng X.

Bước 5: Lặp lại quá trình phân hoạch đối với phần trái và phần phải mới được tạo ở trên cho đến khi nào dãy được sắp xếp hoàn toàn.

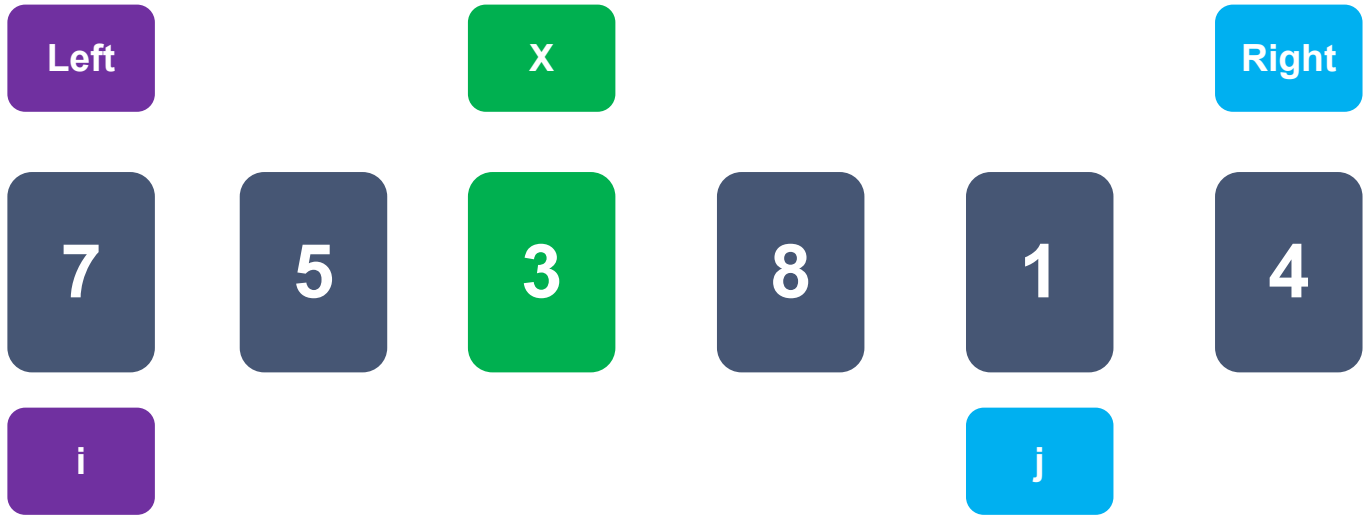
6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

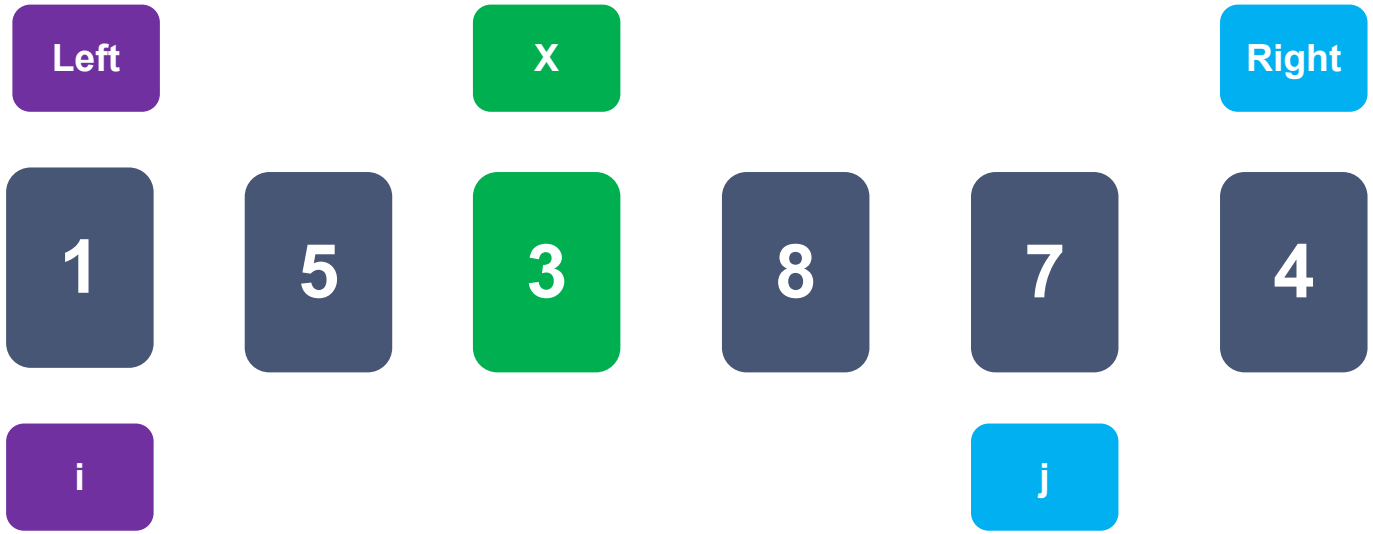


6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

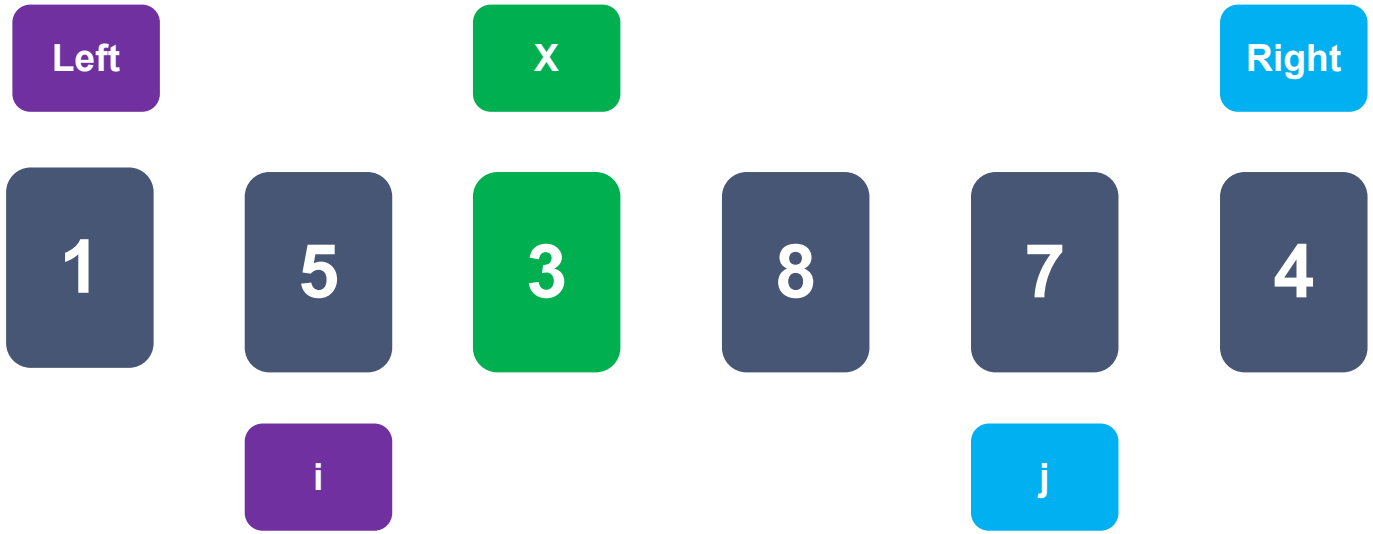
6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



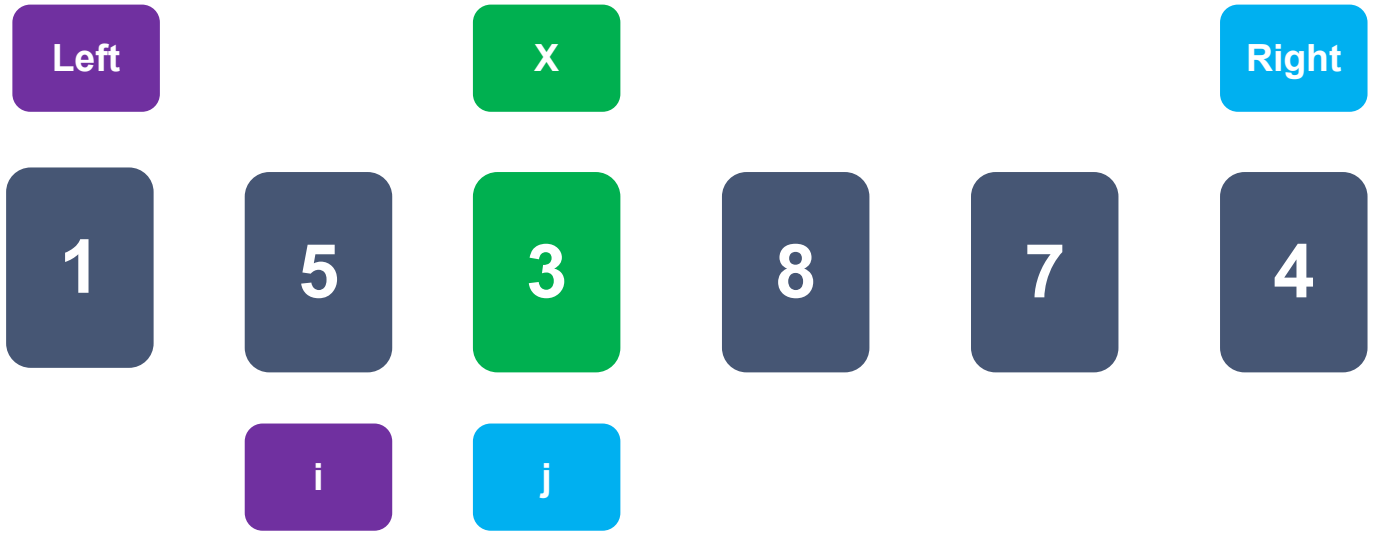
6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



6 SẮP XẾP NHANH – QUICK SORT

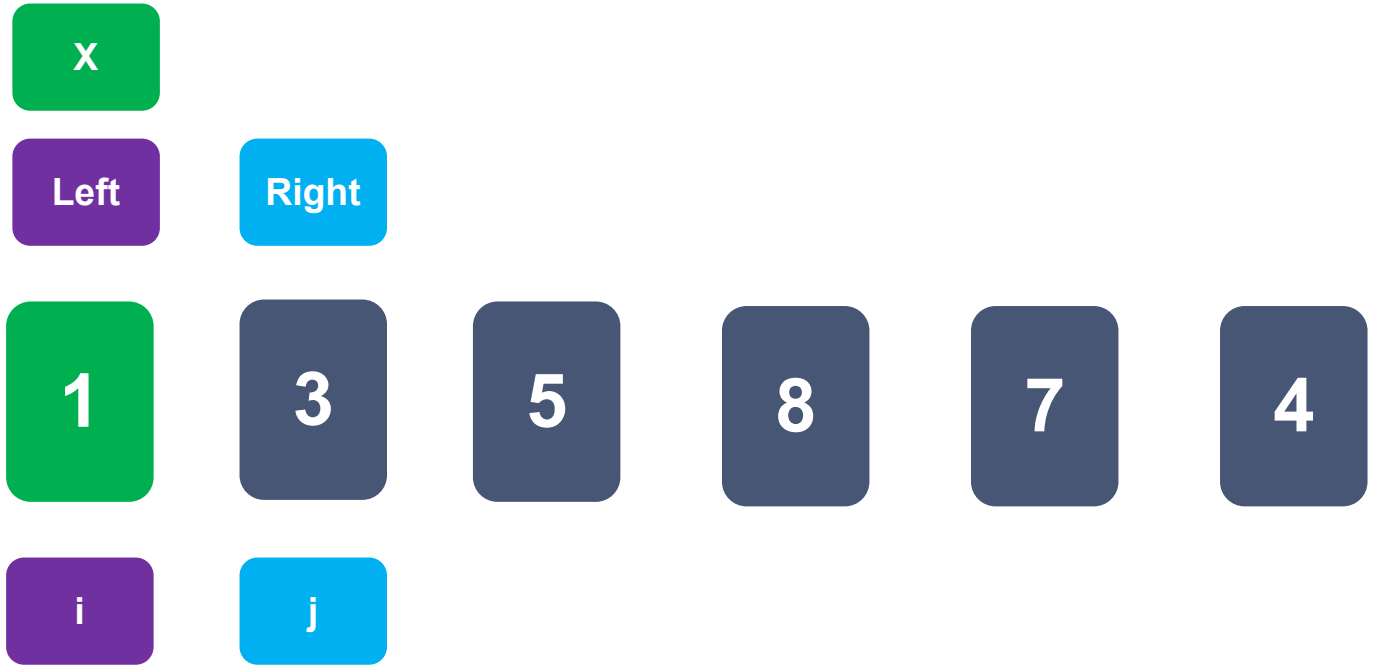
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

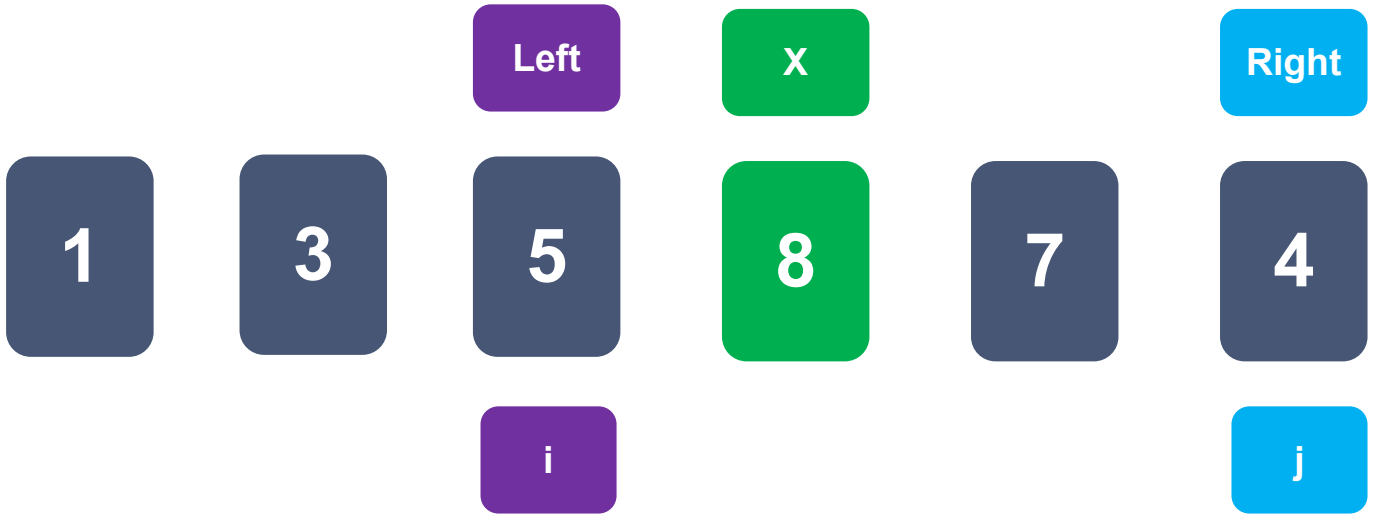
6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



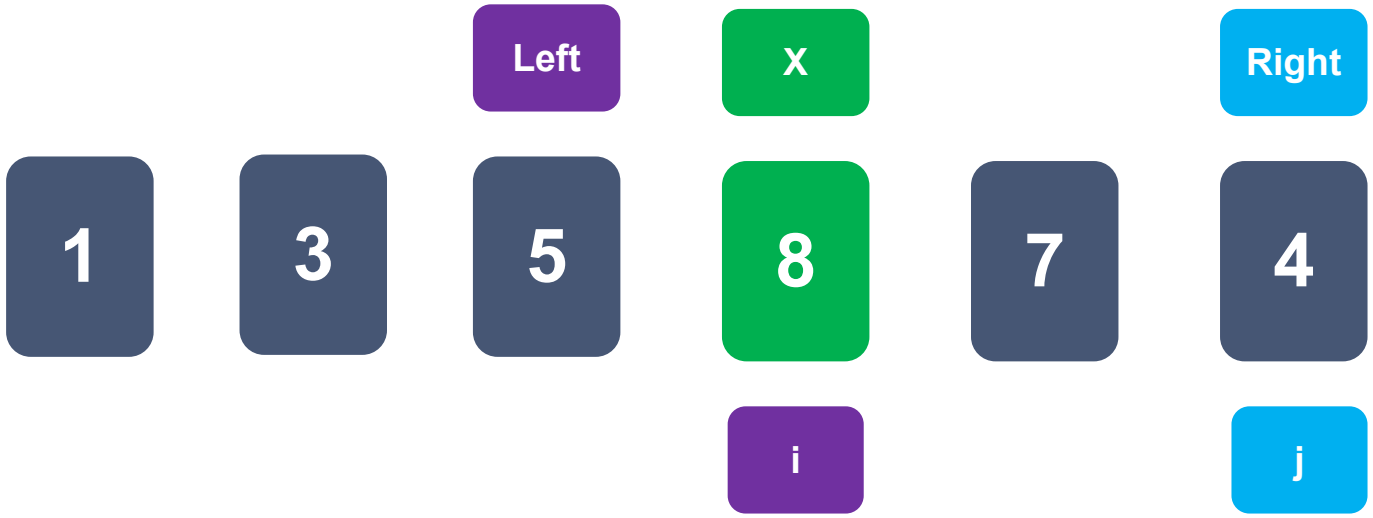
6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

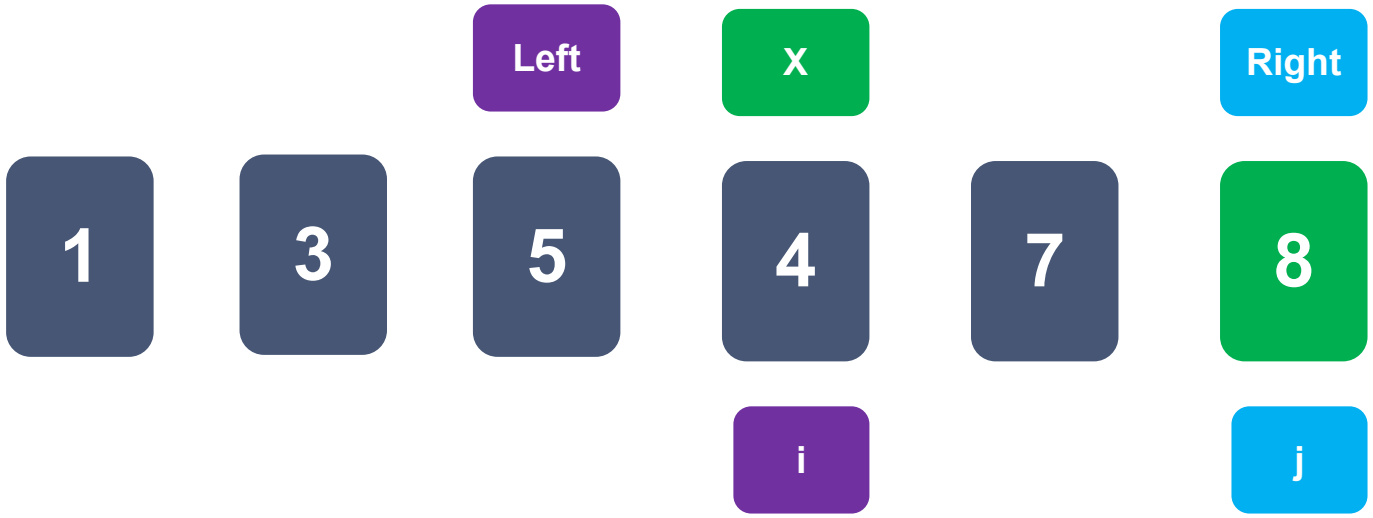


6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

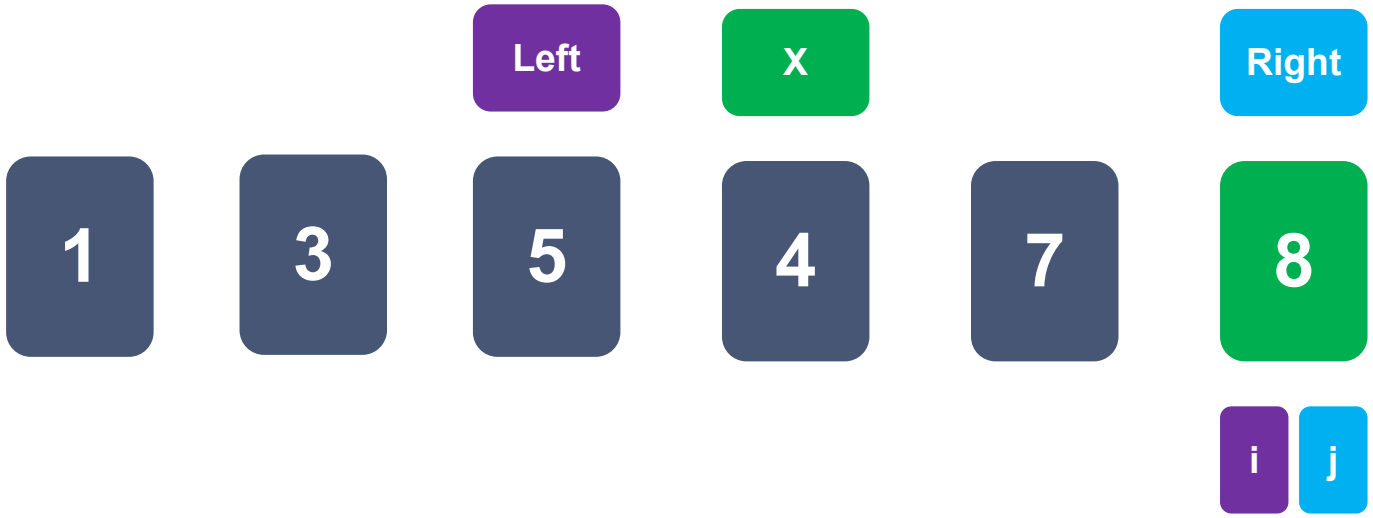
6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

6 SẮP XẾP NHANH – QUICK SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

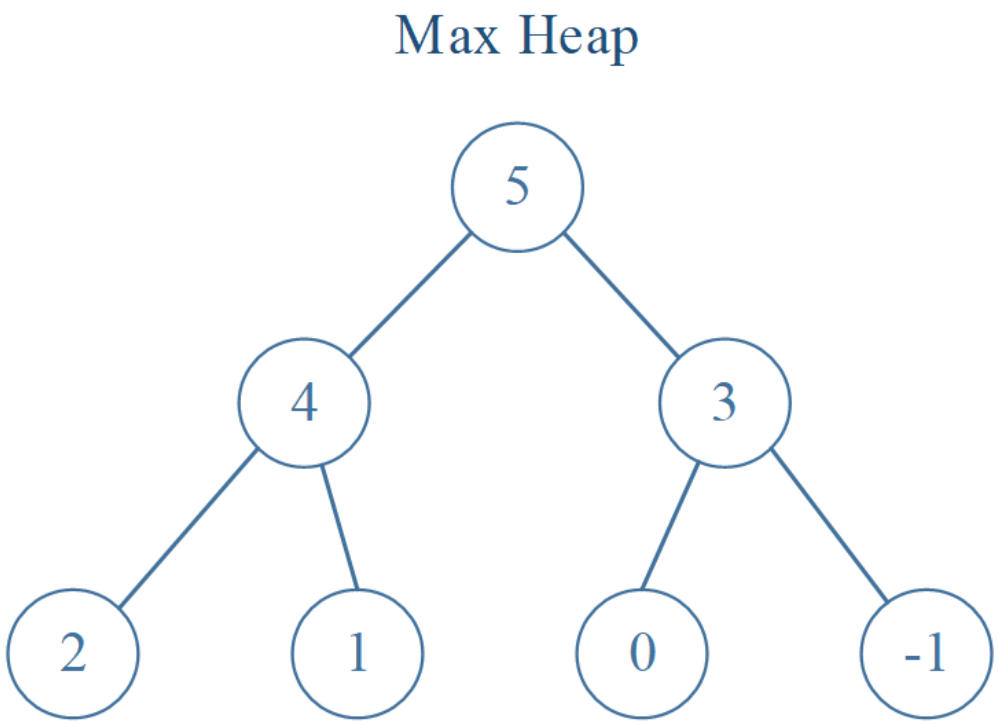
6 SẮP XẾP NHANH – QUICK SORT

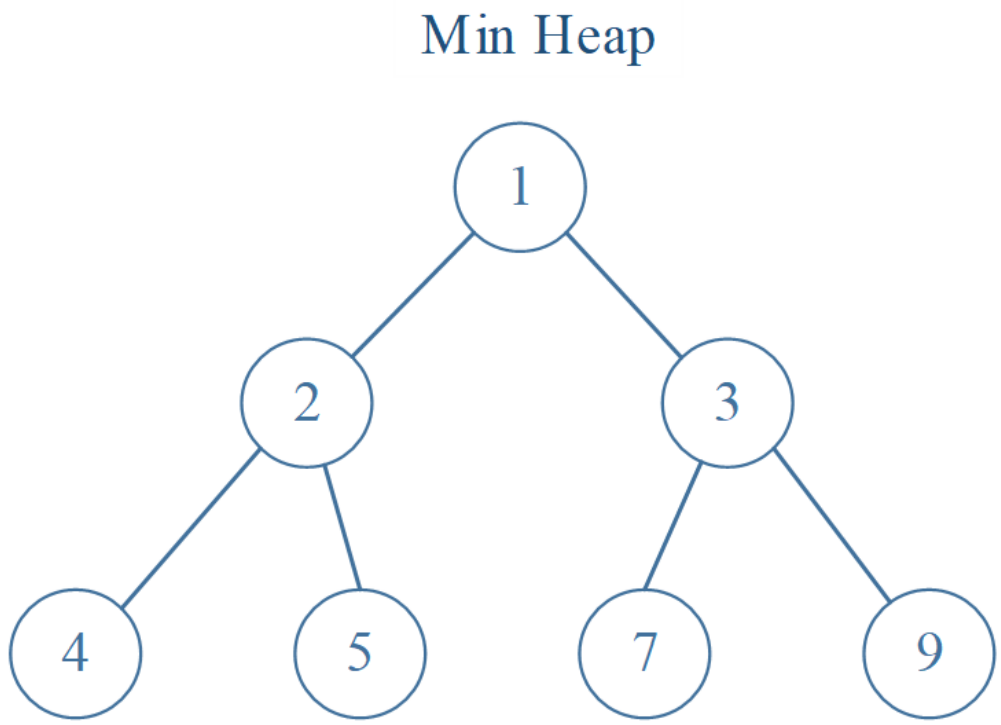
Hàm minh họa giải thuật Quick Sort:

```
void QuickSort (int a[], int left, int right) {
        int i, j, x, t;
        x = a[(left+right)/2]; //Chọn x nằm giữa
        i = left;   j = right;
        do {
                while(a[i] < x) i++;
                while(a[j] > x) j--;
                if (i <= j) {  t = a[i]; a[i] = a[j]; a[j] = t;   i++;   j--; }
        } while (i < j);
        if(left<j)   QuickSort (a, left, j);
        if(i<right)    QuickSort (a, i, right);
}
```

7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Heap là một cây nhị phân hoàn chỉnh có tính chất là khóa ở nút cha bao giờ cũng lớn hơn khóa ở các nút con (max heap) hoặc ngược lại, khóa ở nút cha bé hơn khóa ở các nút con (min heap).





