**Divide and Conquer, Sorting and Searching, and Randomized Algorithms**

**I>Merge Sort**

**Đ/N:** Giống như Quick sort, Merge sort là một thuật toán chia để trị. Thuật toán này chia mảng cần sắp xếp thành 2 nửa. Tiếp tục lặp lại việc này ở các nửa mảng đã chia. Sau cùng gộp các nửa đó thành mảng đã sắp xếp. Hàm merge() được sử dụng để gộp hai nửa mảng. Hàm merge(arr, l, m, r) là tiến trình quan trọng nhất sẽ gộp hai nửa mảng thành 1 mảng sắp xếp, các nửa mảng là arr[l…m] và arr[m+1…r] sau khi gộp sẽ thành một mảng duy nhất đã sắp xếp.

**Mã giả minh họa:**

mergeSort(arr[], l, r)

If r > l

1. Tìm chỉ số nằm giữa mảng để chia mảng thành 2 nửa:

middle m = (l+r)/2

2. Gọi đệ quy hàm mergeSort cho nửa đầu tiên:

mergeSort(arr, l, m)

3. Gọi đệ quy hàm mergeSort cho nửa thứ hai:

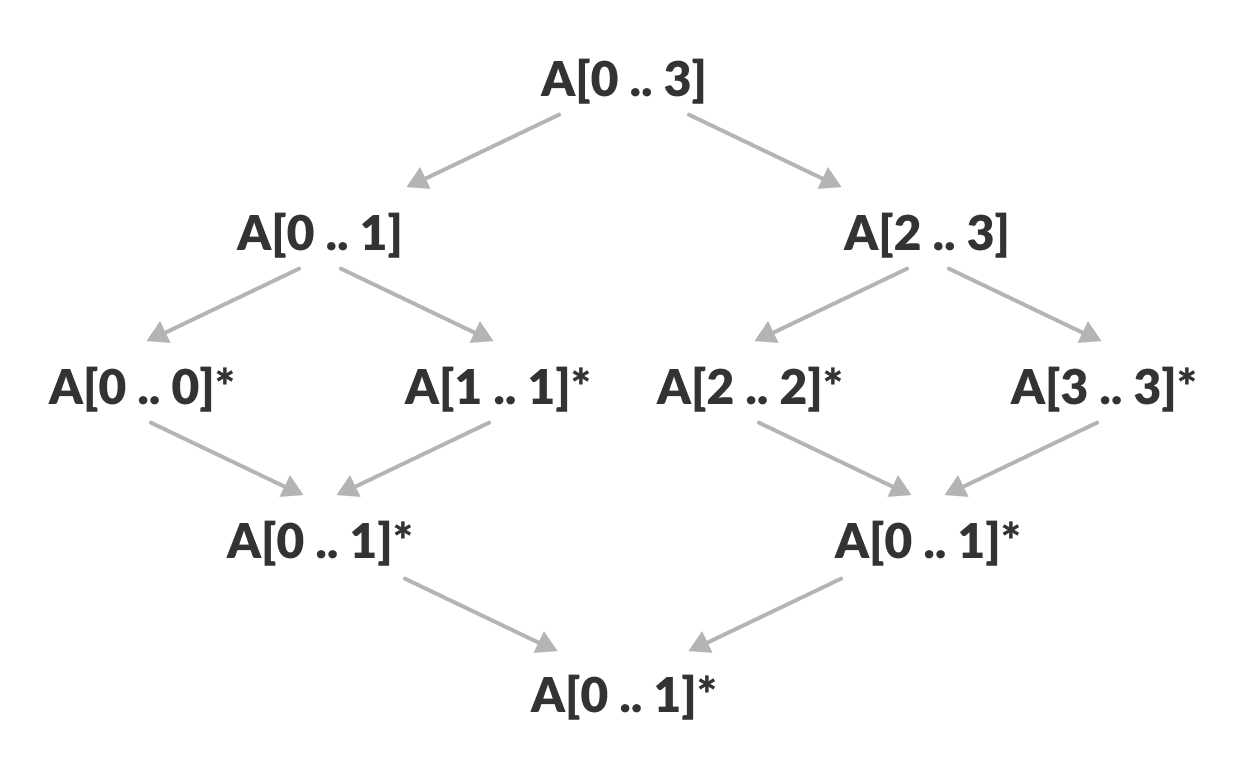
mergeSort(arr, m+1, r)