7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ý tưởng của giải thuật Heap Sort: Sắp xếp thông qua việc tạo các heap (đống) từ dãy ban đầu.

Các bước thực hiện: Việc thực hiện giải thuật được chia thành 2 giai đoạn:

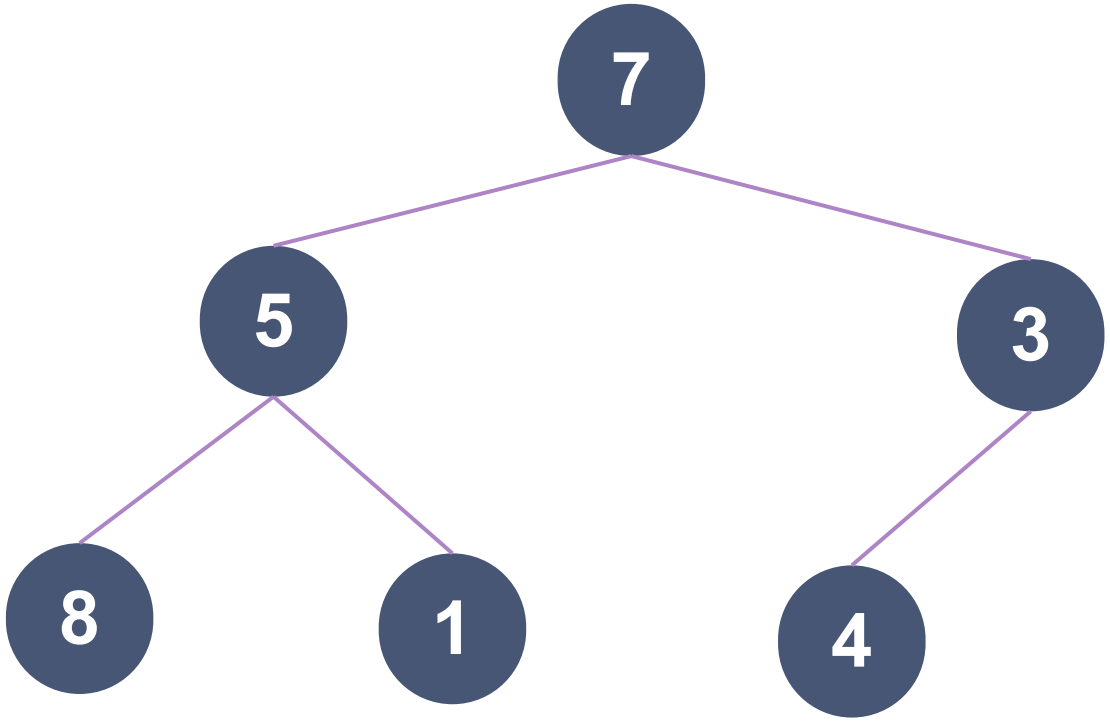
- **Giai đoạn 1:** Tạo heap từ dãy ban đầu, khi đó nút gốc là phần tử lớn nhất.
- **Giai đoạn 2:** Sắp xếp dãy dựa trên heap được tạo (giả sử là max heap):
  - Nút gốc là nút có giá trị lớn nhất, ta chuyển về vị trí cuối cùng của heap.
  - Loại phần tử lớn nhất khỏi heap.
  - Hiệu chỉnh các phần còn lại thành heap.
  - Lặp lại quá trình cho tới khi heap chỉ còn 1 nút.

7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

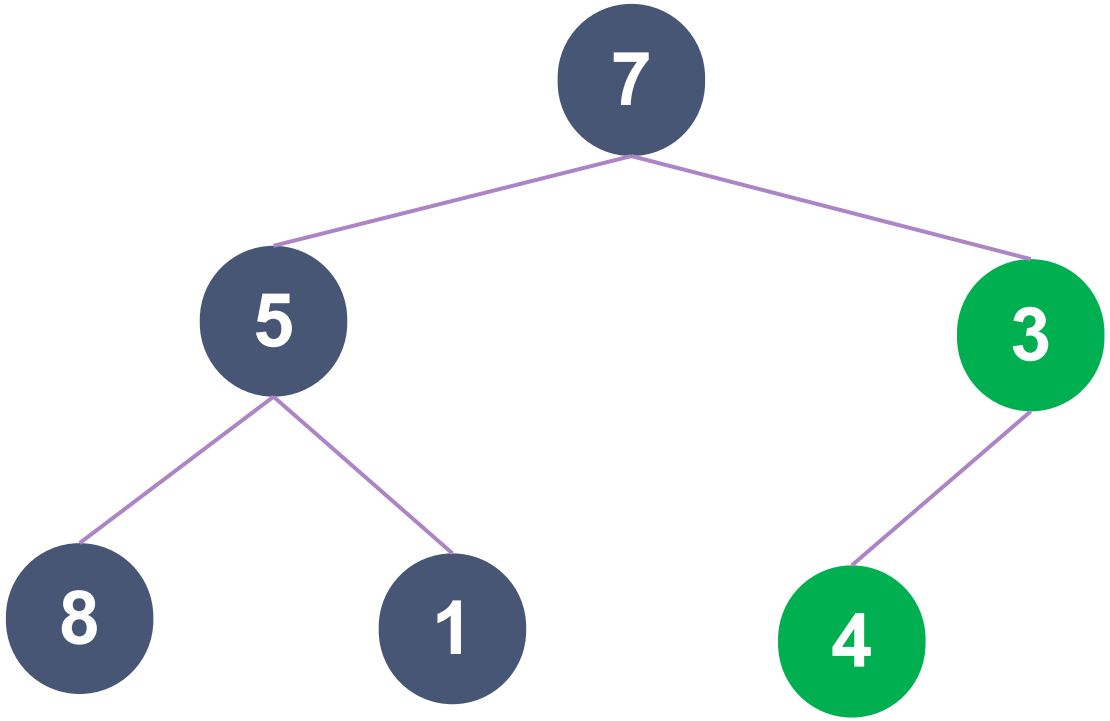
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



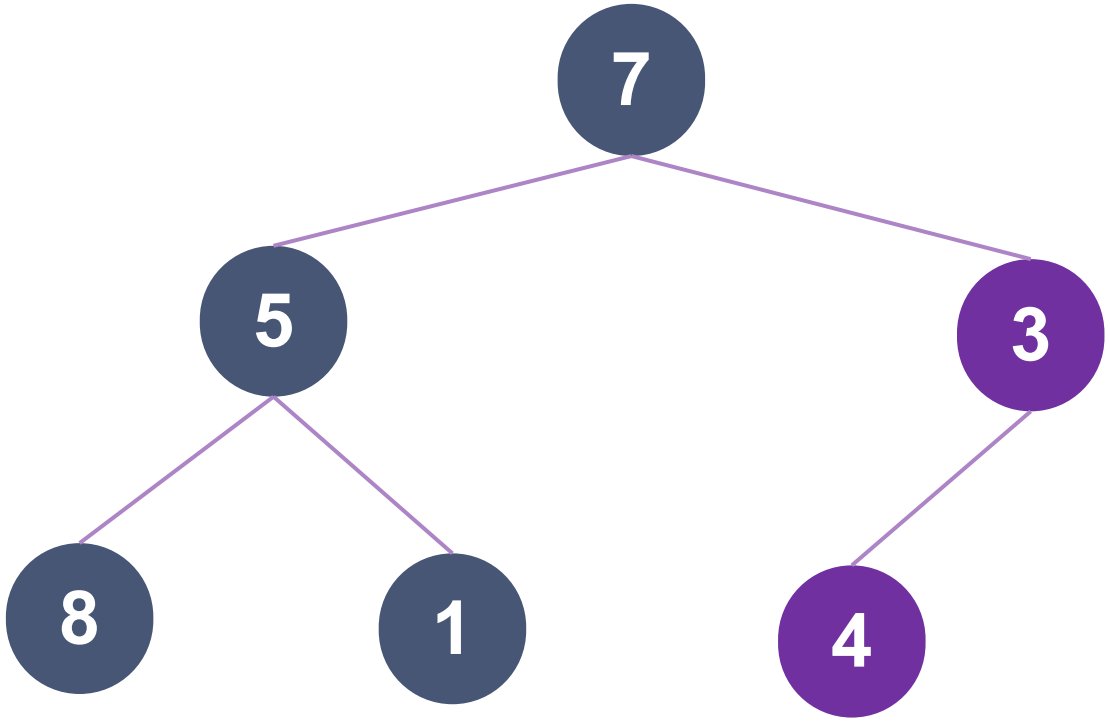
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



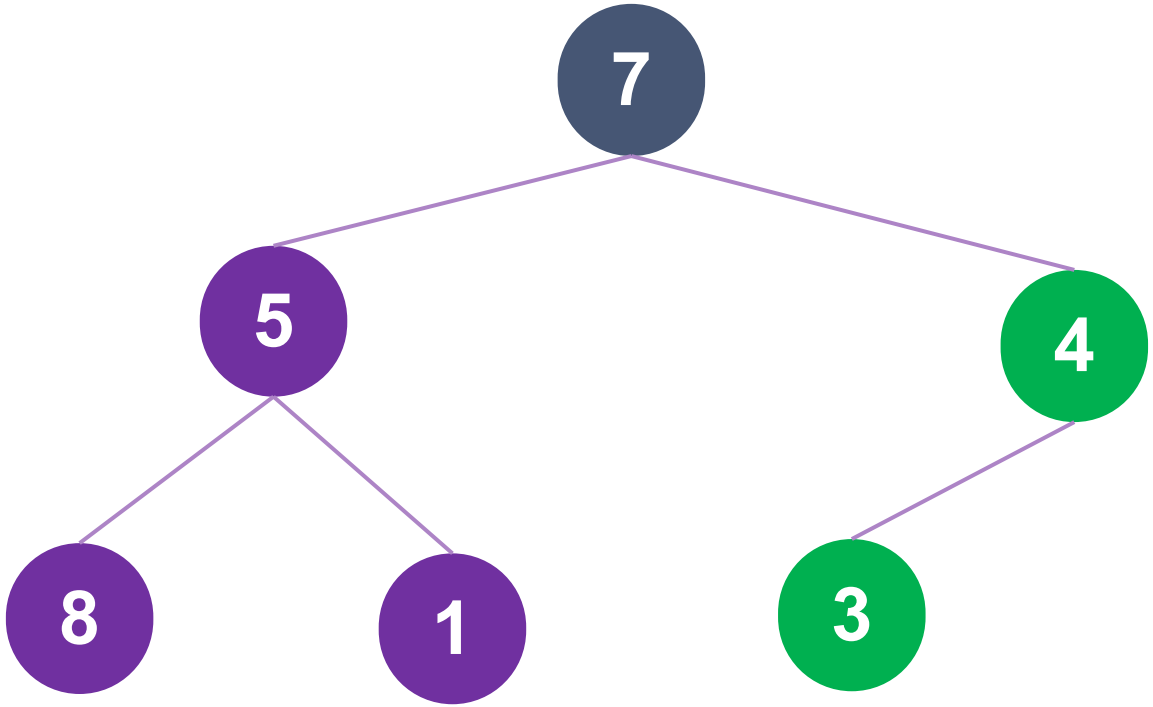
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



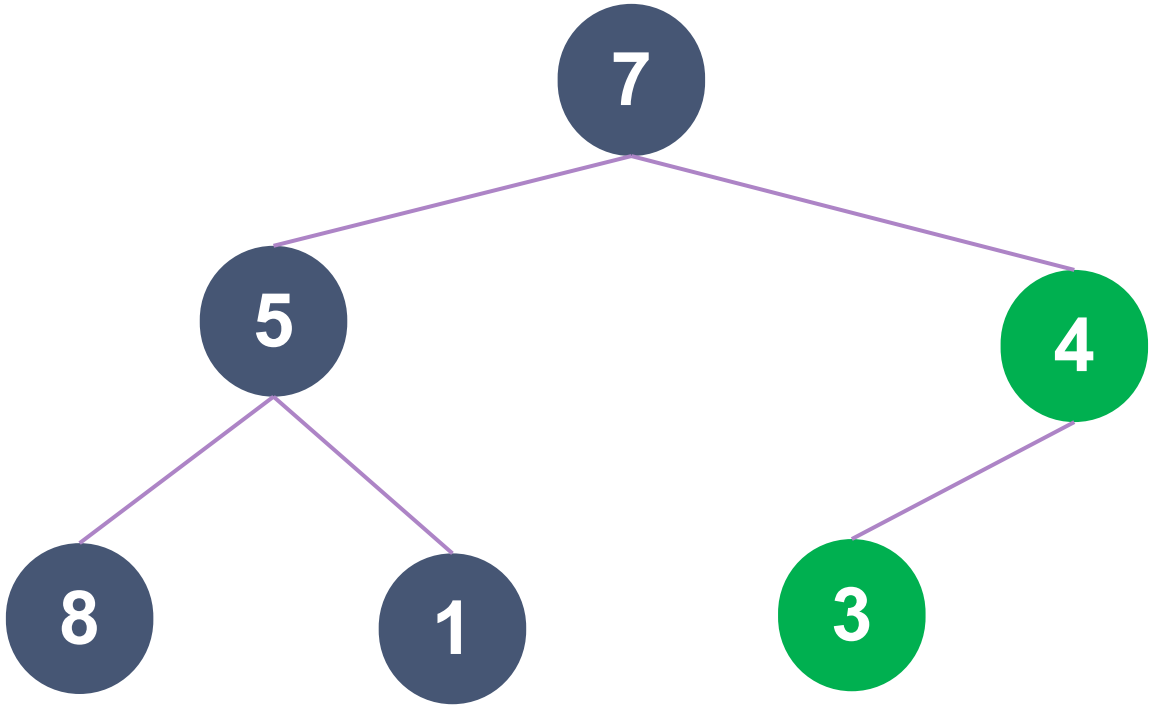
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



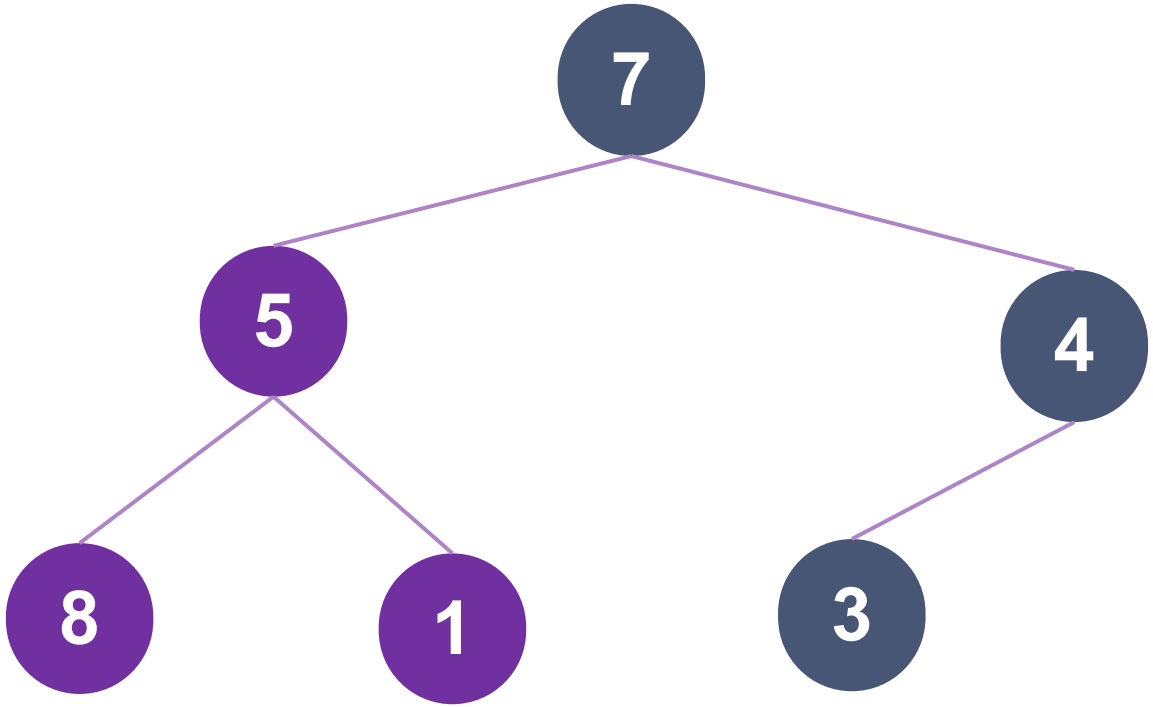
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



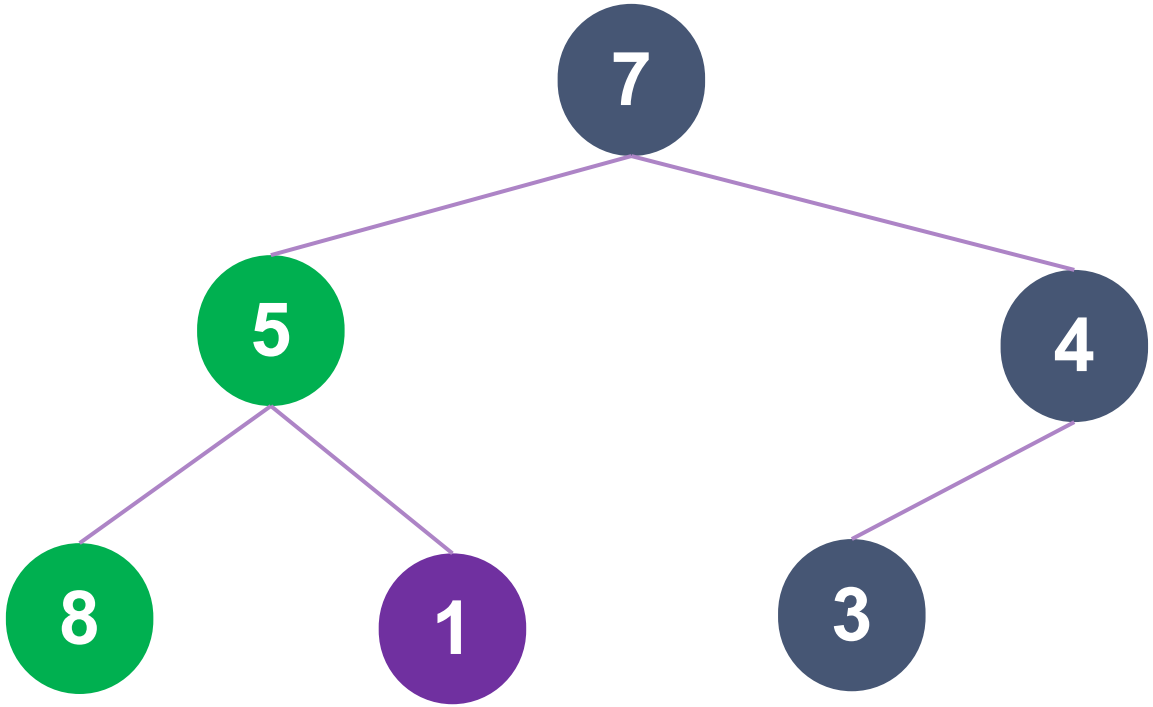
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



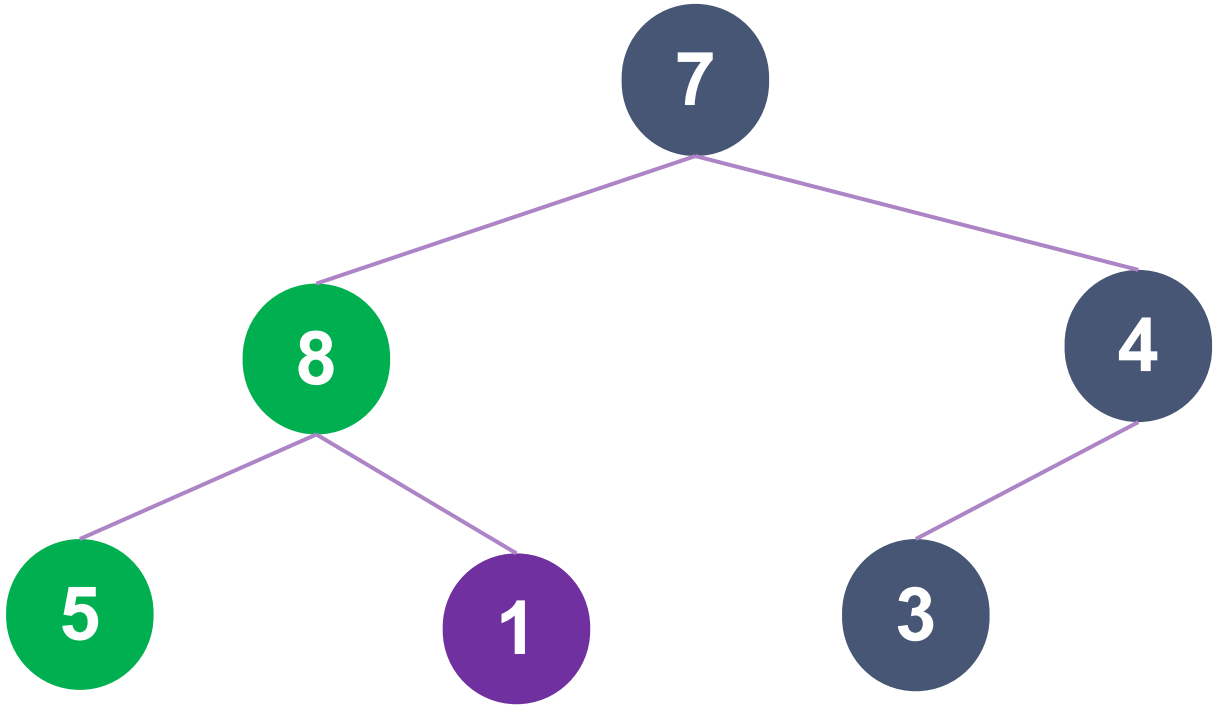
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



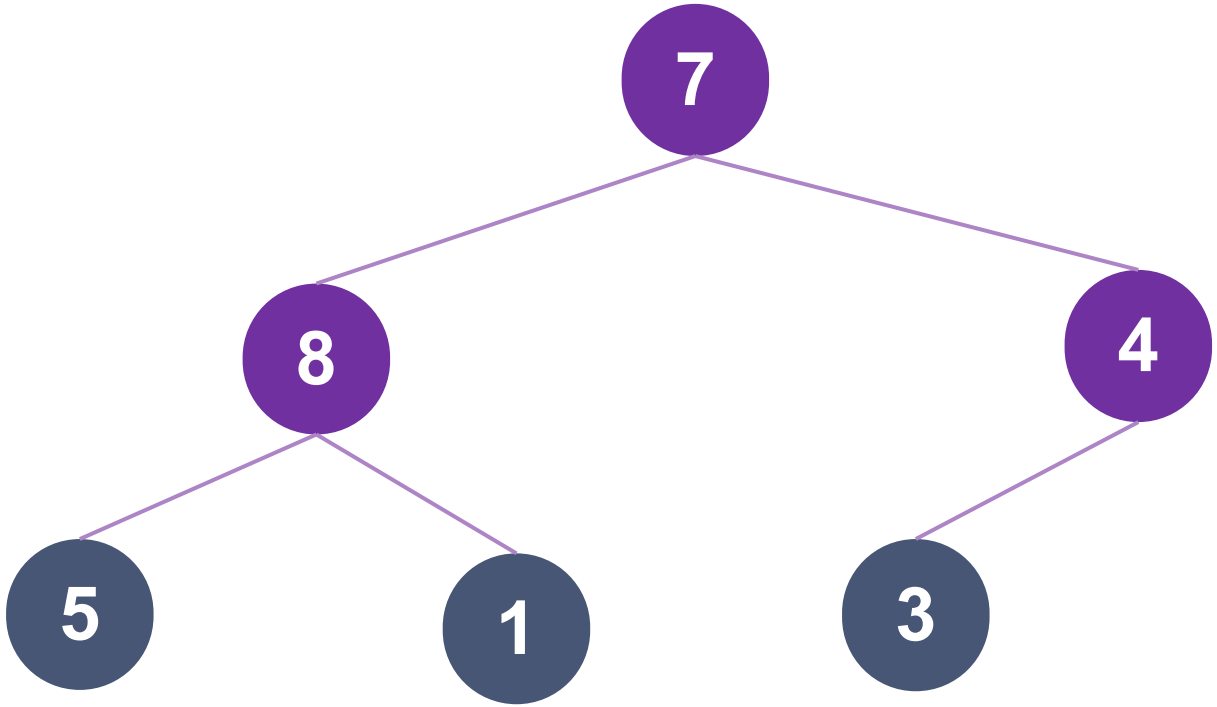
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



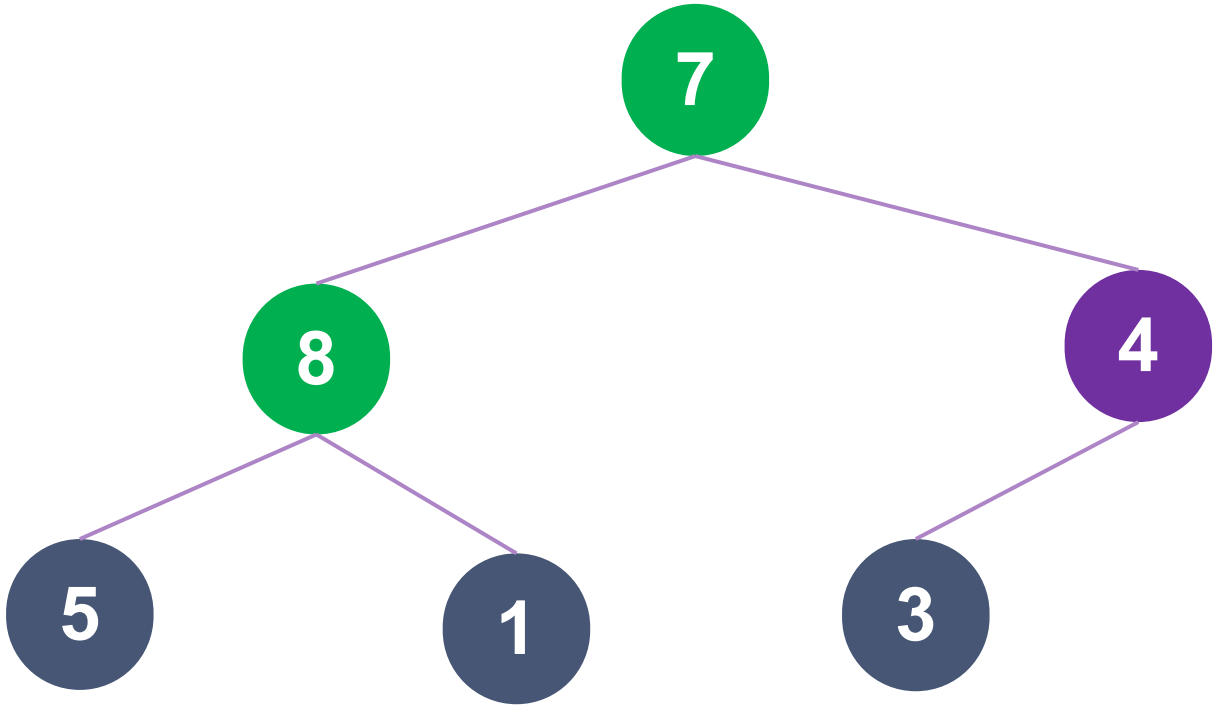
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



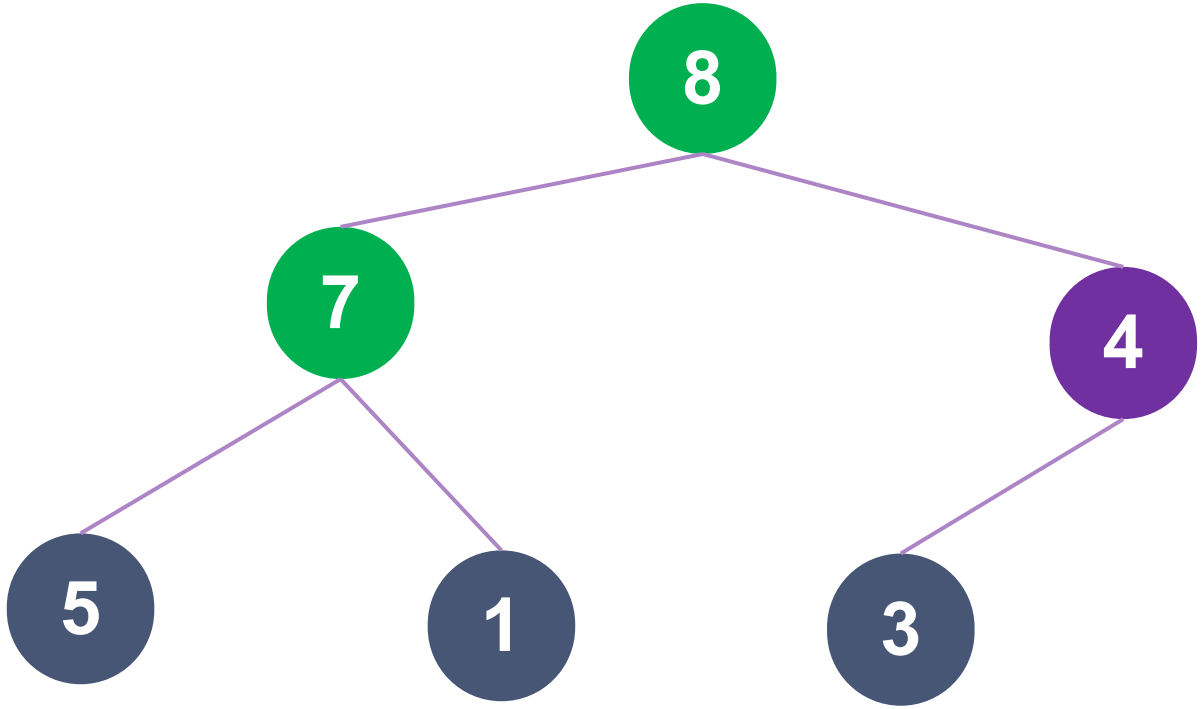
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



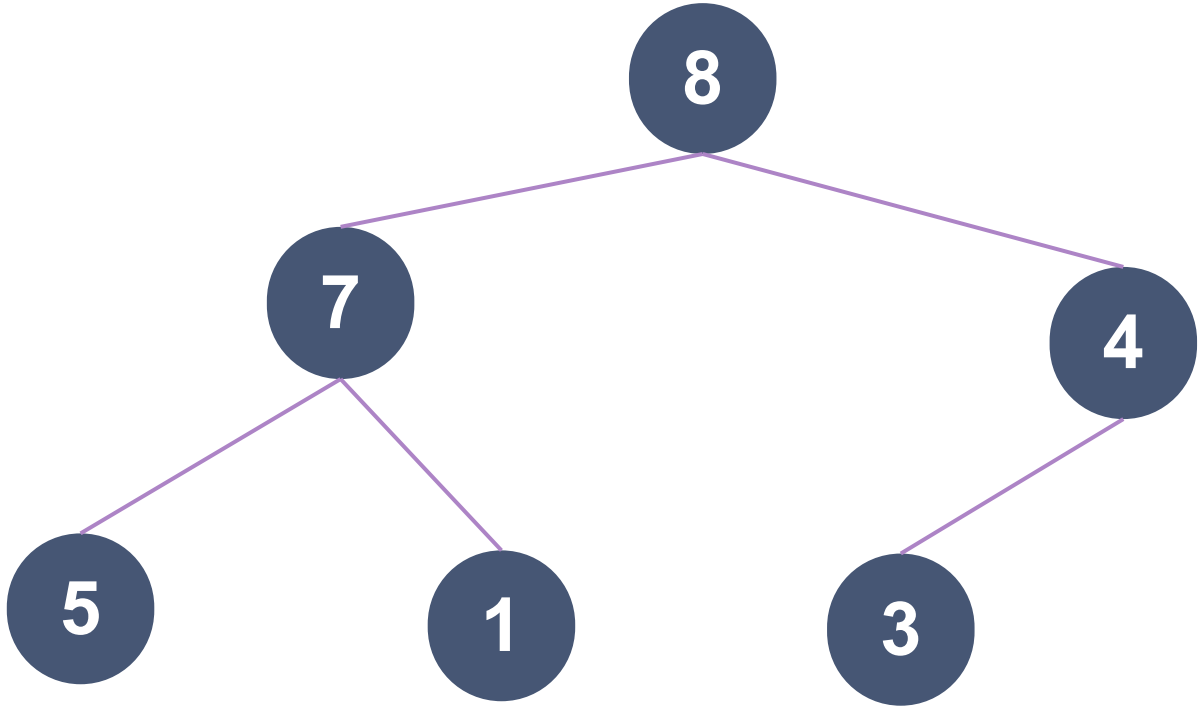
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



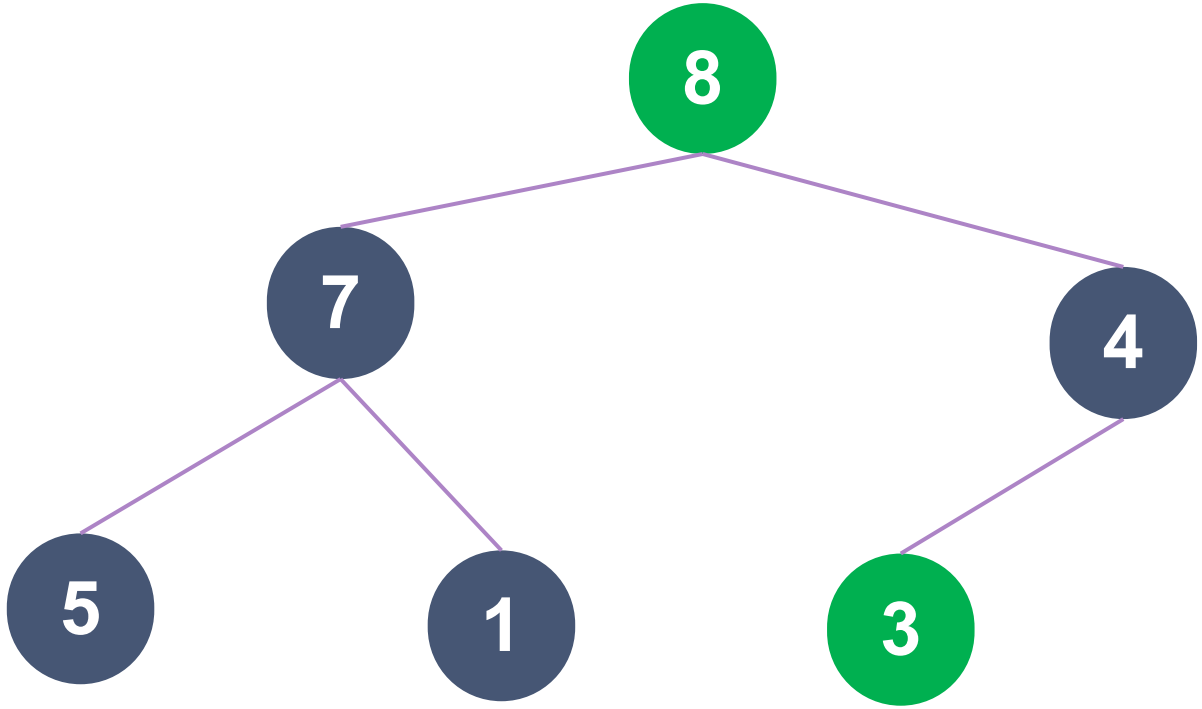
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



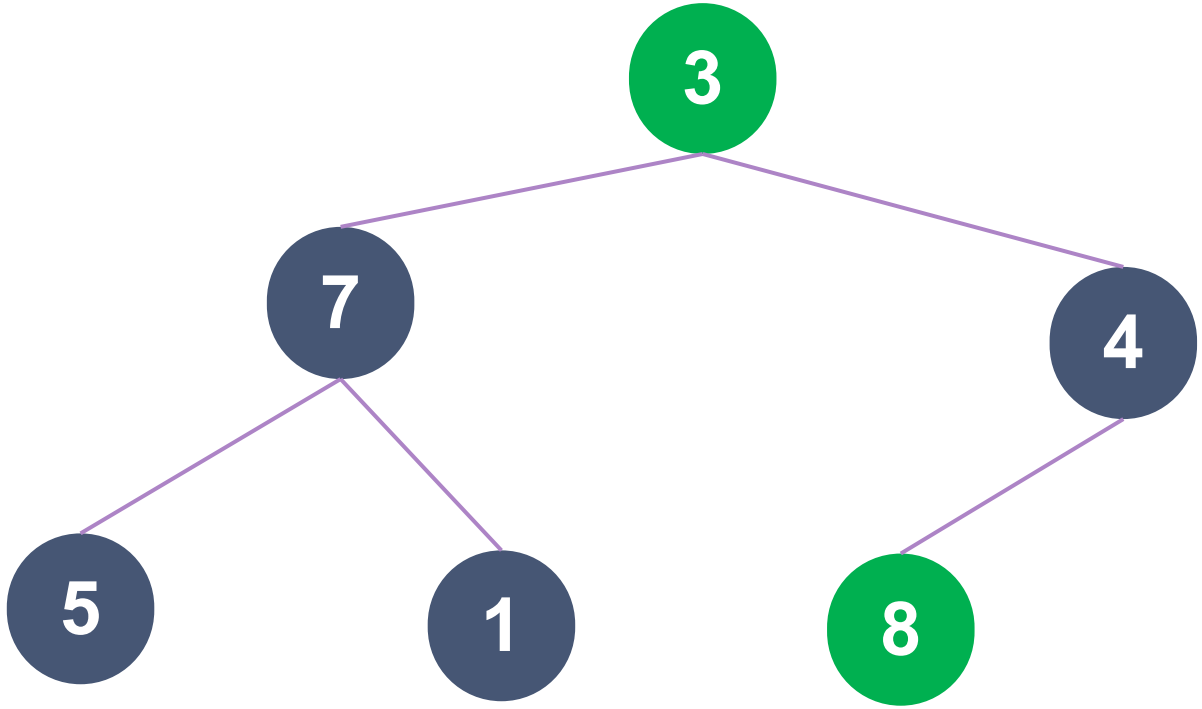
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



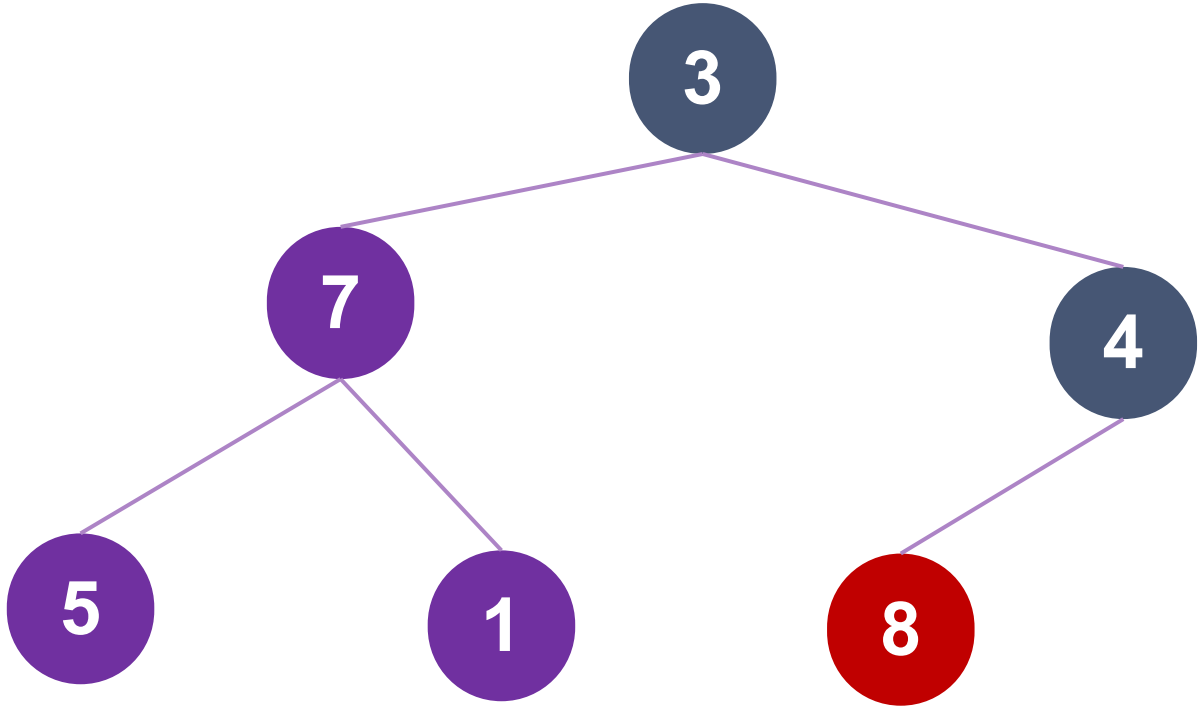
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



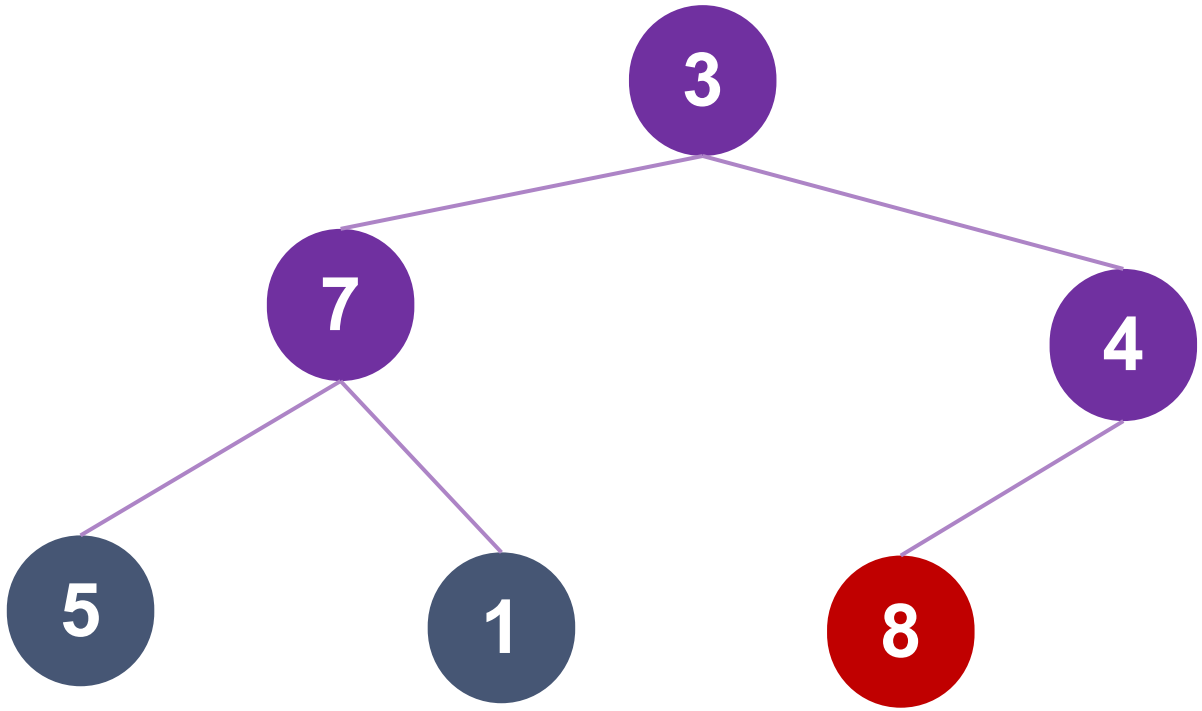
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



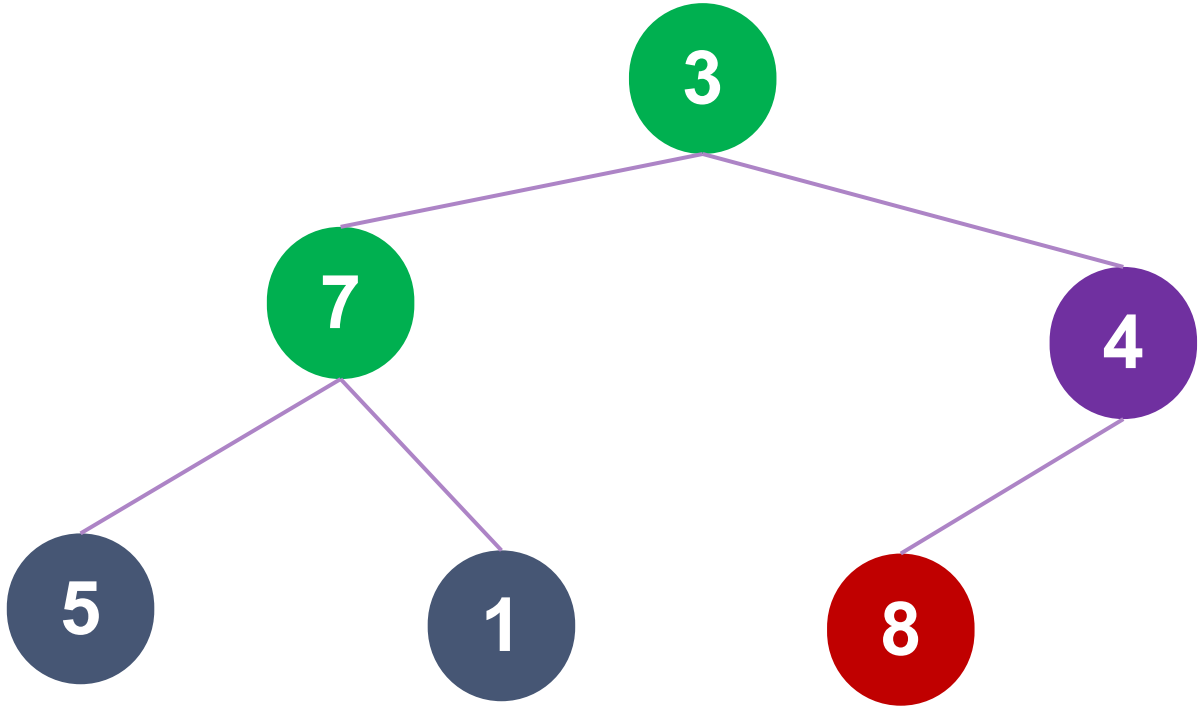
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