4. Gộp 2 nửa mảng đã sắp xếp ở (2) và (3):

merge(arr, l, m, r)

**Ví dụ:**





**\*Các bước:**

**Bước 1: Tạo bản sao của các mảng con được sắp xếp**

// Create L ← A[p..q] and M ← A[q+1..r]

int n1 = q - p + 1 = 3 - 0 + 1 = 4;

int n2 = r - q = 5 - 3 = 2;

int L[4], M[2];

for (int i = 0; i < 4; i++)

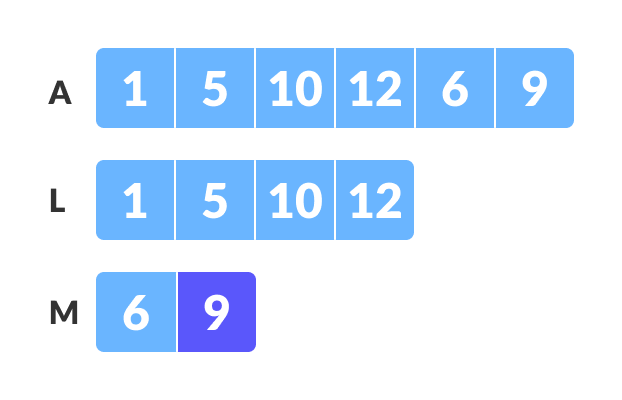
L[i] = arr[p + i];

// L[0,1,2,3] = A[0,1,2,3] = [1,5,10,12]

for (int j = 0; j < 2; j++)

M[j] = arr[q + 1 + j];

// M[0,1] = A[4,5] = [6,9]



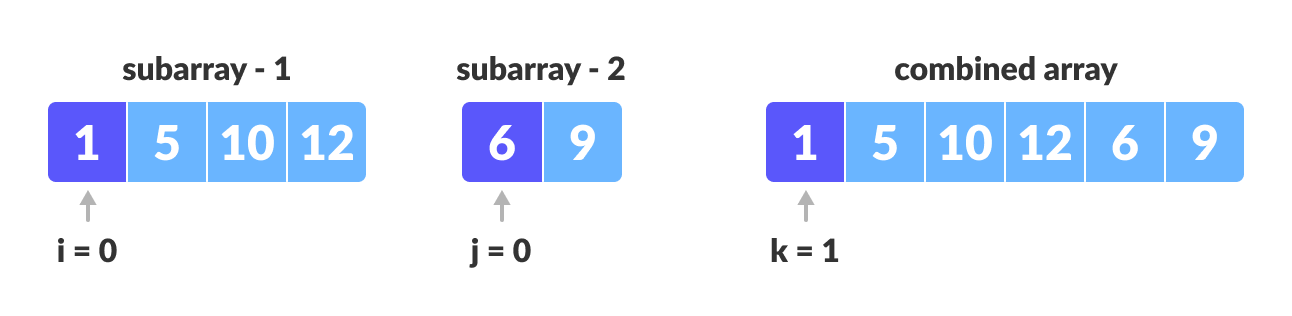
**Bước 2: Duy trì chỉ số hiện tại của mảng con và mảng chính**

int i, j, k;

i = 0;

j = 0;

k = p;



**Bước 3: Cho đến khi chúng ta đến cuối L hoặc M, hãy chọn phần tử lớn hơn trong số L và M và đặt chúng vào đúng vị trí tại A [p..r]**

while (i < n1 && j < n2) {

if (L[i] <= M[j]) {

arr[k] = L[i]; i++;