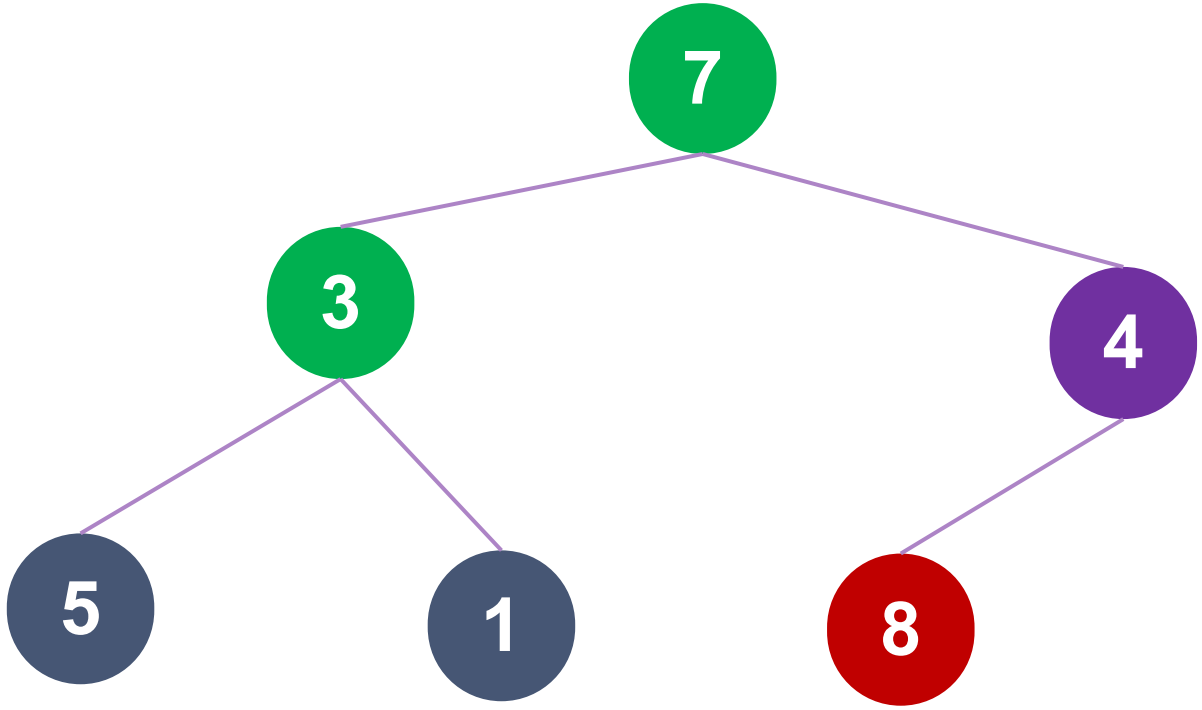
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



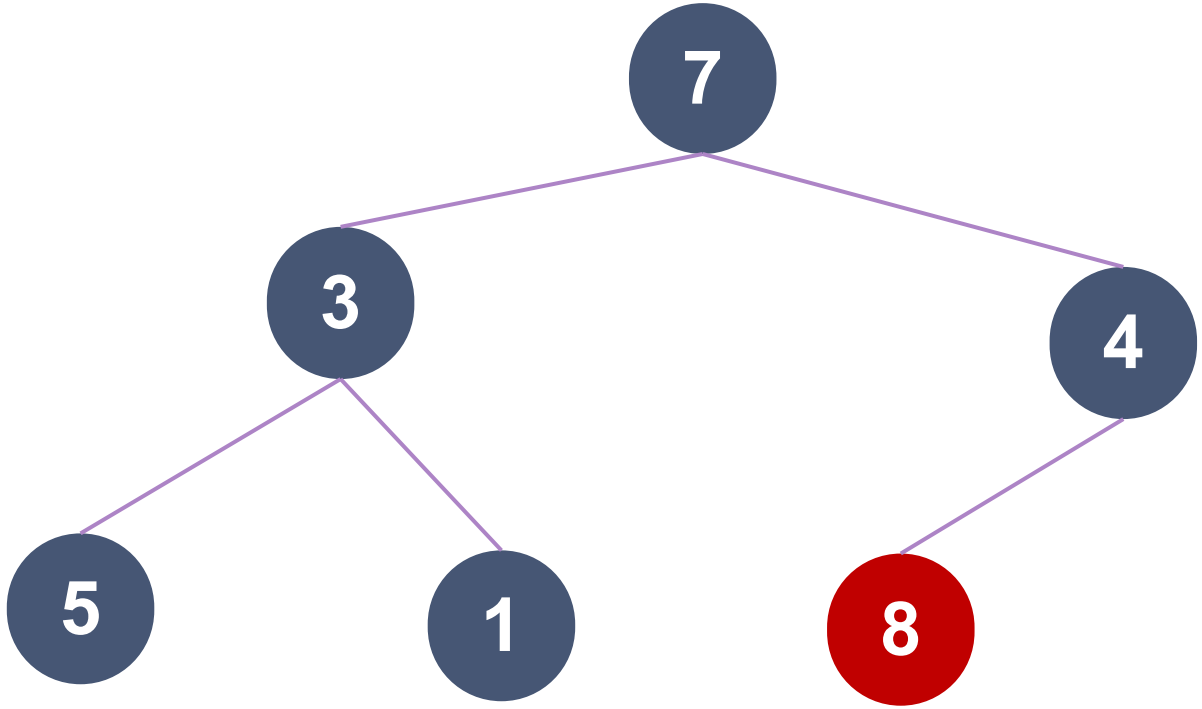
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



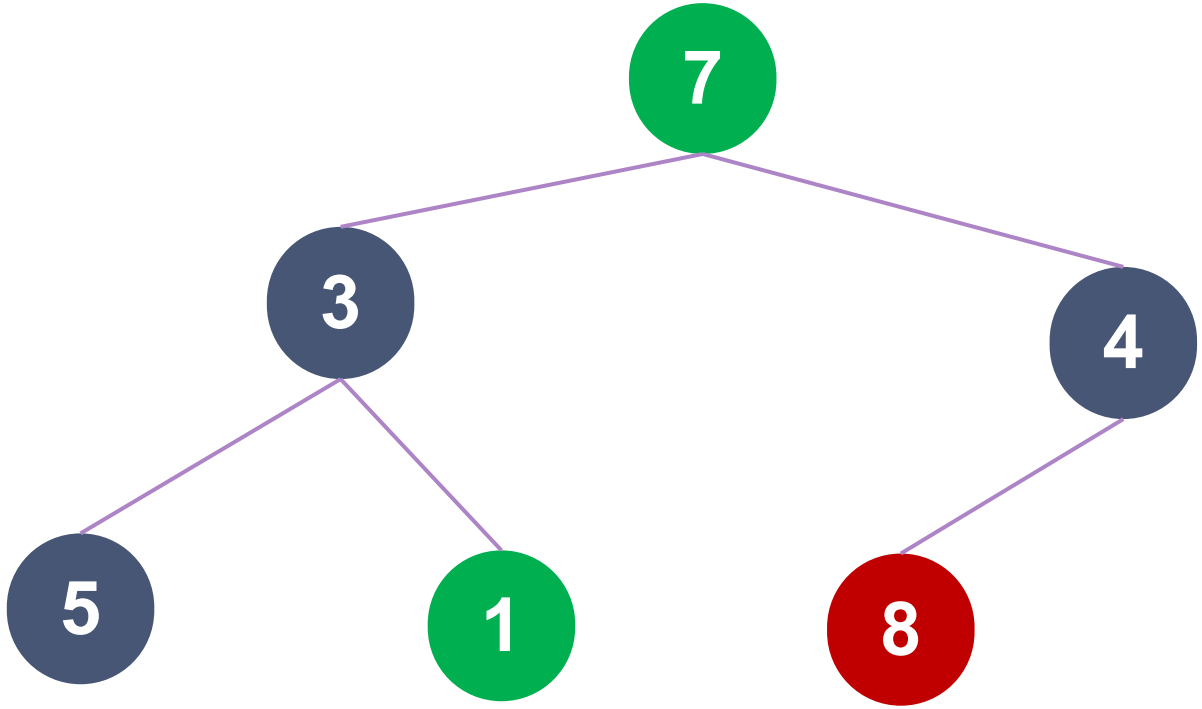
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



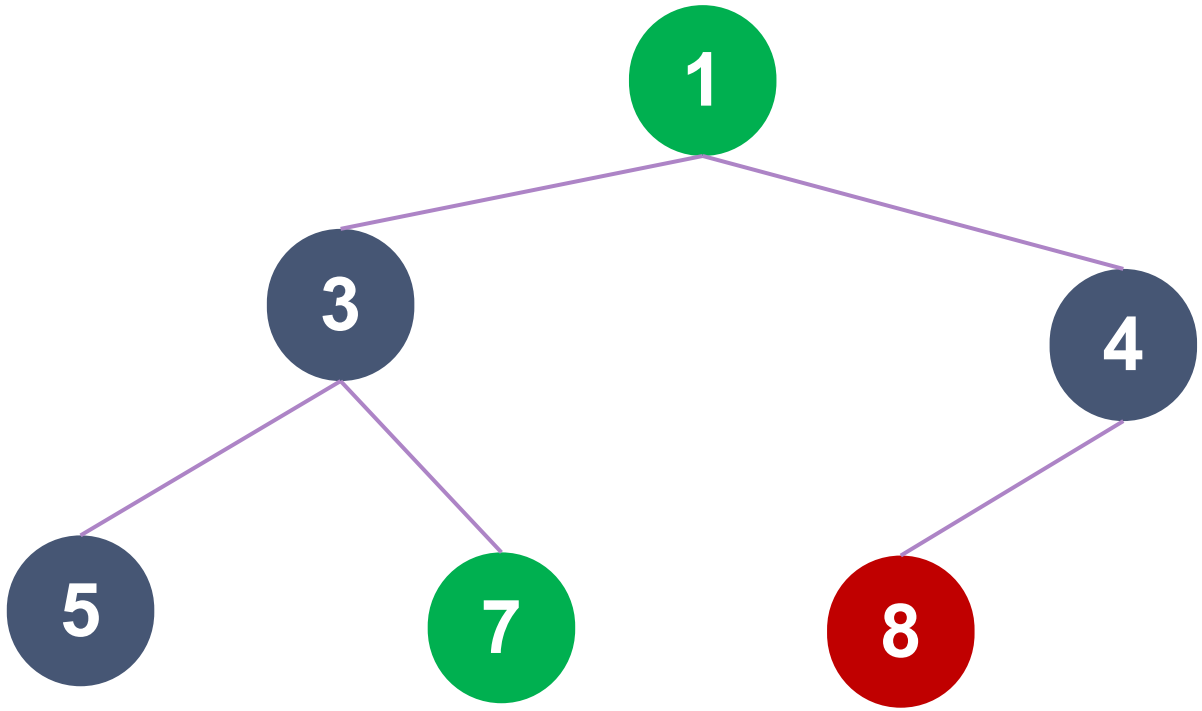
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



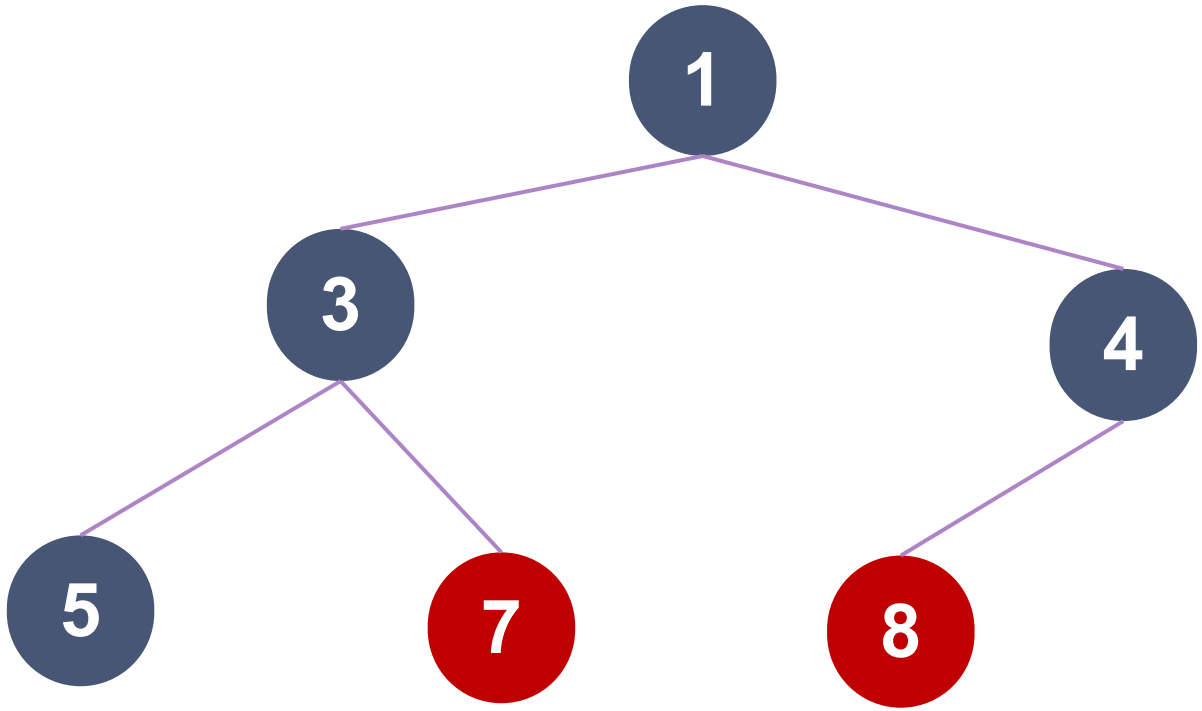
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



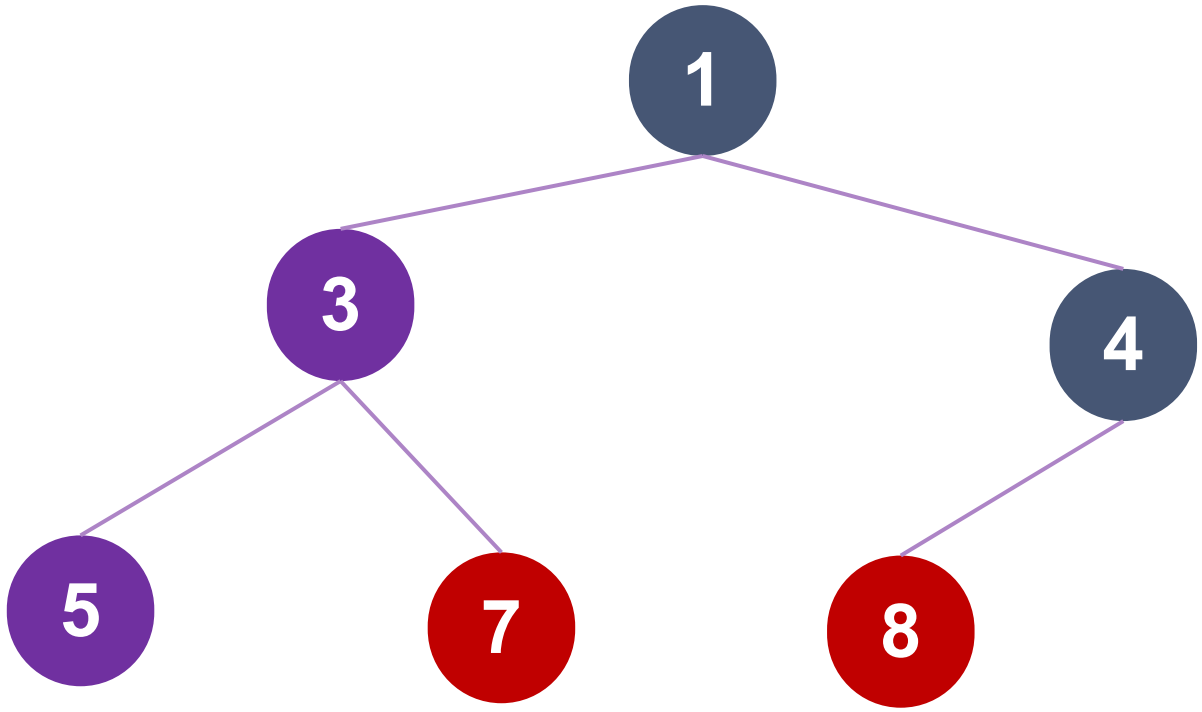
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



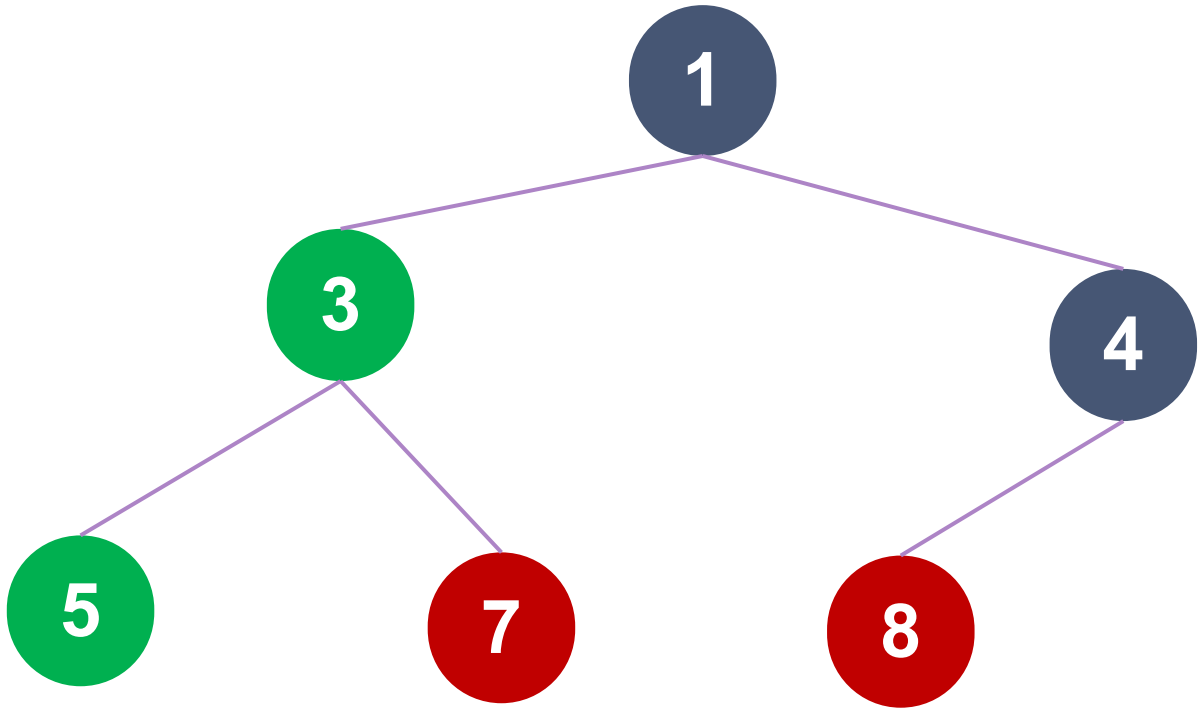
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



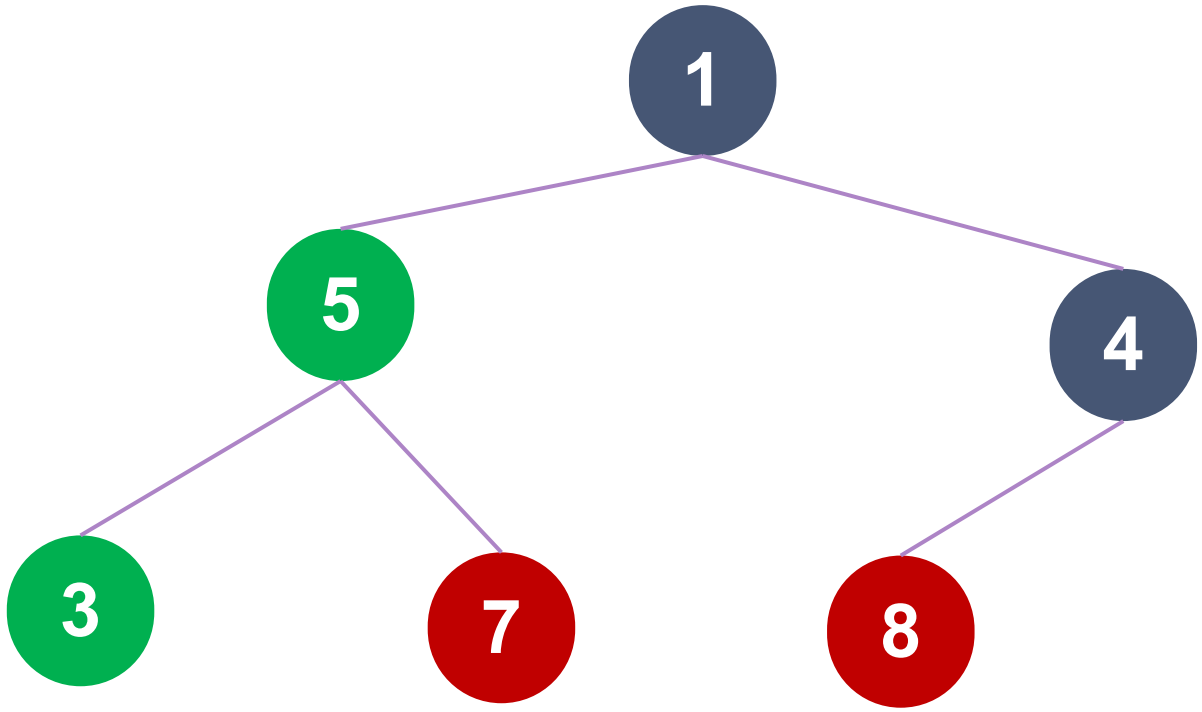
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



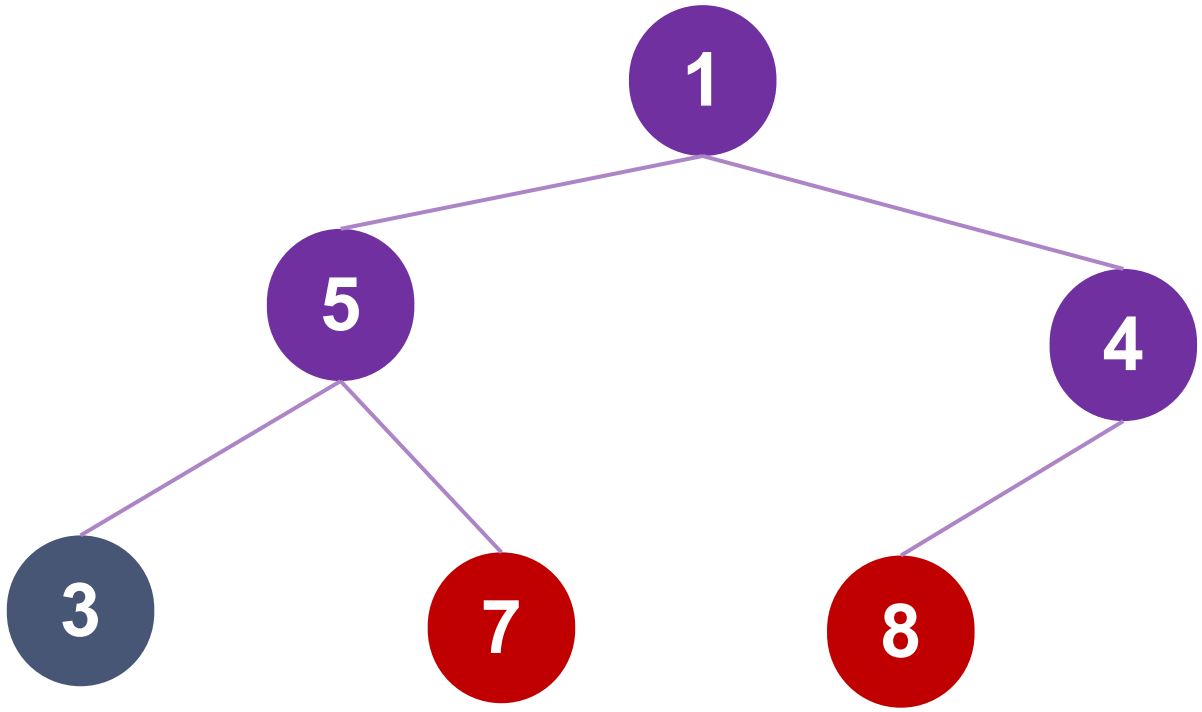
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



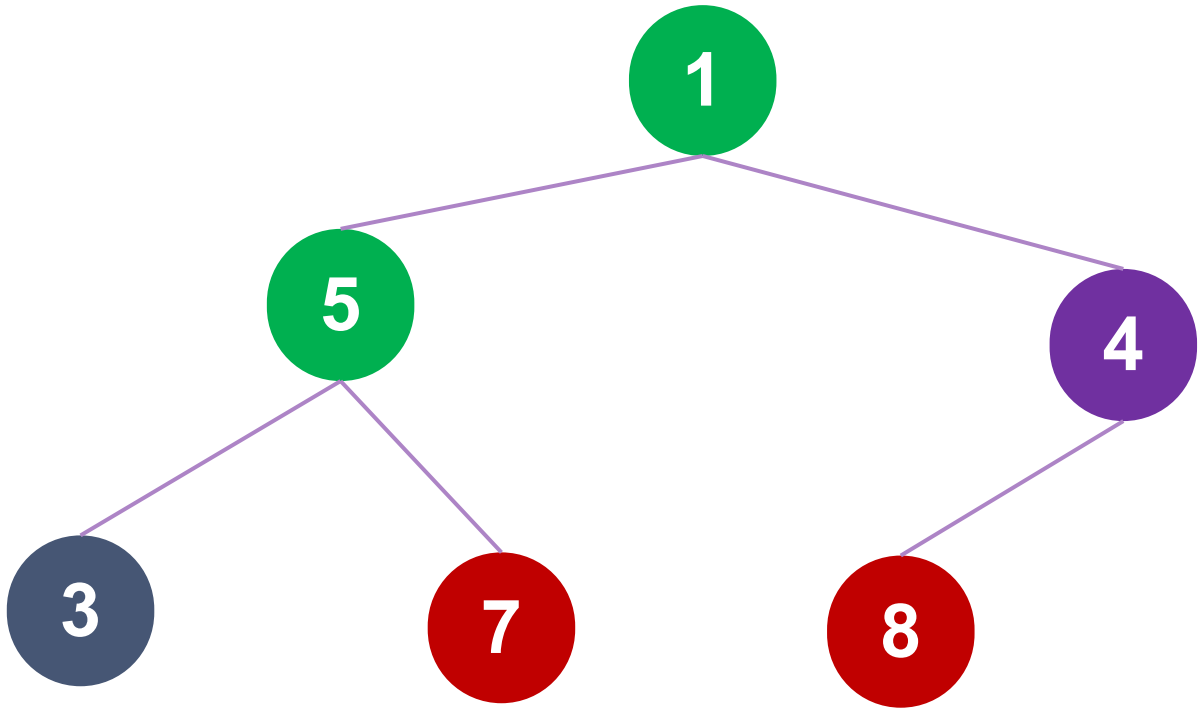
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



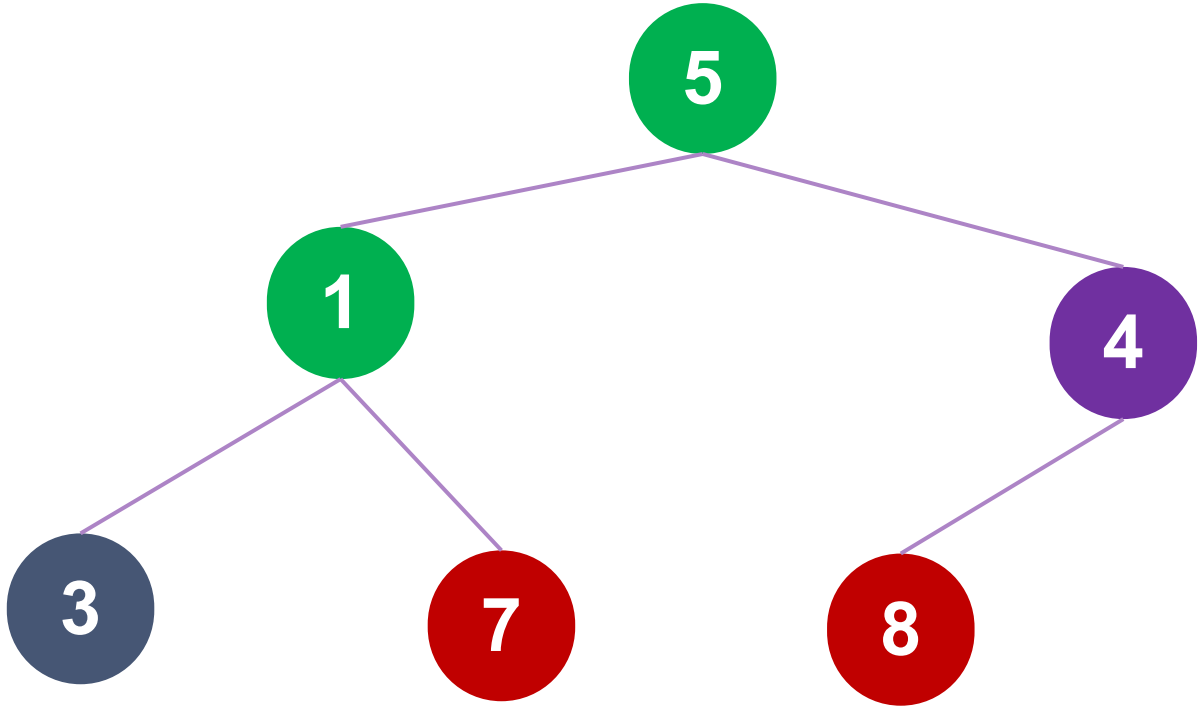
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



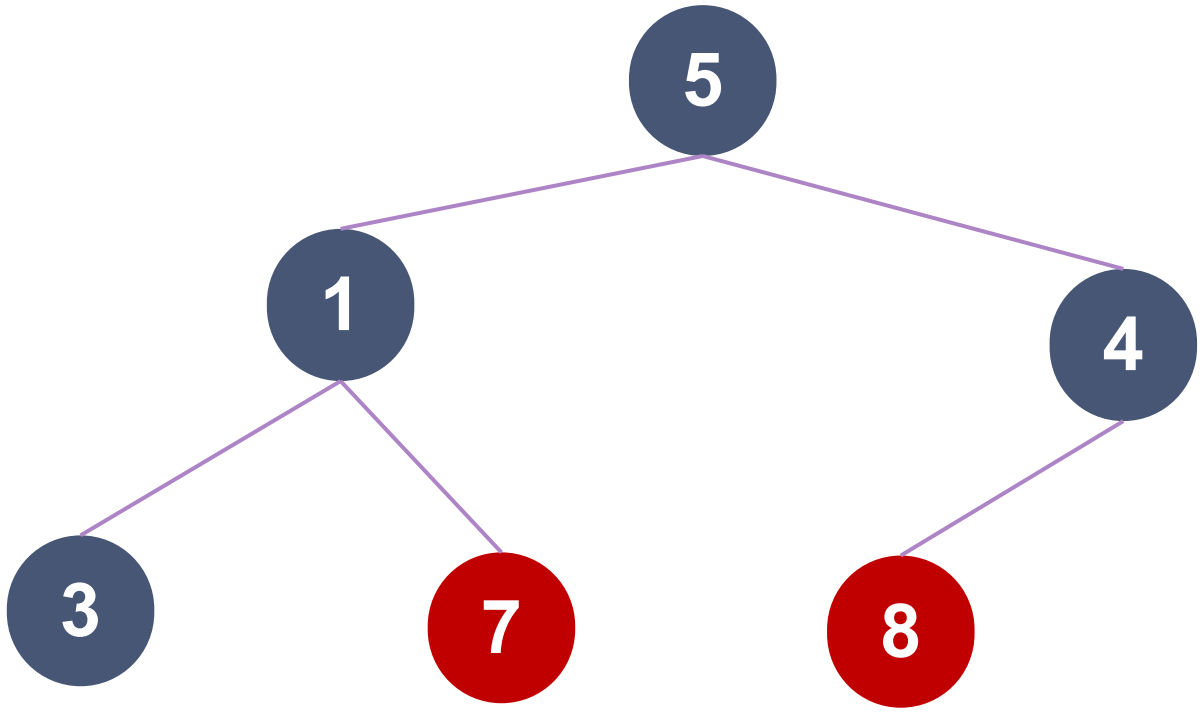
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



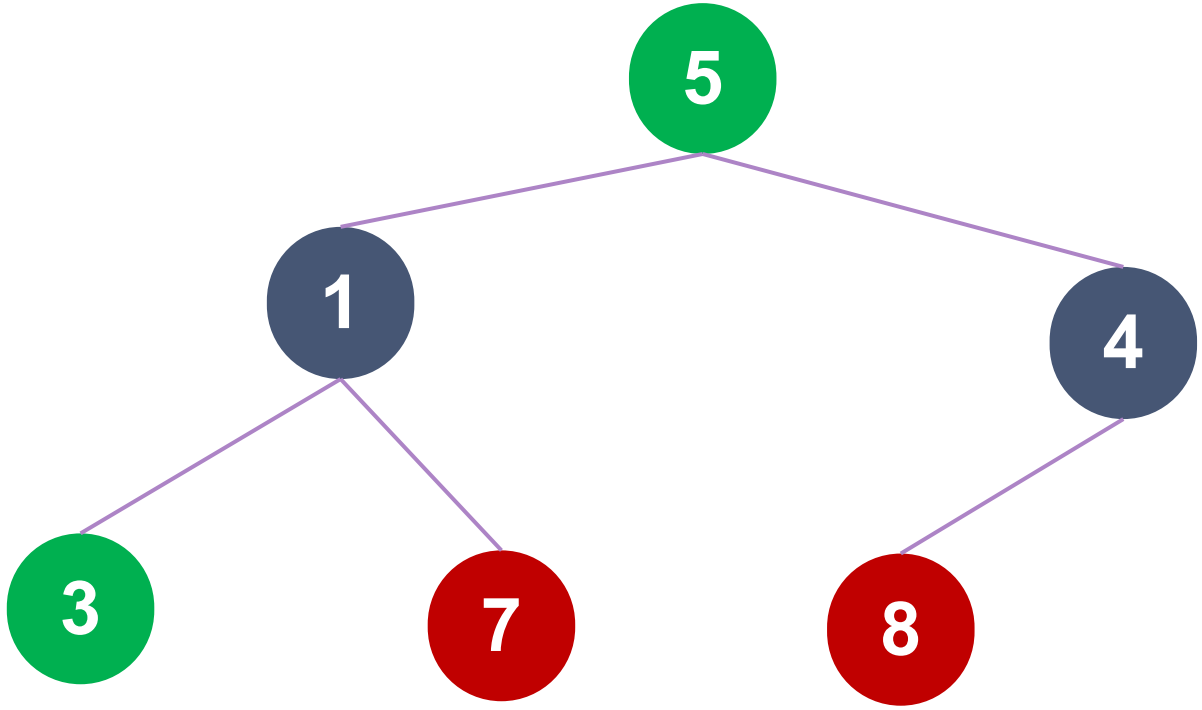
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



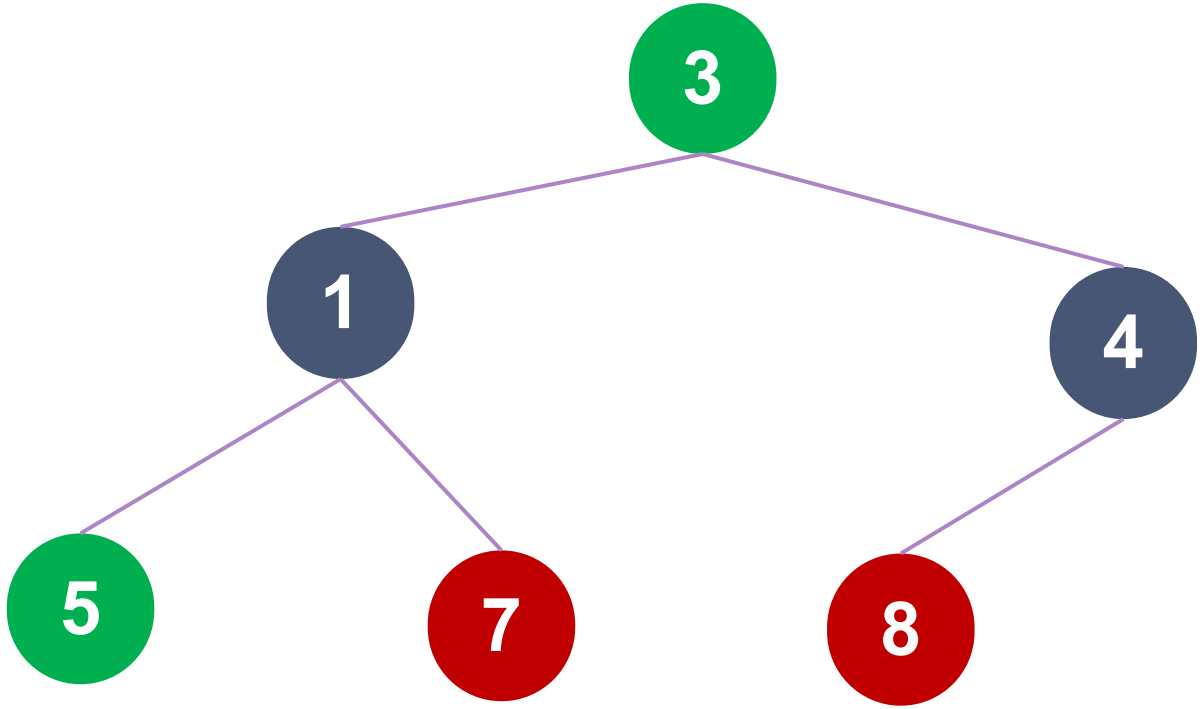
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



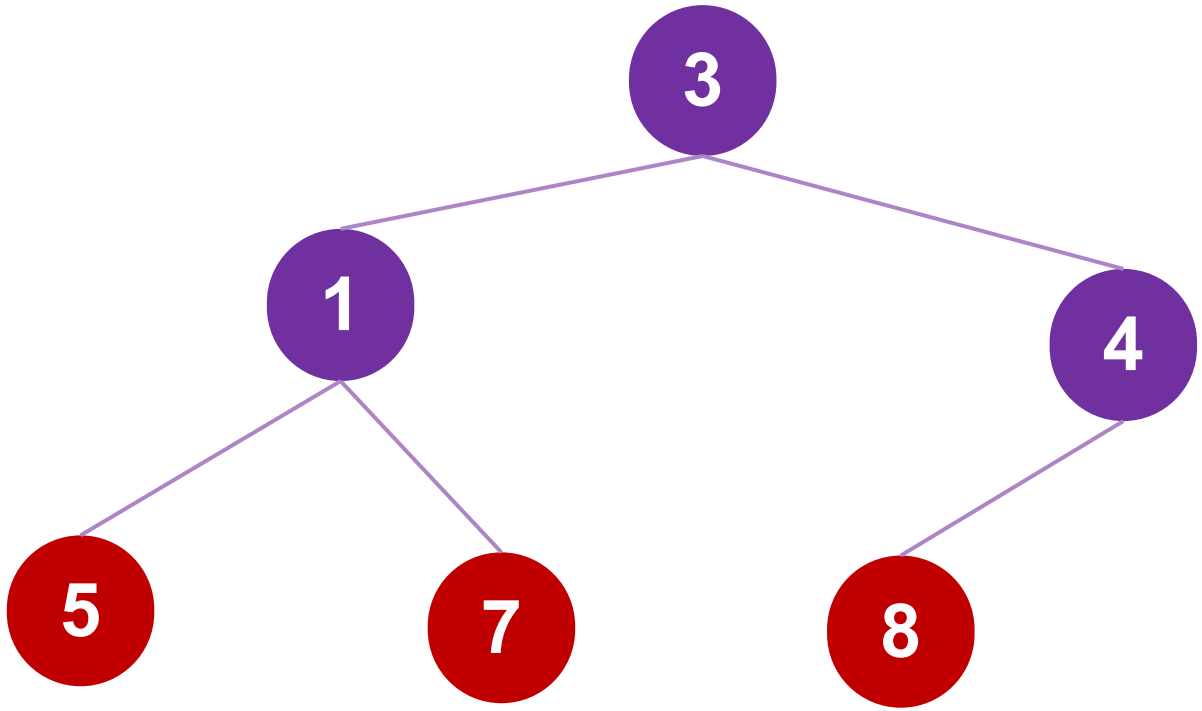
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



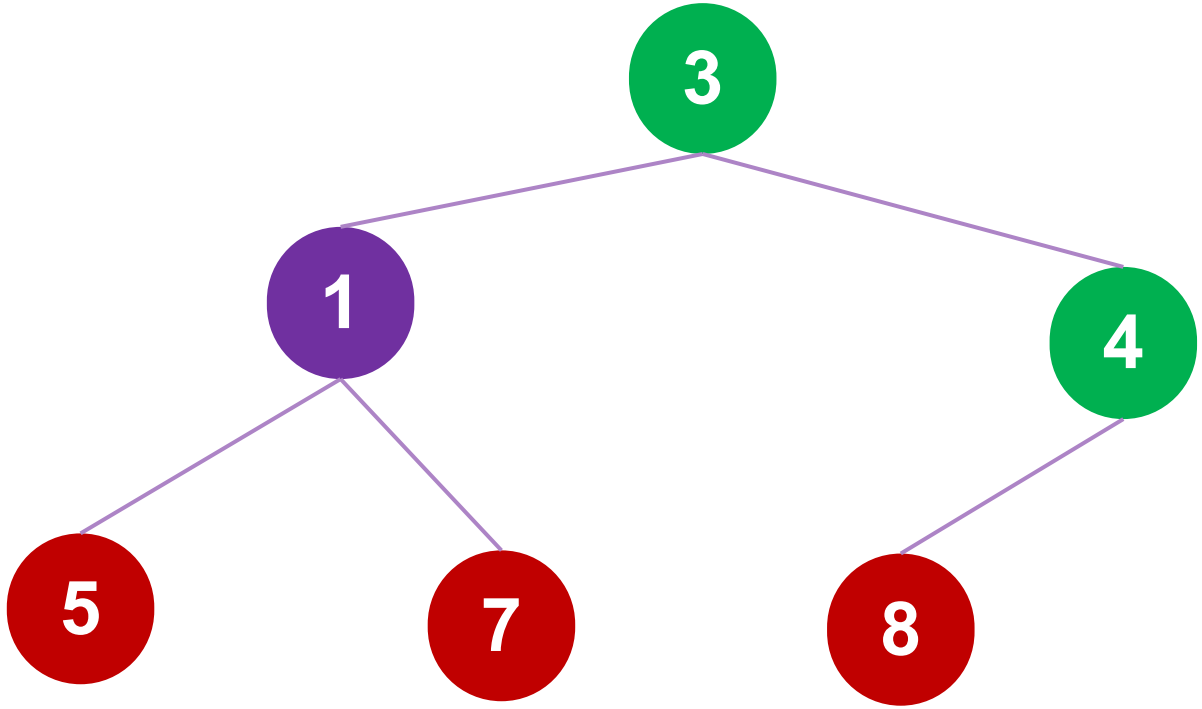
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



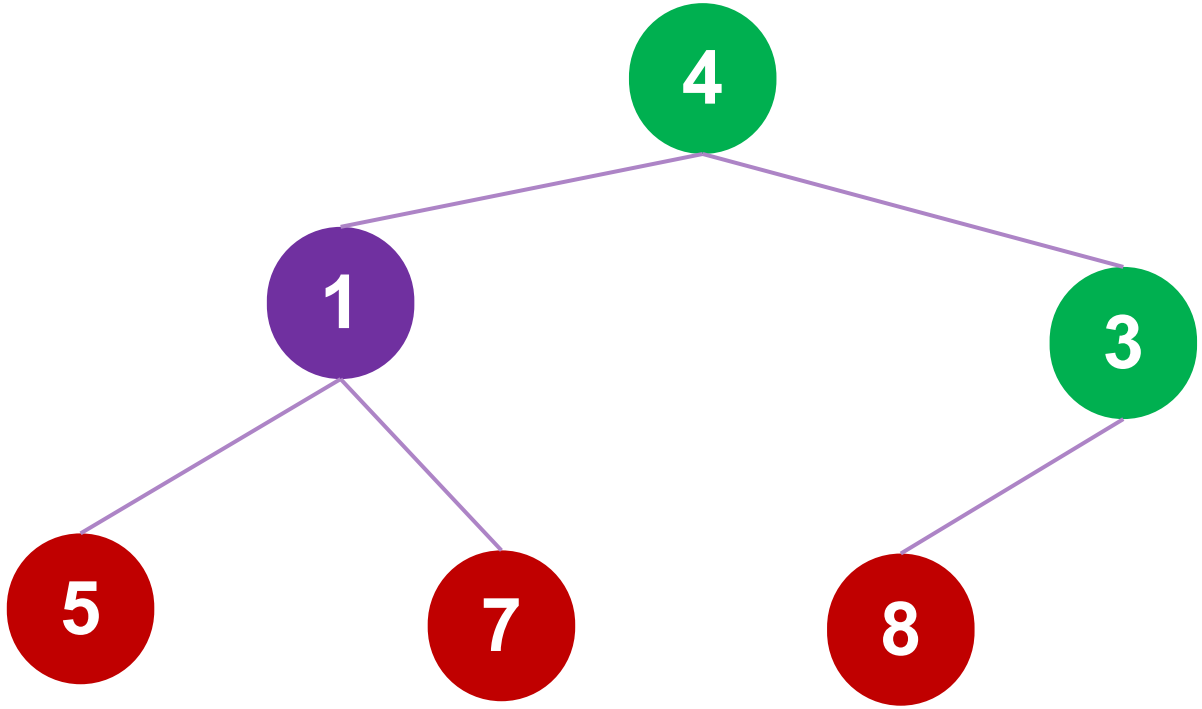
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



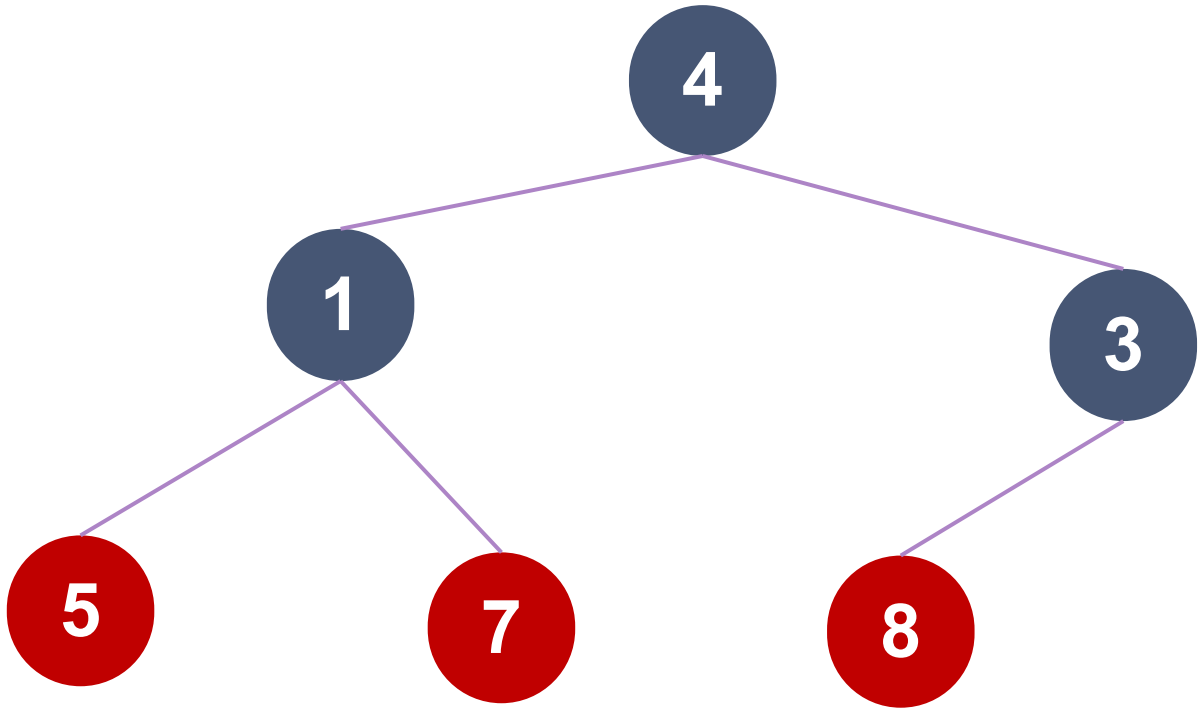
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



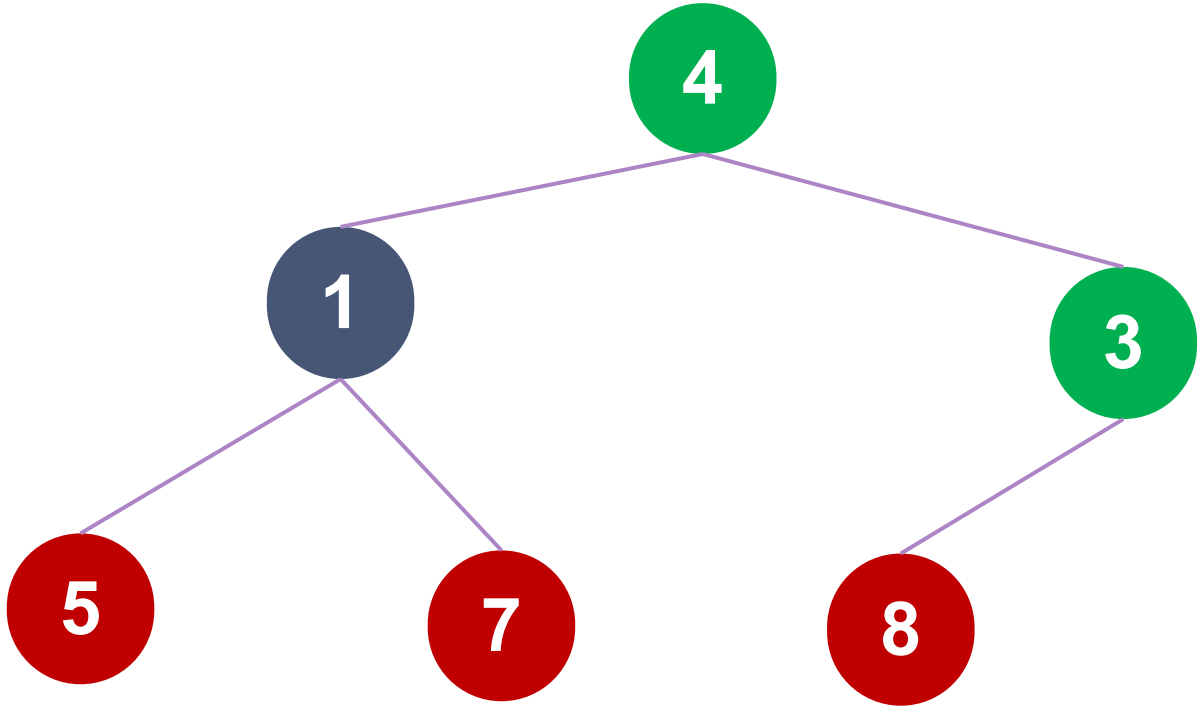
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