}

else {

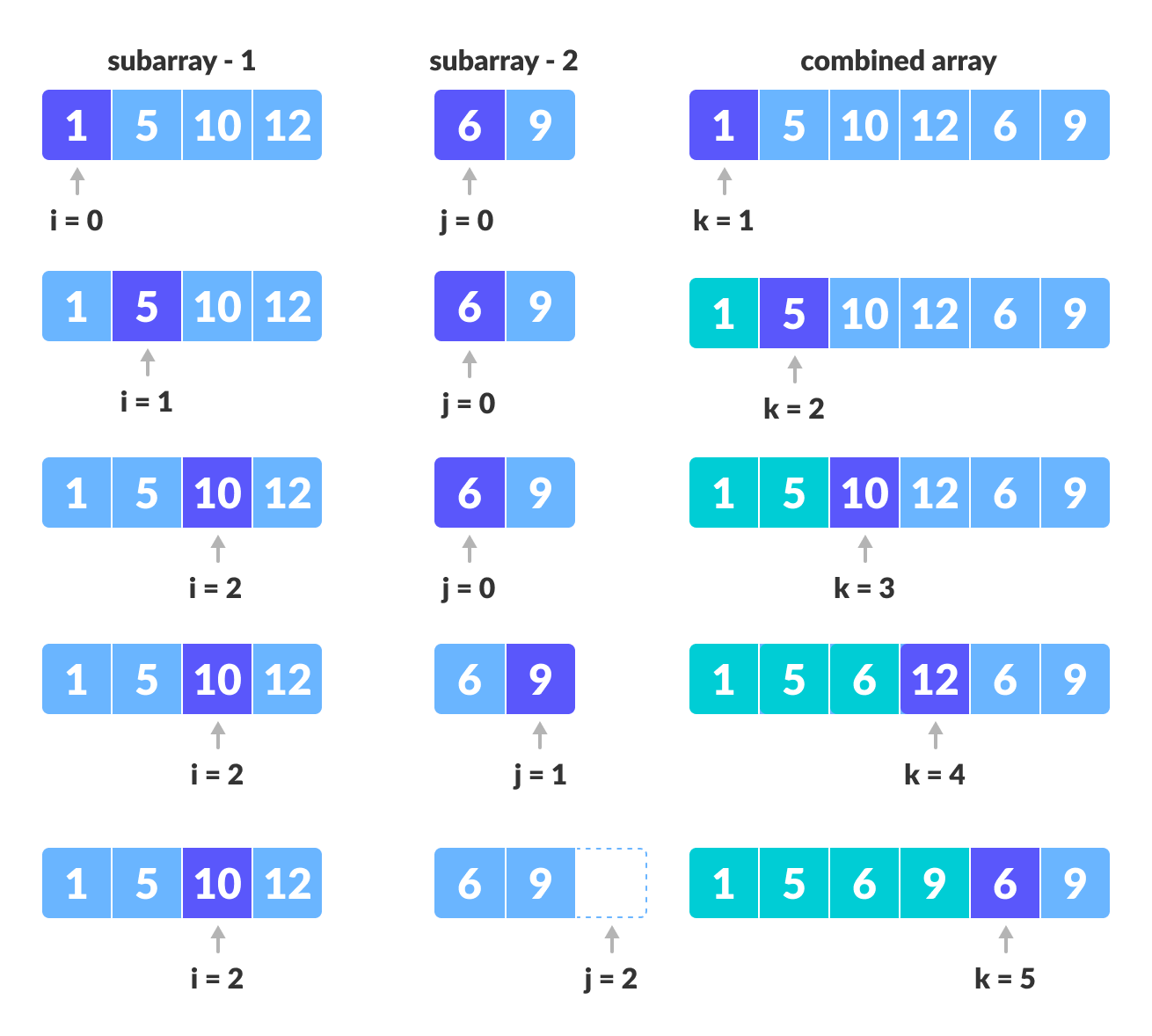
arr[k] = M[j];

j++;

}

k++;