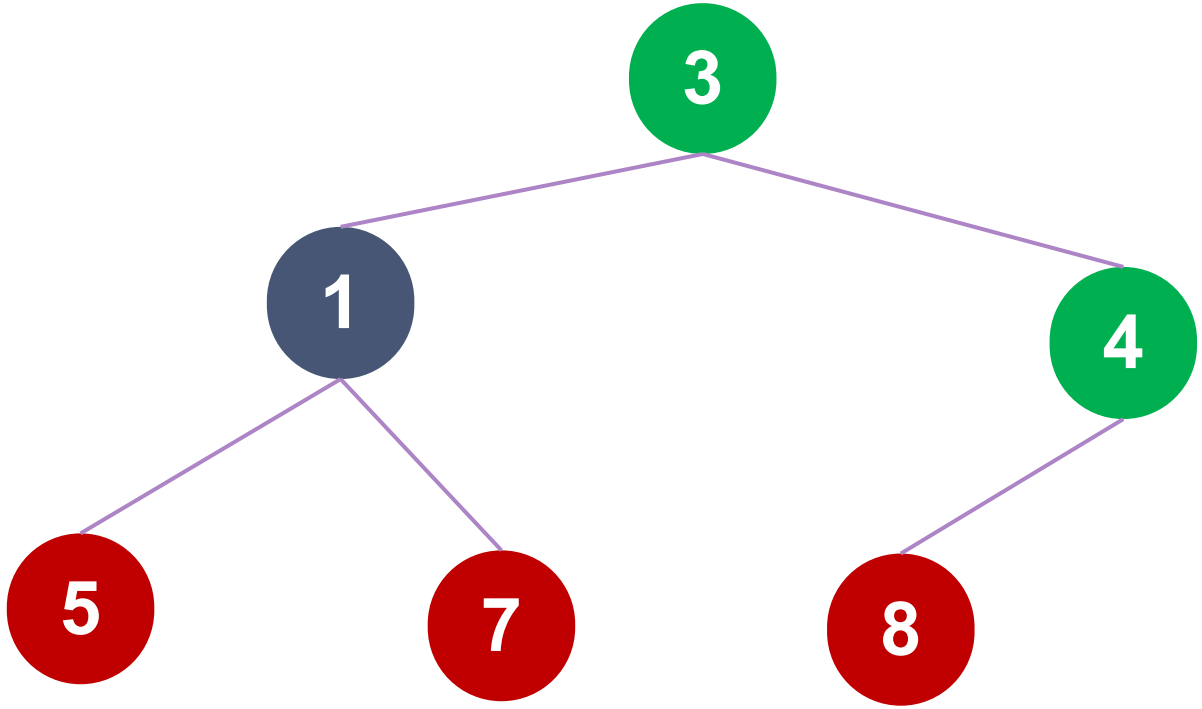
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



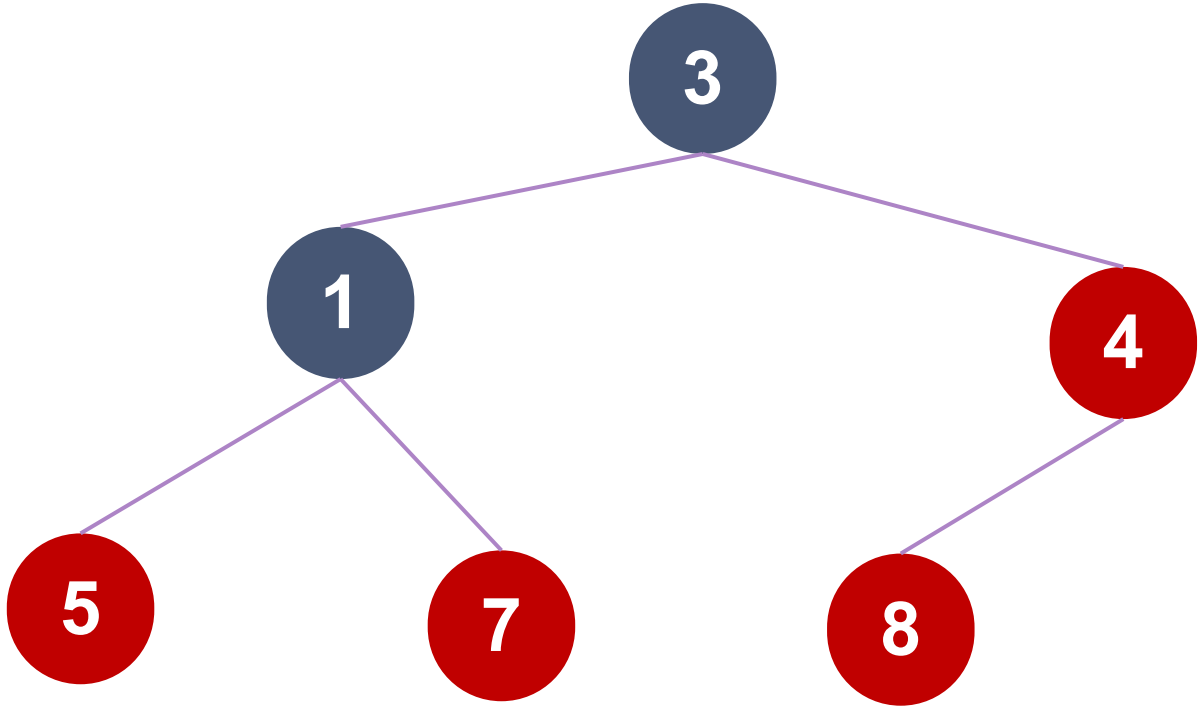
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



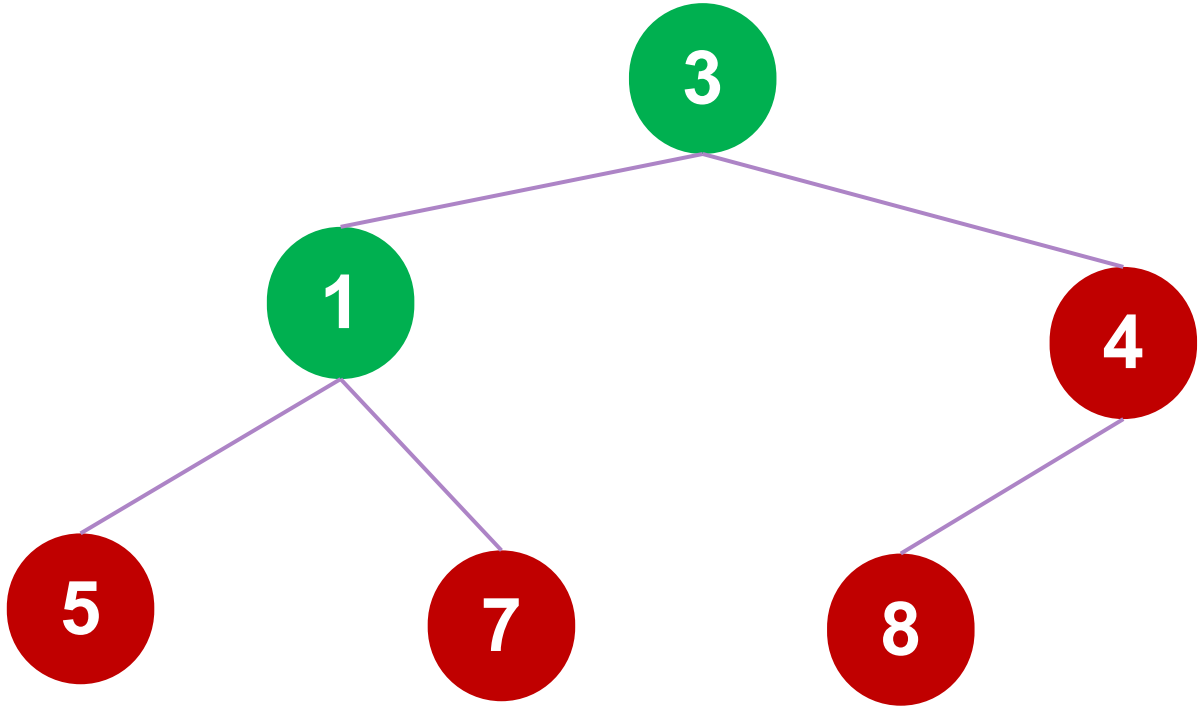
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



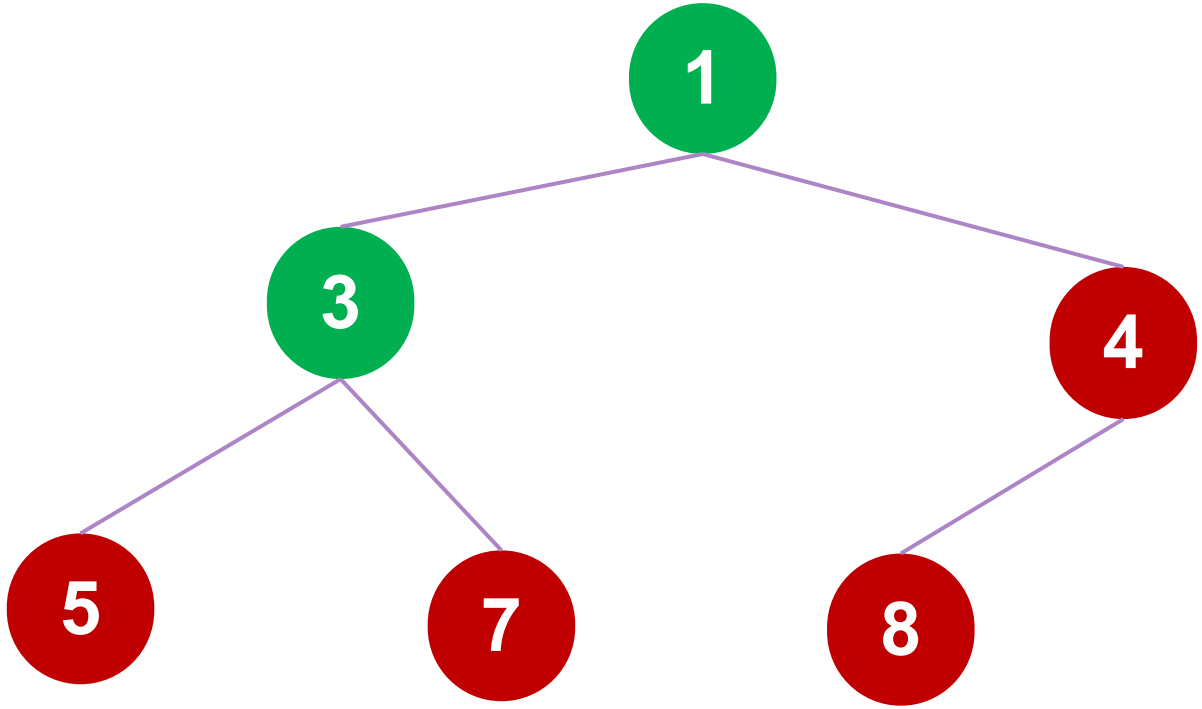
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



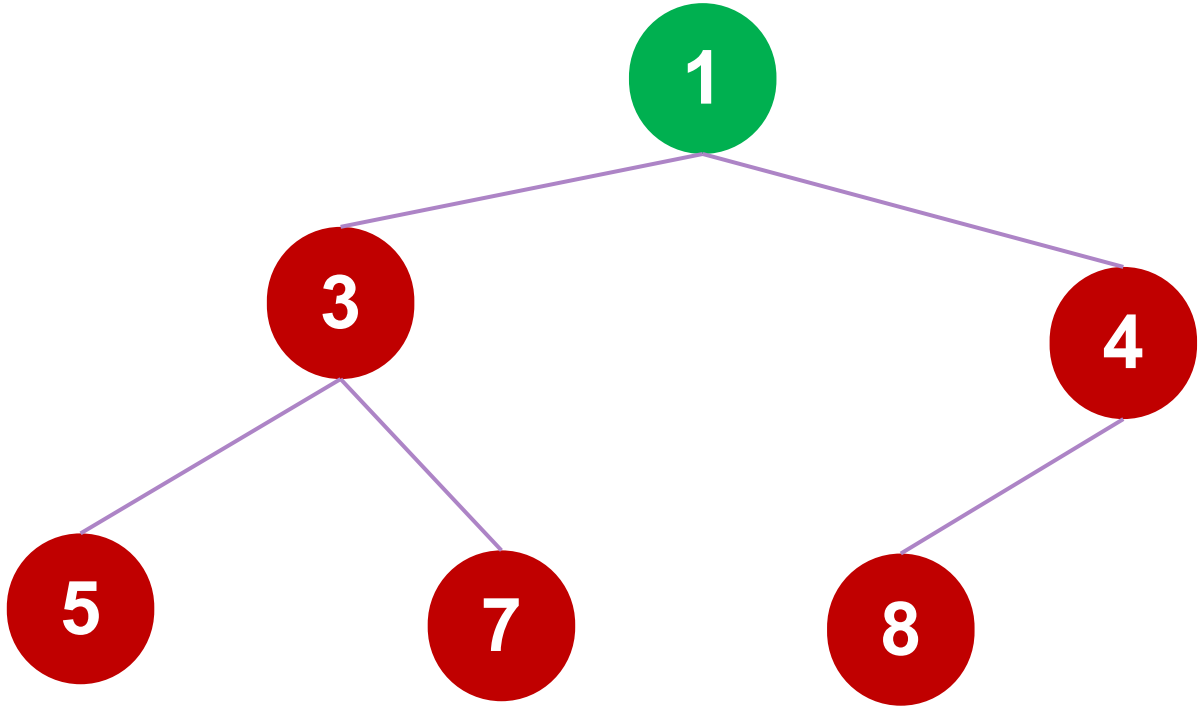
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



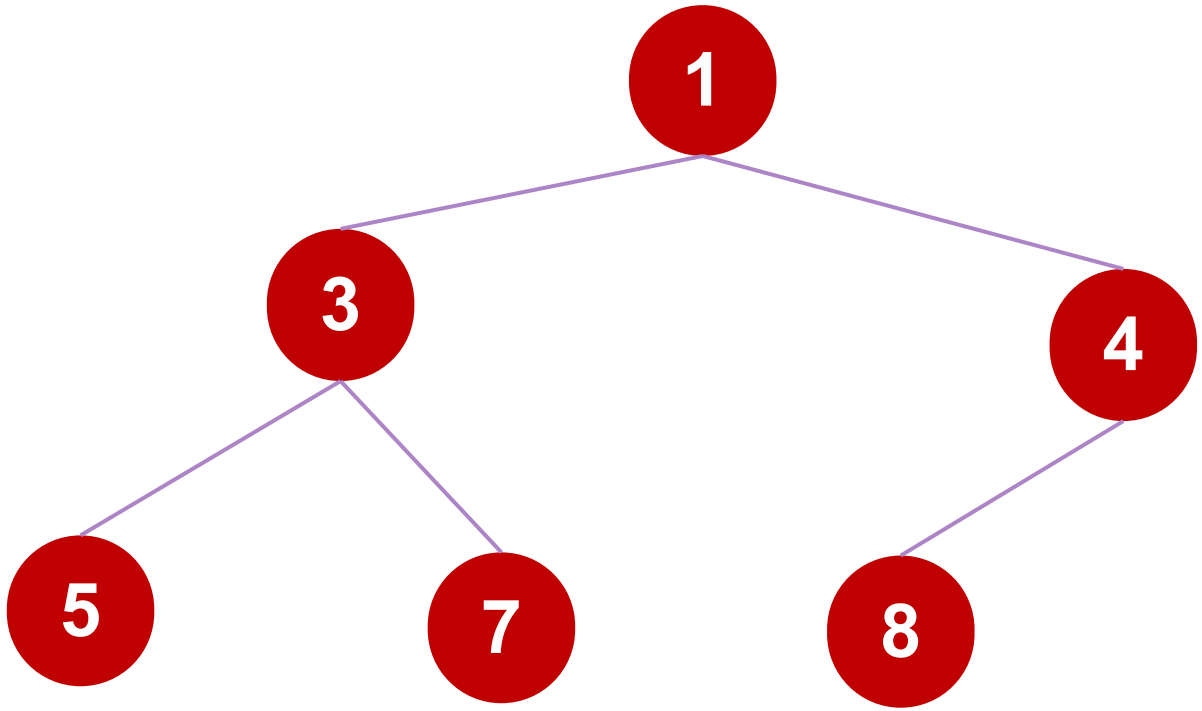
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



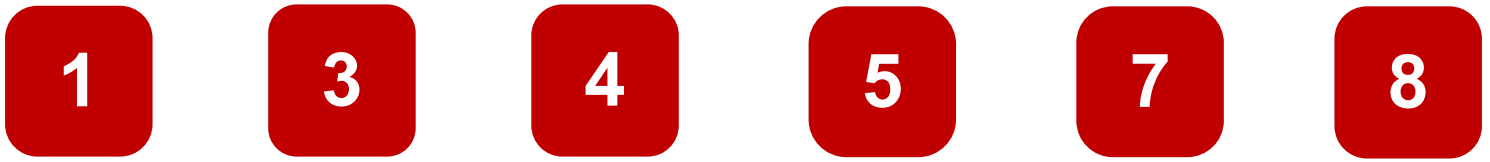
7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Hàm hiệu chỉnh heap

```
void Shift (int a[], int l, int r) {
        int x, i, j;
        i = l;
        j = 2*i + 1;
        x = a[i];
        while (j <= r) {
              if(j<r)
                    if(a[j]<a[j+1]) //Tìm phần tử lớn nhất giữa a[j] va a[j+1]
                                j++; //Lưu chỉ số của phần tử lớn nhất
                    if(a[j]<=x)
                          return;
                    else {
                          a[i]=a[j]; a[j]=x;
                          i=j; j=2*i+1; x=a[i];
                    }
        }
}
```

7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Hàm tạo heap từ dãy ban đầu:

```
void CreateHeap (int a[], int n) {
        int l = n/2-1;
        while (l >= 0) {
                Shift (a, l, n-1);
                l = l-1;
        }
}
```

7 SẮP XẾP VUN ĐỐNG – HEAP SORT

Hàm sắp xếp Heap Sort:

```
void HeapSort (int a[], int n) {
        int r;
        CreateHeap (a, n);
        r = n-1;
        while (r>0) {
                int t = a[0]; a[0]= a[r]; a[r]=t;
                r--;
                if (r>0)
                        Shift (a, 0, r);
        }
}
```

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Shell Sort là giải thuật sắp xếp dựa trên giải thuật sắp xếp chèn trực tiếp (Insertion Sort). Shell Sort tránh các trường hợp phải tráo đổi vị trí của hai phần tử cách xa nhau trong giải thuật sắp xếp chọn.

Ý tưởng của giải thuật sắp xếp Shell Sort:

- Phân hoạch dãy ban đầu thành các dãy con.
- Sắp xếp các dãy con theo phương pháp chèn trực tiếp.
- Dùng phương pháp chèn trực tiếp sắp xếp lại cả dãy.

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Cách thực hiện giải thuật:

- Phân chia dãy ban đầu thành những dãy con gồm các phần tử ở cách nhau h vị trí.
- Dãy ban đầu: a_1, a_2, ..., a_n được xem như sự xen kẽ của các dãy con sau:
  - Dãy con thứ nhất: a_1, a_{h+1}, a_{2h+1}, ...
  - Dãy con thứ hai: a_2, a_{h+2}, a_{2h+2}, ...
  - ....
  - Dãy con thứ h: a_h, a_{2h}, a_{3h}, ...
- Tiến hành sắp xếp các phần tử trong cùng dãy con sẽ làm cho các phần tử được đưa về vị trí đúng tương đối.
- Giảm khoảng cách h để tạo thành các dãy con mới.
- Dừng khi h = 1.

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Các bước thực hiện giải thuật:

Bước 1:

- Chọn k khoảng cách h[1], h[2], ..., h[k].
- i = 1.

Bước 2:

- Phân chia dãy ban đầu thành các dãy con cách nhau h[i] phần tử.
- Sắp xếp từng dãy con bằng phương pháp chèn trực tiếp.

Bước 3:

- i = i+1; Nếu i > k thì dừng, ngược lại ta lặp lại bước 2.

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Hàm minh họa giải thuật Shell Sort:

```
void ShellSort (int a[], int n, int h[], int k){
        int step, i, j, x, len;
        for (step = 0; step <k; step++) {
                len = h[step];
                for (i = len; i<n; i++) {
                        x = a[i]; j = i-len;
                        while ((x<a[j]) && (j>=0)) {
                                a[j+len] = a[j];
                                j = j - len;
                        }
                        a[j+len] = x;
                }
        }
}
```

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

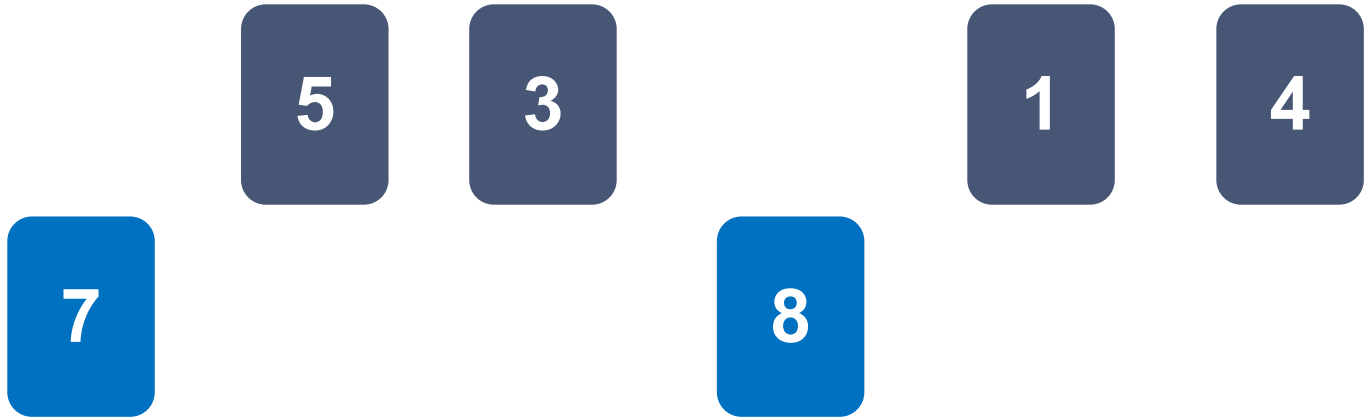
Chọn tập khoảng cách k **3** **2**

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k 2

Khoảng cách phân hoạch 3



8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **2**

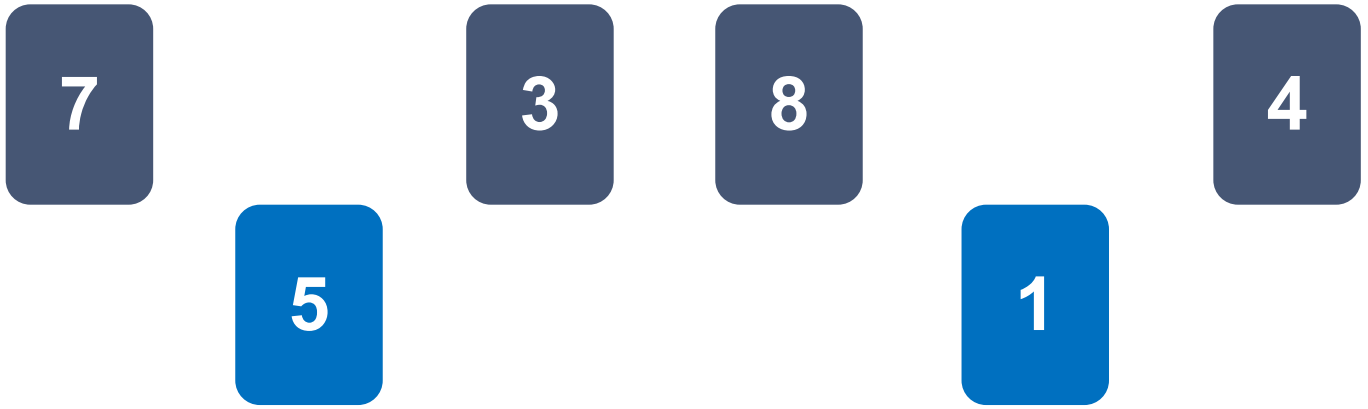
Khoảng cách phân hoạch **3**

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **2**

Khoảng cách phân hoạch **3**



8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **2**

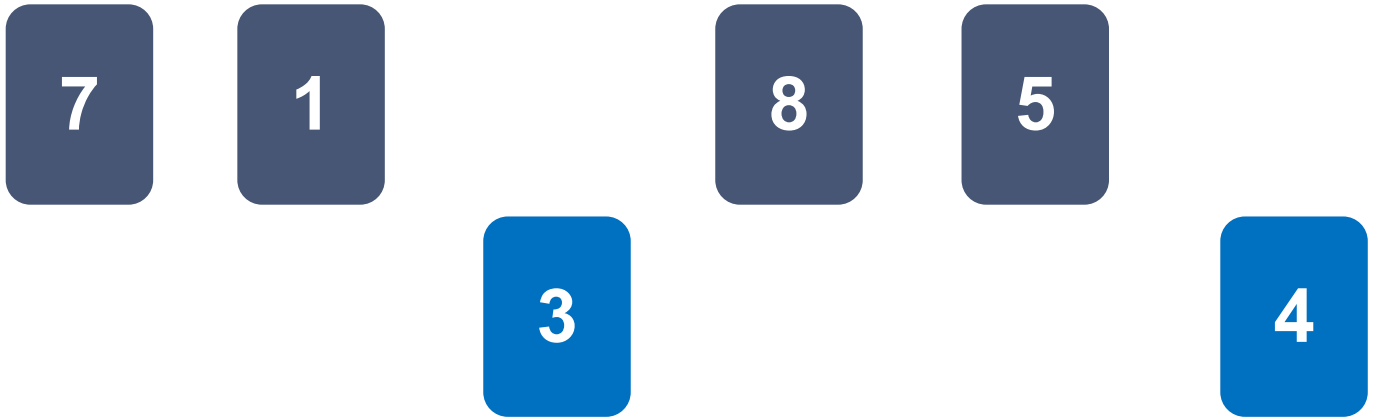
Khoảng cách phân hoạch **3**

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **2**

Khoảng cách phân hoạch **3**



8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **2**

Khoảng cách phân hoạch **3**

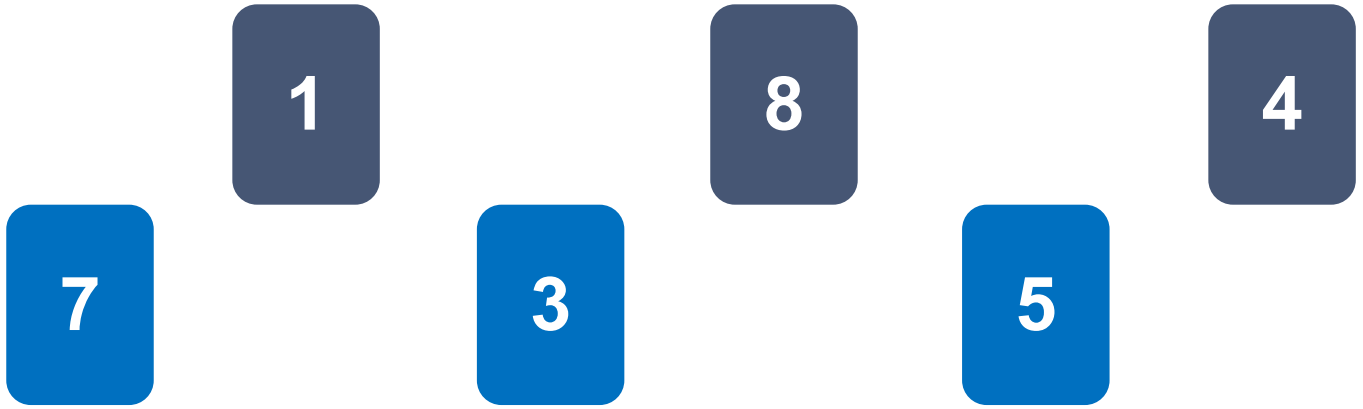
7	1	3	8	5	4

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**



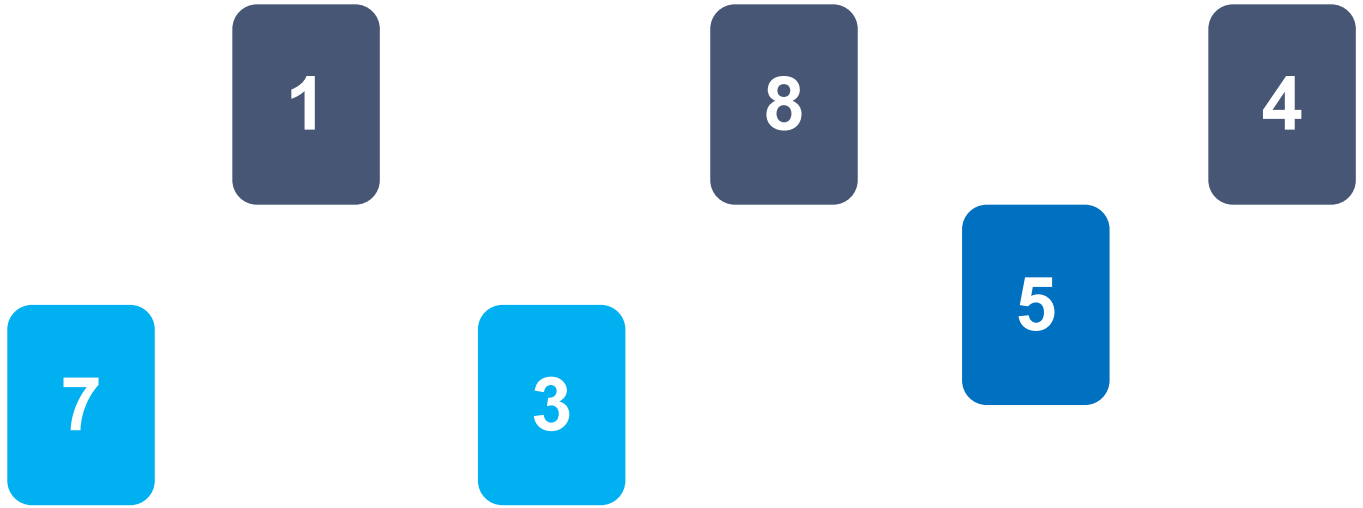
Sắp xếp dãy con 7, 3, 5 bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**



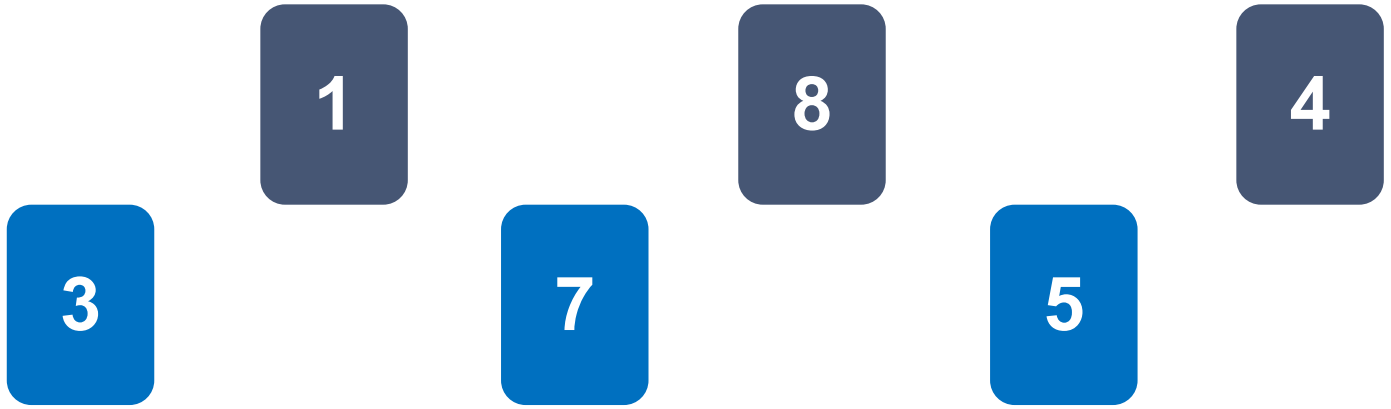
Sắp xếp dãy con 7, 3, 5 bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**



Sắp xếp dãy con 7, 3, 5 bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**

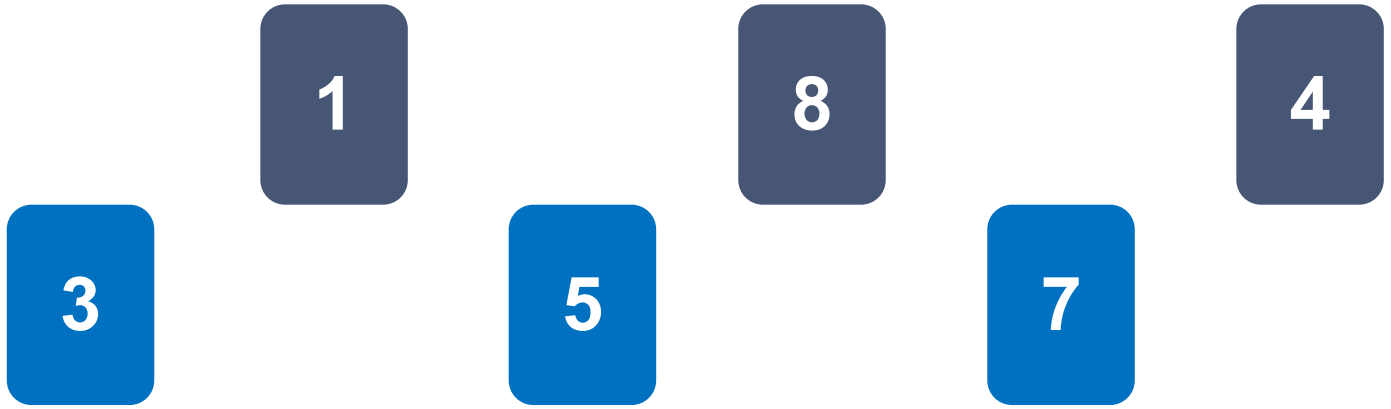
Sắp xếp dãy con 7, 3, 5 bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**



Sắp xếp dãy con 7, 3, 5 bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

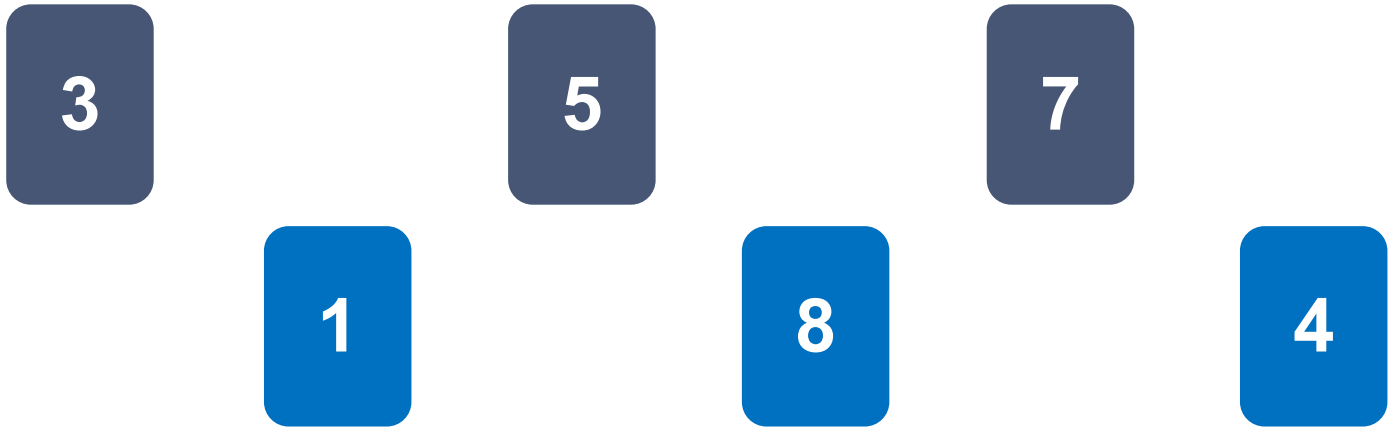
Khoảng cách phân hoạch **2**

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**



Sắp xếp dãy con 1, 8, 4 bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**

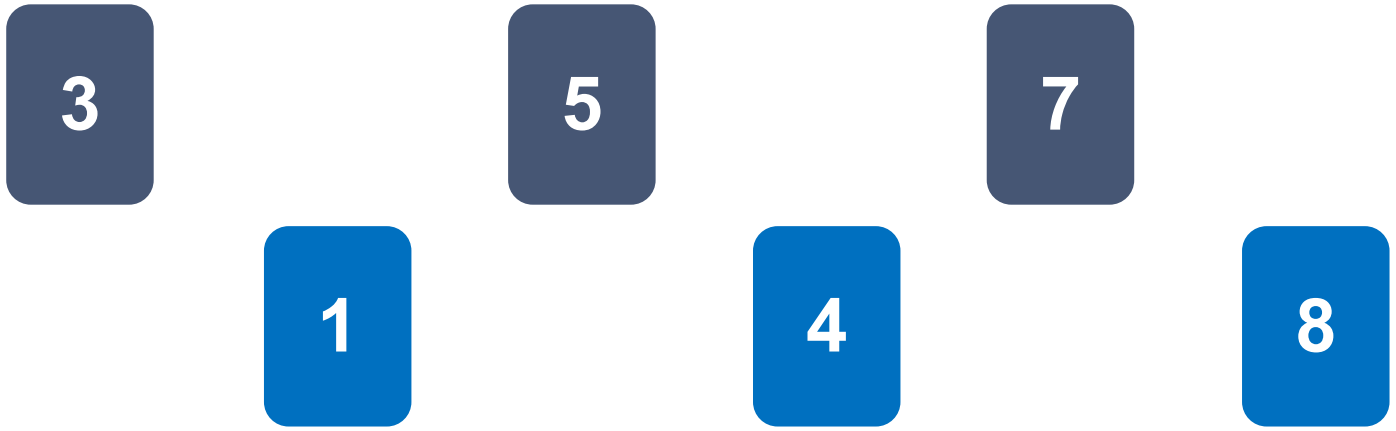
Sắp xếp dãy con 1, 8, 4 bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**



Sắp xếp dãy con 1, 8, 4 bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**

Sắp xếp cả dãy bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**

Sắp xếp cả dãy bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**

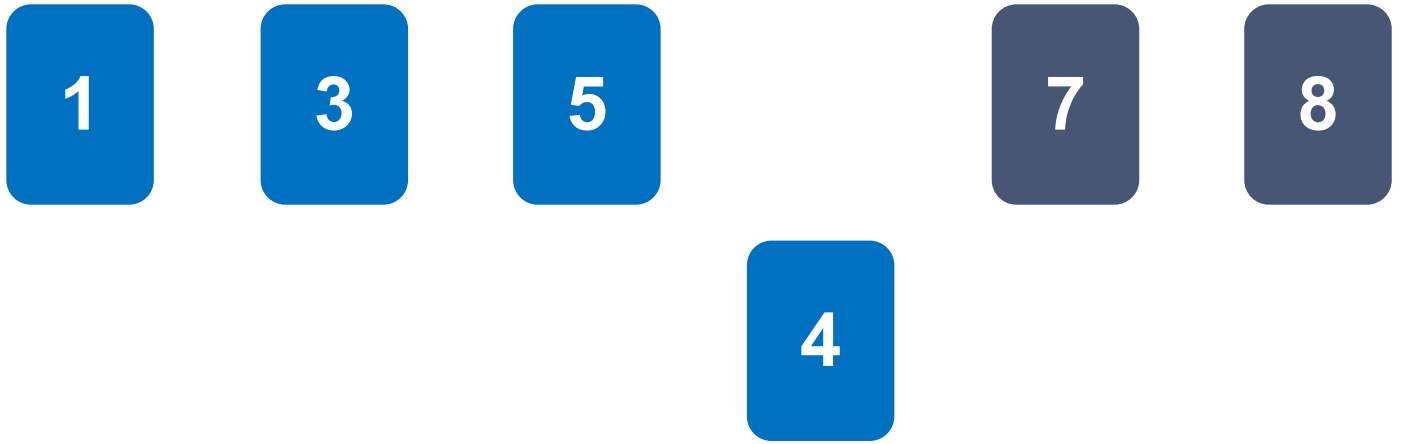
Sắp xếp cả dãy bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**



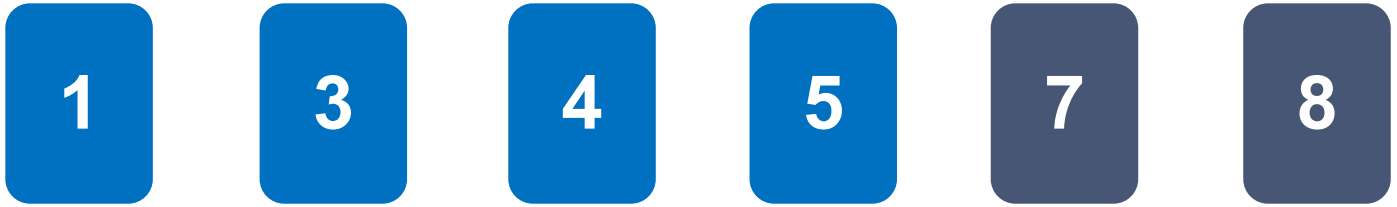
Sắp xếp cả dãy bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**



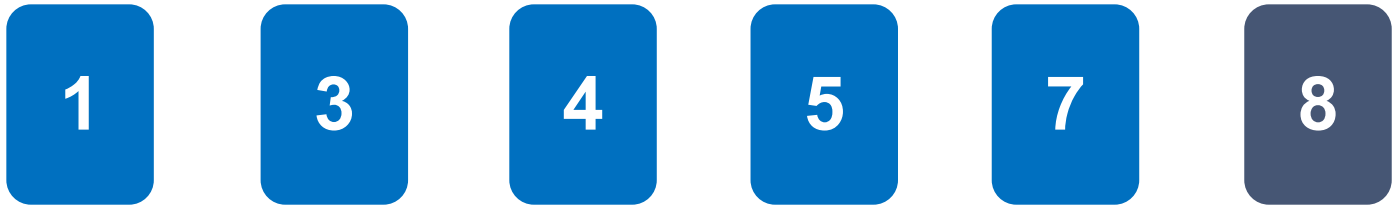
Sắp xếp cả dãy bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**



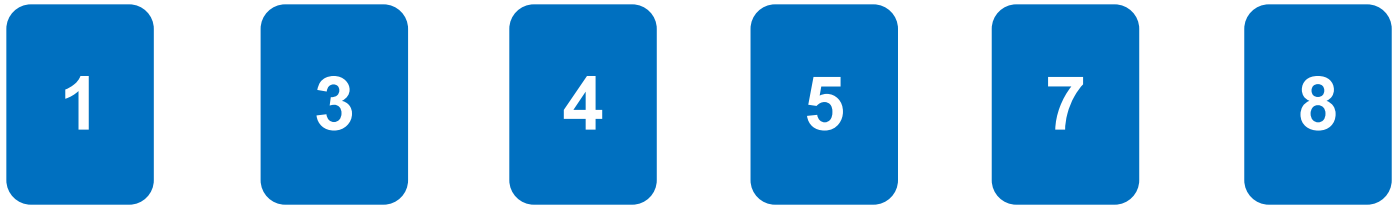
Sắp xếp cả dãy bằng chèn trực tiếp

8 SẮP XẾP PHÂN HOẠCH – SHELL SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

Chọn tập khoảng cách k **3**

Khoảng cách phân hoạch **2**



Sắp xếp cả dãy bằng chèn trực tiếp

9 SẮP XẾP TRỘN – MERGE SORT

Ý tưởng của giải thuật sắp xếp trộn (Merge Sort): Ý tưởng của giải thuật là trộn 2 dãy đã được sắp xếp thành 1 dãy mới cũng được sắp xếp:

- Đầu tiên ta coi mỗi phần tử của dãy là 1 danh sách con gồm 1 phần tử đã được sắp.
- Tiếp theo tiến hành trộn từng cặp 2 dãy con 1 phần tử kề nhau để tạo thành các dãy con 2 phần tử được sắp xếp.
- Các dãy con 2 phần tử được sắp này lại được trộn với nhau tạo thành dãy con 4 phần tử được sắp.
- Quá trình tiếp tục đến khi chỉ còn 1 dãy con duy nhất được sắp, đó chính là dãy ban đầu.

9 SẮP XẾP TRỘN – MERGE SORT

Các bước thực hiện:

Bước 1: k = 1; //k là chiều dài của dãy con trong bước hiện hành

Bước 2: Tách dãy $a_0, a_1, ..., a_{n-1}$ thành 2 dãy b, c theo nguyên tắc luân phiên từng nhóm k phần tử:

- $b = a_0, a_k, a_{2k}, a_{3k}, \ldots$
- $c = a_1, a_{k+1}, a_{2k+1}, a_{3k+1}, \ldots$

Bước 3: Trộn từng cặp dãy con gồm k phần tử của 2 dãy b, c vào a.

Bước 4:

- *k = k*2.*
- *Nếu k < n thì trở lại bước 2.*
- *Ngược lại dừng thuật toán.*

9 SẮP XẾP TRỘN – MERGE SORT

Hàm phân phối đều luân phiên các dãy con độ dài k từ mảng a vào hai mảng con b và c:

```
void Distribute (int a[], int n, int b[], int c[], int k, int sizeB, int sizeC)
{
        int ib, ic, i, j;
        i = ib = ic = 0;
        while (i*k<n) {
                for (j=0; j<k; j++) {
                        if (i%2 == 0 && i*k+j<n && ib<sizeB) {
                                b[ib] = a[i*k+j]; ib++;
                        }
                        if (i%2 == 1 && i*k+j<n && ic<sizeC) {
                                c[ic]=a[i*k+j]; ic++;
                        }
                }
                i++;
        }
}
```

9 SẮP XẾP TRỘN – MERGE SORT

Hàm trộn mảng b và c vào mảng a:

```
void Merge (int a[], int b[], int c[], int k, int sizeB, int sizeC) {
    int maxSize = sizeB > sizeC ? sizeB : sizeC;
    int totalBlockMerge = maxSize%k == 0 ? maxSize/k : maxSize/k + 1;
    /* totalBlockMerge là số cụm phần tử từ b và c sẽ trộn và gộp vào a*/
    int i, ia = 0, ib = 0, ic = 0;
    for (i = 0; i<totalBlockMerge; i++) {
          int limitIndexBlock = (i+1)*k;
          while (ib<limitIndexBlock && ic<limitIndexBlock) {
                if (ib==sizeB || ic == sizeC) break;
                if (b[ib]<c[ic]) { a[ia] = b[ib]; ib++; }
                else { a[ia] = c[ic]; ic++; }
                ia++;
          }
          while (ib<limitIndexBlock && ib<sizeB) {a[ia] = b[ib]; ia++; ib++; }
          while (ic<limitIndexBlock && ic<sizeC) {a[ia] = c[ic]; ia++; ic }
}}
```

9 SẮP XẾP TRỘN – MERGE SORT

Hàm sắp xếp trộn Merge Sort:

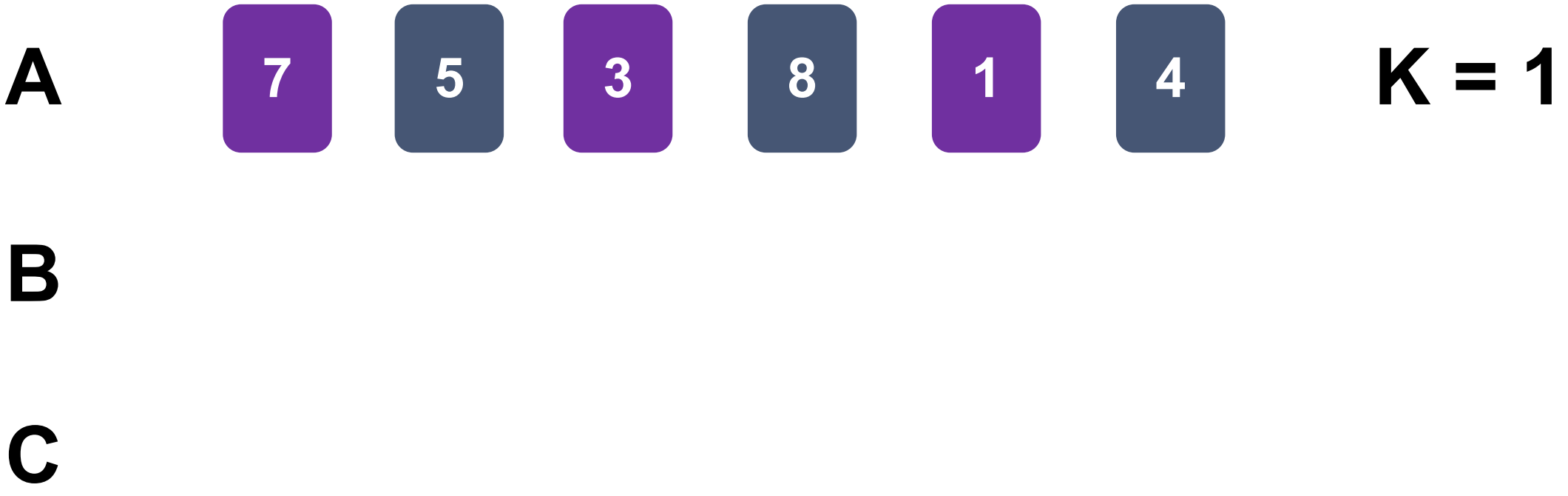
```
void MergeSort (int a[], int sizeA){
        int k, i;
        for (k = 1; k < sizeA; k *= 2) {
                int sizeB, sizeC;
                if (2*k >= sizeA) sizeB = k;
                else {
                        int x = sizeA/k,  y = sizeA%k;
                        if (x%2 == 0) sizeB = (x/2)*k + y;
                        else sizeB = (x/2+1)*k;
                }
                sizeC = sizeA - sizeB;
                int b[sizeB], c[sizeC];
                Distribute (a, sizeA, b, c, k, sizeB, sizeC);
                Merge (a, b, c, k, sizeB, sizeC);
        }
}
```