}



**Bước 4: Khi chúng ta sử dụng hết các phần tử trong L hoặc M, hãy chọn các phần tử còn lại và đưa vào A [p..r]**

while (i < n1)

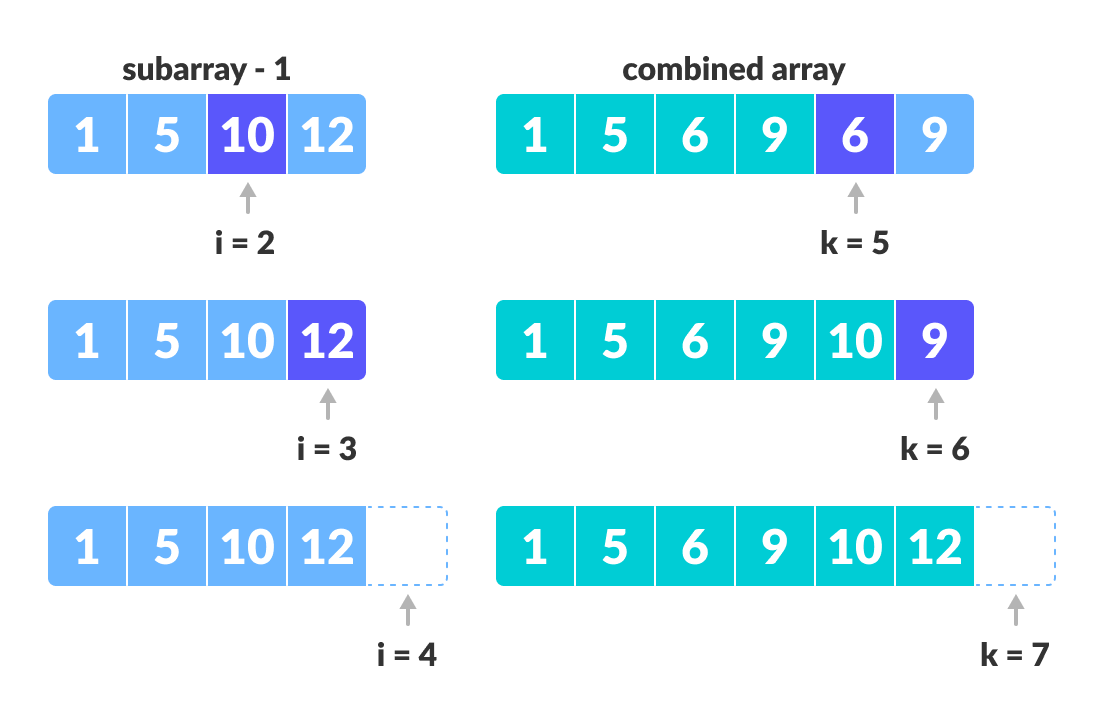
{

arr[k] = L[i];

i++;

k++;

}



while (j < n2)

{

arr[k] = M[j];

j++;

k++;