9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

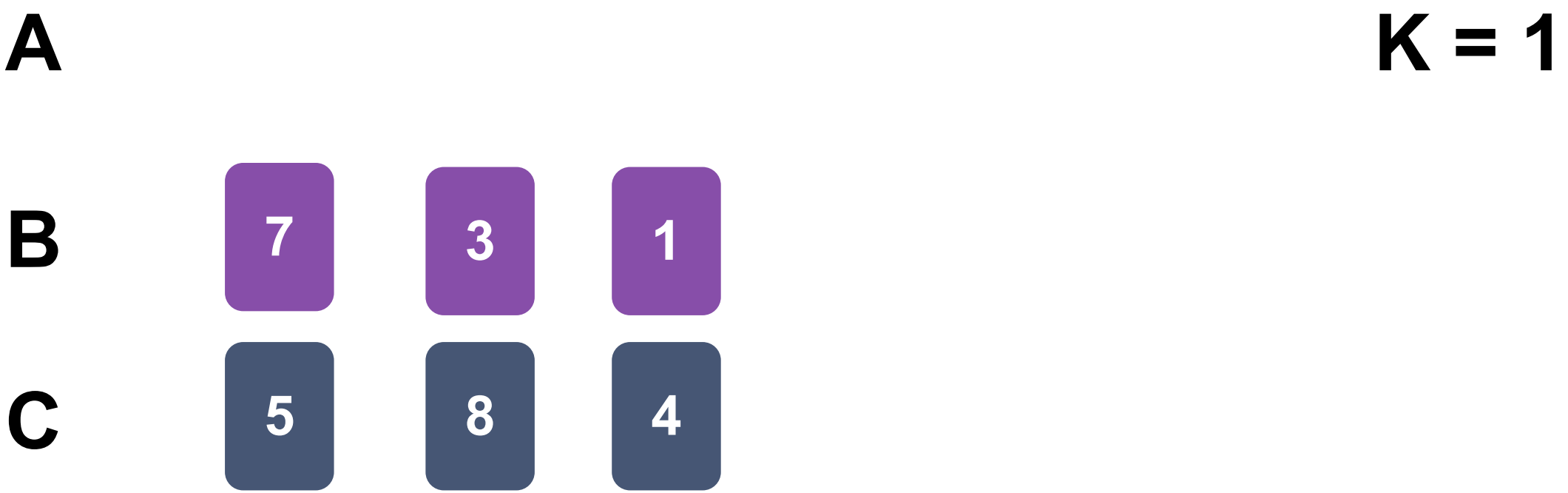
9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



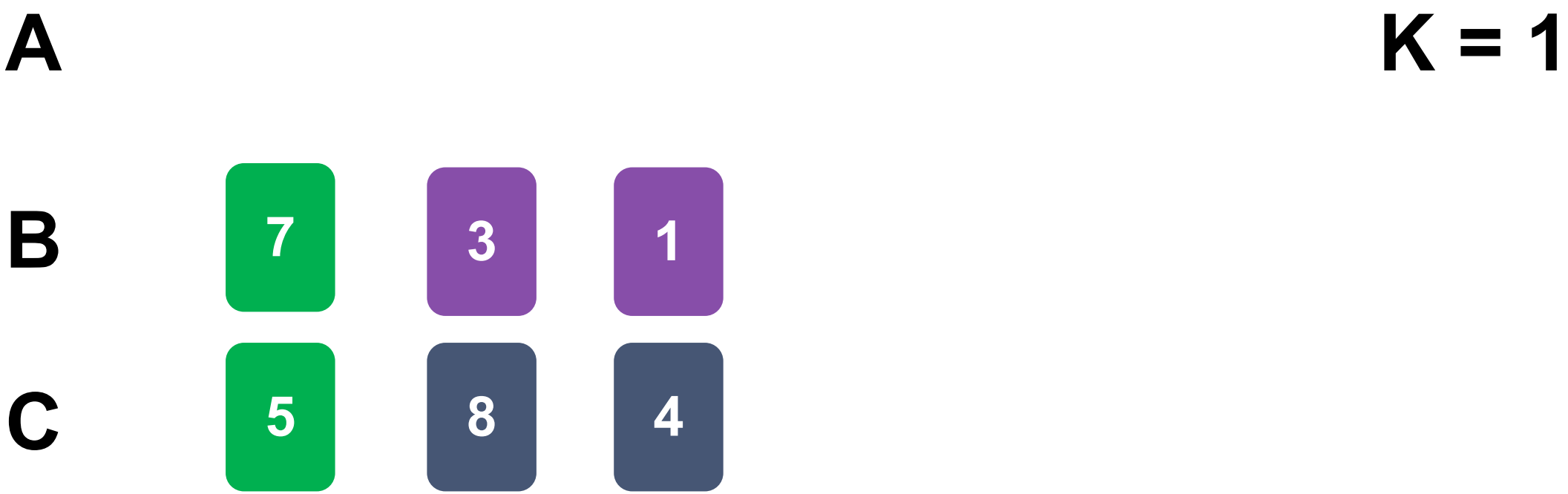
9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

K = 1

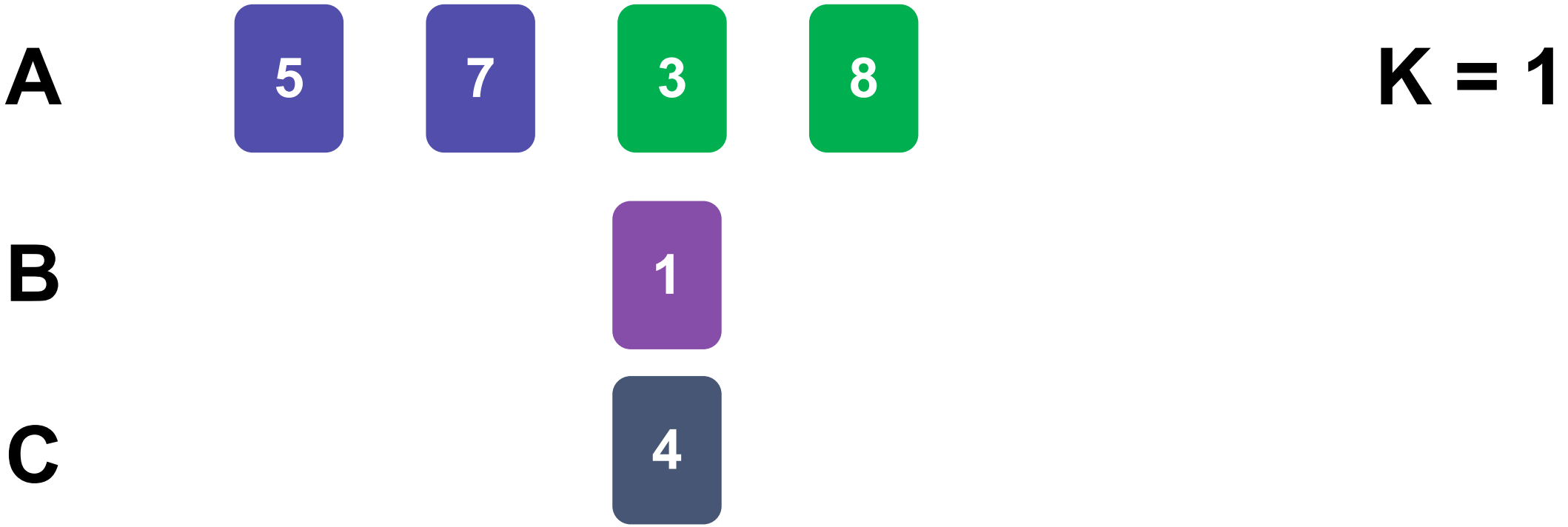
9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

K = 1

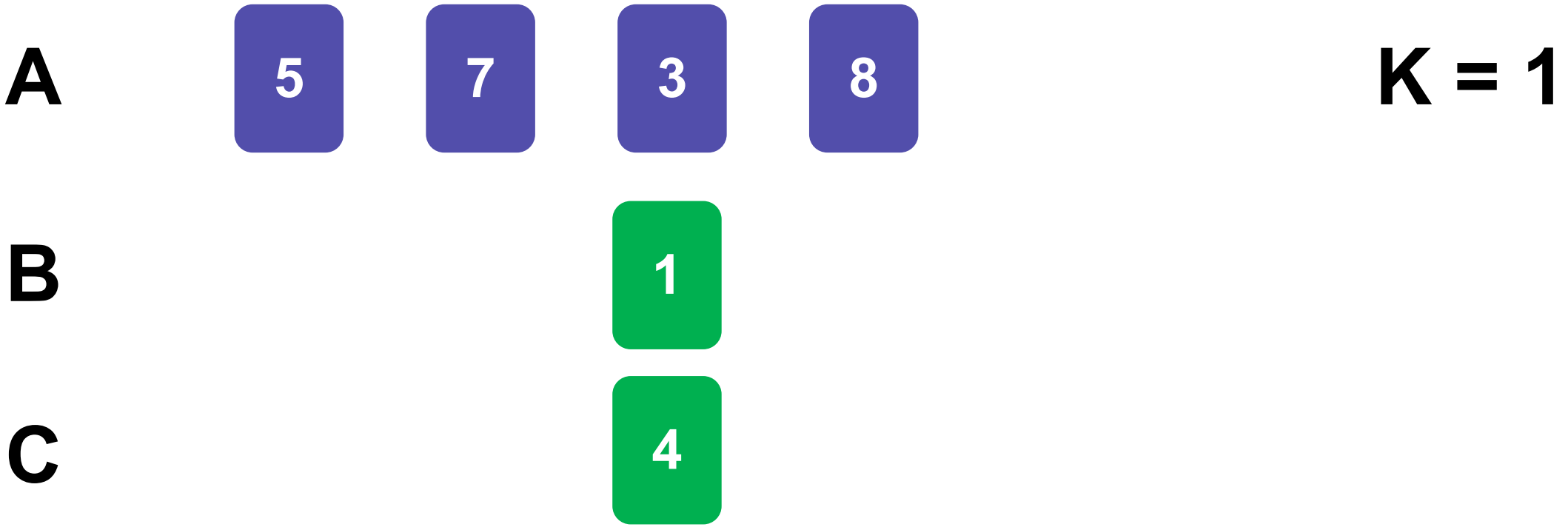
9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



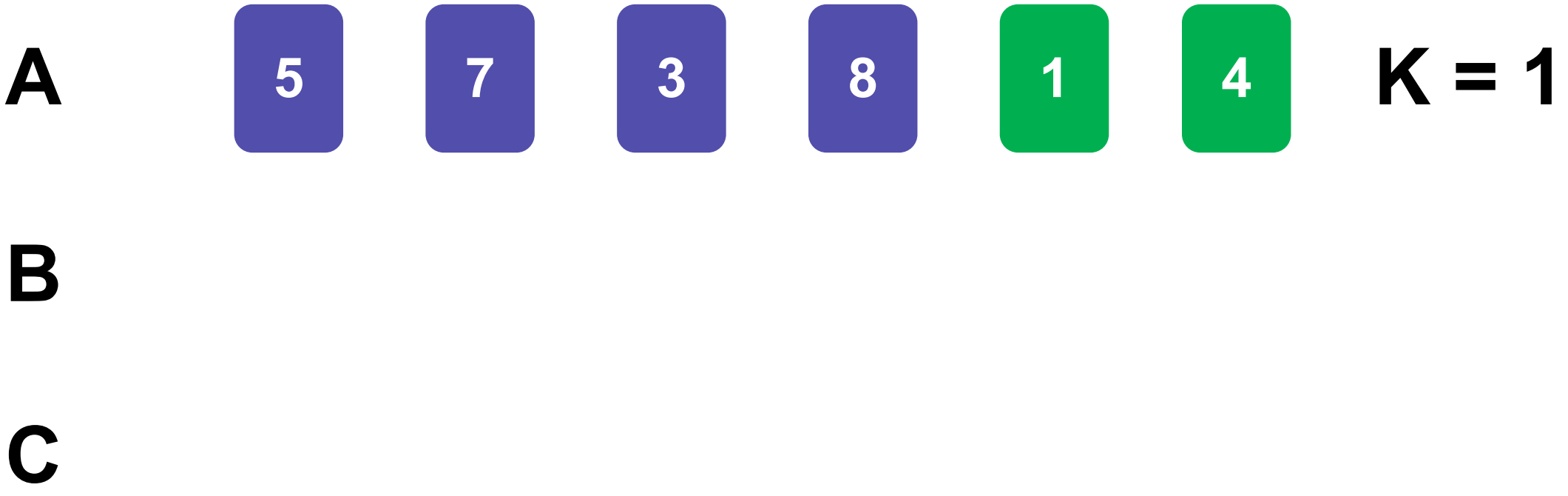
9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

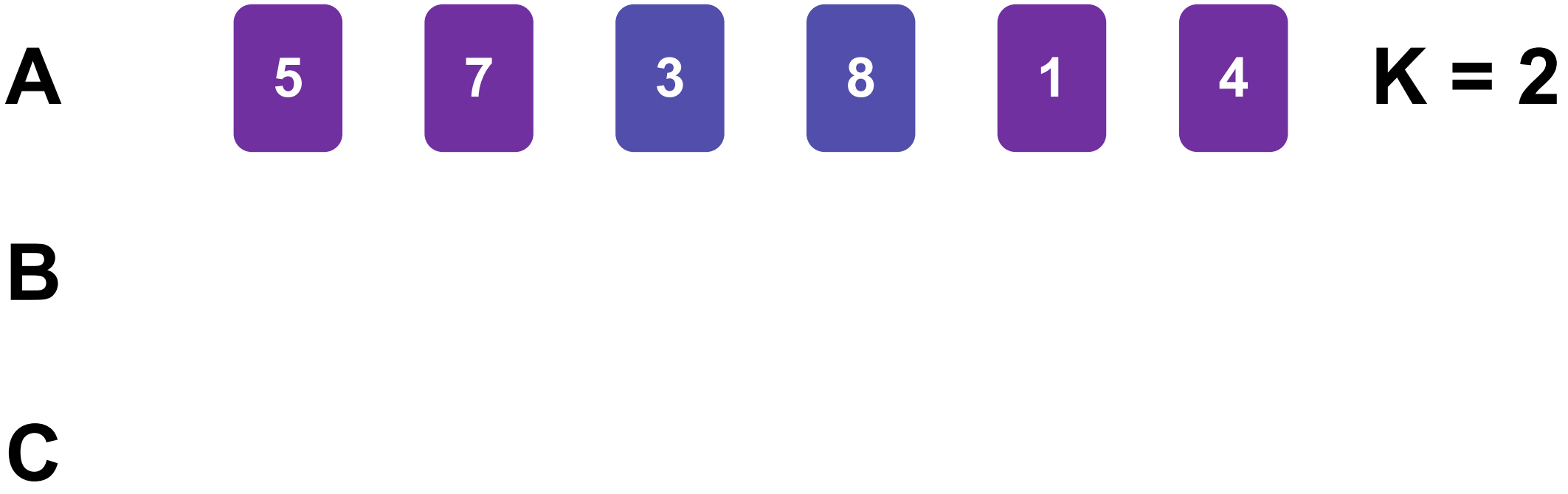


9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



9 SẮP XẾP TRỘN – MERGE SORT

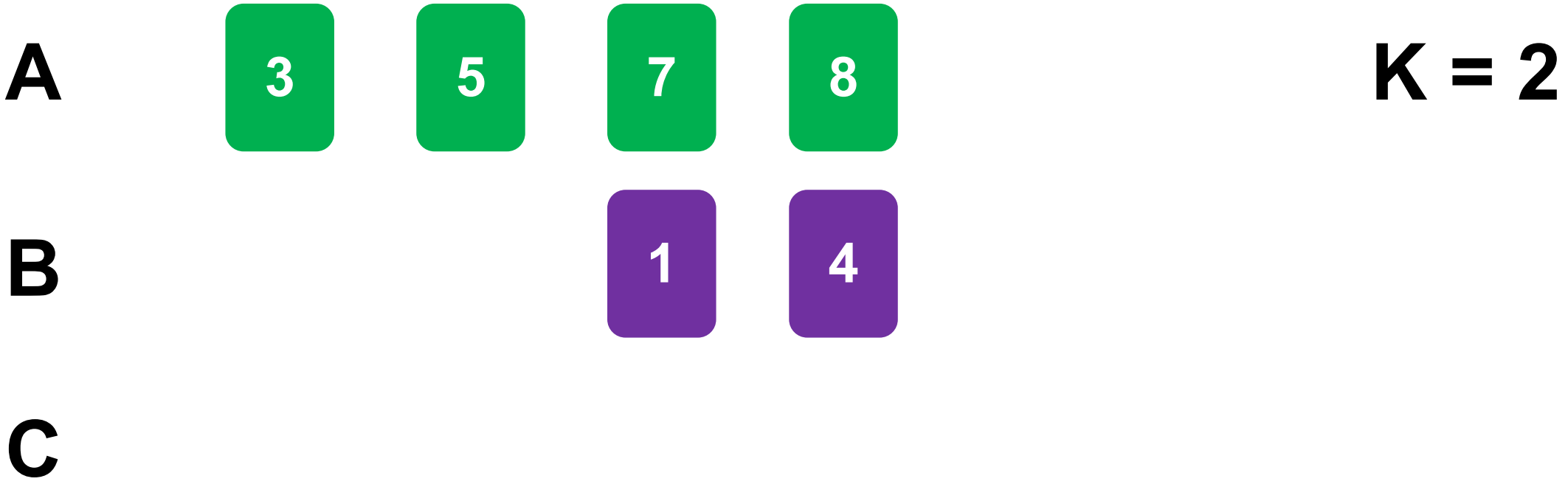
Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

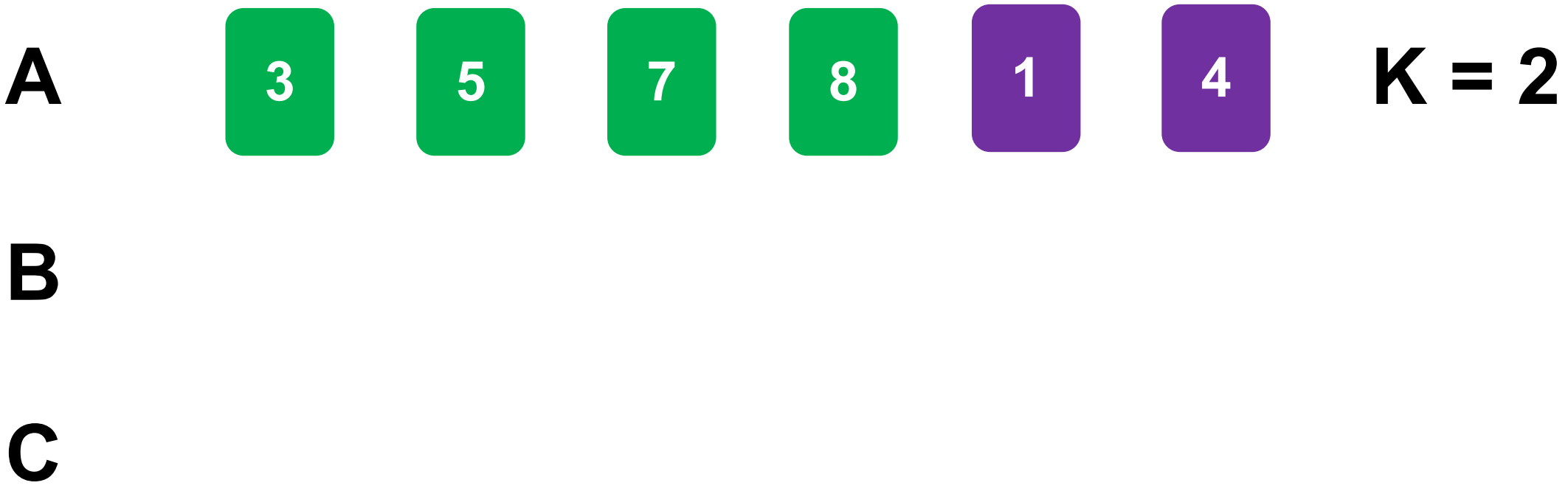
9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4



9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

9 SẮP XẾP TRỘN – MERGE SORT

Ví dụ: sắp xếp dãy số sau tăng dần: 7, 5, 3, 8, 1, 4

10 ĐÁNH GIÁ ĐỘ PHỨC TẠP CỦA CÁC THUẬT TOÁN SẮP XẾP

Các giải thuật sắp xếp được đánh giá bằng độ phức tạp thời gian, độ phức tạp không gian và độ ổn định.

- Độ phức tạp thời gian là thời gian mà giải thuật sắp xếp cần để sắp xếp dữ liệu.
- Độ phức tạp không gian là lượng bộ nhớ mà giải thuật sắp xếp sử dụng để sắp xếp dữ liệu.
- Độ ổn định là tính chất của giải thuật sắp xếp giữ nguyên thứ tự của các phần tử có cùng giá trị trong dãy ban đầu.

10 ĐÁNH GIÁ ĐỘ PHỨC TẠP CỦA CÁC THUẬT TOÁN SẮP XẾP

- Đối với danh sách có kích thước nhỏ, các giải thuật sắp xếp đơn giản như bubble sort hay insertion sort là lựa chọn phù hợp. Những giải thuật này có thời gian thực thi ngắn, nhưng hiệu quả không cao đối với danh sách có kích thước lớn.
- Đối với danh sách có kích thước lớn, các giải thuật sắp xếp hiệu quả hơn như merge sort hay quick sort nên được ưu tiên sử dụng. Những giải thuật này có thời gian thực thi dài hơn các giải thuật sắp xếp đơn giản, nhưng hiệu quả cao hơn đáng kể đối với danh sách có kích thước lớn.
- Nếu danh sách đã được sắp xếp một phần, các giải thuật sắp xếp như insertion sort hay heap sort có thể tận dụng điều này để cải thiện hiệu suất.
- Nếu danh sách không có thứ tự cụ thể, các giải thuật sắp xếp như bubble sort hay merge sort sẽ hoạt động tốt hơn.
- Nếu ứng dụng có giới hạn về bộ nhớ, các giải thuật sắp xếp không yêu cầu bộ nhớ tạm như insertion sort hay heap sort nên được ưu tiên sử dụng.

10 ĐÁNH GIÁ ĐỘ PHỨC TẠP CỦA CÁC THUẬT TOÁN SẮP XẾP

Giải thuật	Độ phức tạp theo thời gian			Độ phức tạp không gian
	Tốt nhất	Trung bình	Xấu nhất	
Đổi chỗ trực tiếp	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)
Chọn trực tiếp	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)
Chèn trực tiếp	O(n)	$O(n^2)$	$O(n^2)$	O(1)
Bubble Sort	O(n)	$O(n^2)$	$O(n^2)$	O(1)
Quick Sort	O(nlogn)	O(nlogn)	$O(n^2)$	O(n)
Heap Sort	O(nlogn)	O(nlogn)	O(nlogn)	O(1)
Shell Sort	O(n)	$O((nlogn)^2)$	$O((nlogn)^2)$	O(1)
Merge Sort	O(nlogn)	O(nlogn)	O(nlogn)	O(n)
Radix Sort	O(nk)	O(nk)	O(nk)	O(n+k)

BÀI TẬP CHƯƠNG 3

Bài 1. Viết chương trình sắp xếp dãy số nguyên A tăng dần và giảm dần theo các giải thuật sau (in ra màn hình quá trình sắp xếp và kết quả cuối cùng):

a. Interchange Sort
b. Selection Sort
c. Insertion Sort
d. Bubble Sort
e. Quick Sort
f. Heap Sort
g. Shell Sort
h. Merge Sort

BÀI TẬP CHƯƠNG 3

Bài 2. Cho dãy số nguyên A: 3, 9, 2, 22, 45, 6, -4, 125, 0.

Hãy liệt kê sự thay đổi của dãy A nếu nó được sắp xếp tăng dần bằng các giải thuật:

a. Interchange Sort
b. Selection Sort
c. Insertion Sort
d. Bubble Sort
e. Quick Sort
f. Heap Sort
g. Shell Sort
h. Merge Sort

BÀI TẬP CHƯƠNG 3

Bài 3. Nhập dữ liệu cho mảng cấu trúc gồm n sinh viên, mỗi sinh viên gồm các thông tin: Họ tên, mã số sinh viên, điểm tổng kết.

a. Sắp xếp mảng theo mã số sinh viên tăng dần.

b. Sắp xếp mảng theo điểm tổng kết giảm dần.

c. Sắp xếp mảng theo tên sinh viên (thứ tự A, B, C...).