}

}

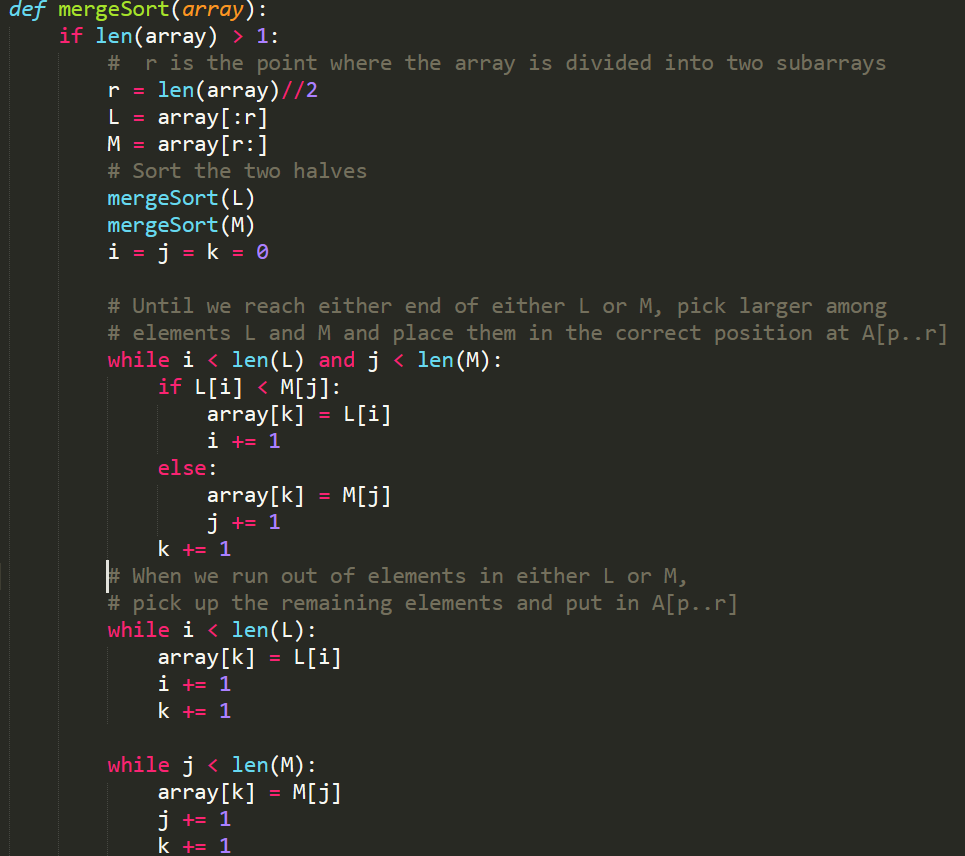


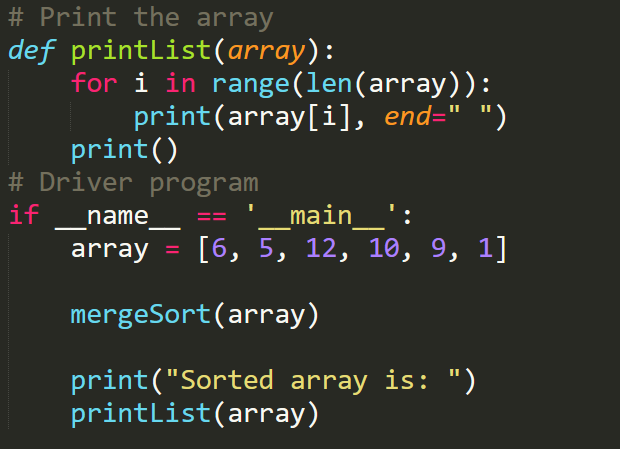
Bước này sẽ cần thiết nếu kích thước của M lớn hơn L.

Khi kết thúc hàm hợp nhất, mảng con A [p..r] được sắp xếp.

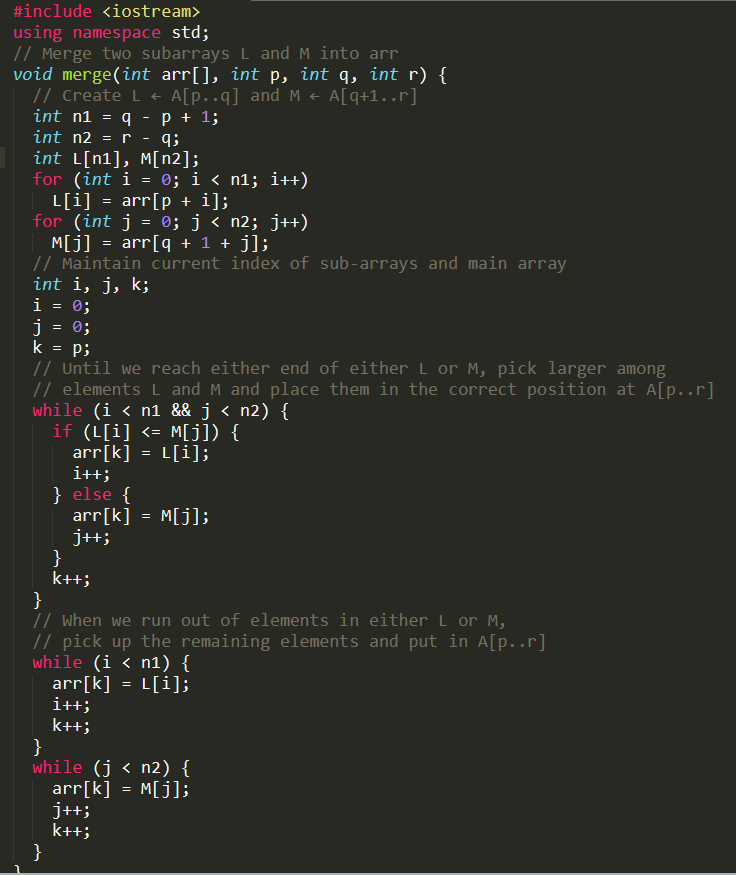
**CODE:**

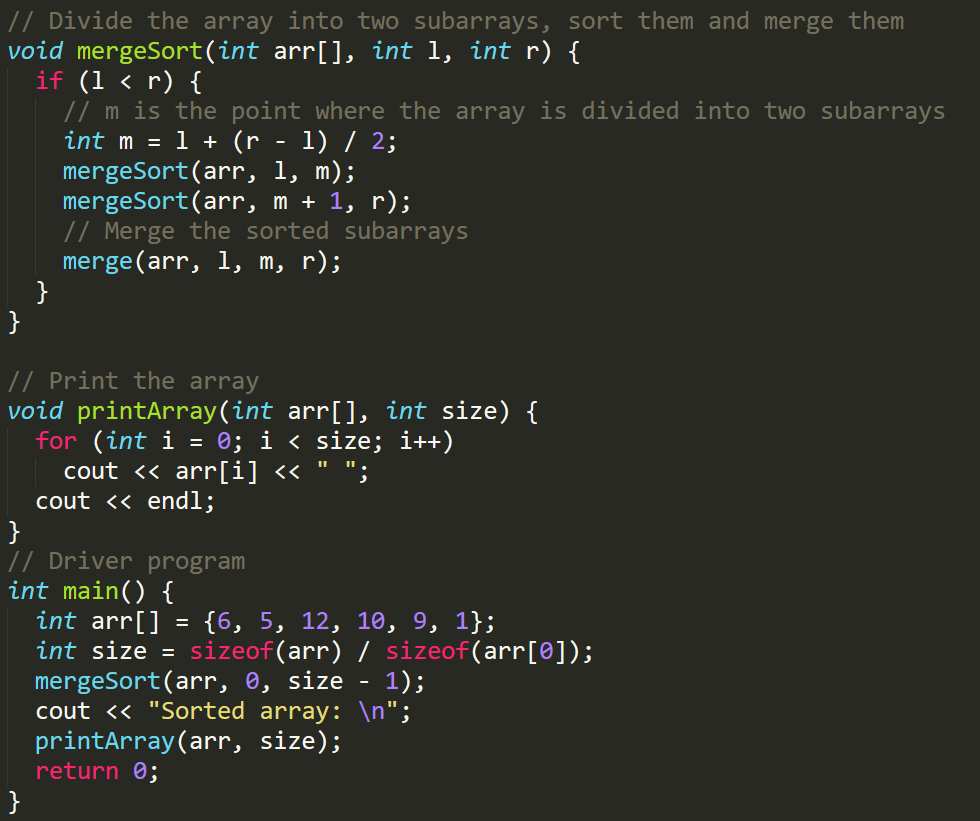
**\*PYTHON**





**\*C++:**





**Độ phức tạp thuật toán**

Trường hợp tốt: O(nlog(n))

Trung bình: O(nlog(n))

Trường hợp xấu: O(nlog(n))

Không gian bộ nhớ sử dụng: O(n)