Cấu trúc dữ liệu và giải thuật

TS. Phạm Tuấn Minh
Khoa Công nghệ Thông tin, Đại học Phenikaa
minh.phamtuan@phenikaa-uni.edu.vn
https://sites.google.com/site/phamtuanminh/

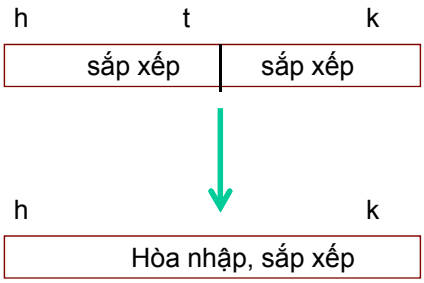
Chương 4: Sắp xếp

- Sắp xếp MergeSort

Hòa nhập hai dãy đã sắp xếp

- Sắp xếp b[h..k]
- Điều kiện: b[h..t] và b[t+1..k] đã sắp xếp

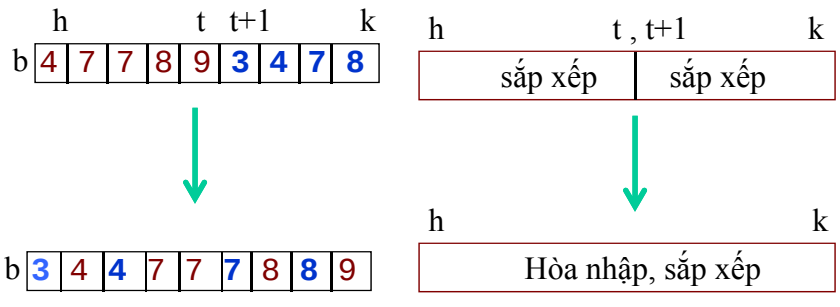
```
void merge(int b[], int h, int t, int k) {
    Chép b[h..t] vào mảng mới c;
    Hòa nhập c và b[t+1..k] thành b[h..k];
}
```



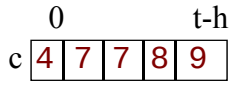
Hòa nhập hai dãy đã sắp xếp

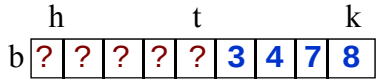
- Sắp xếp b[h..k]
- Điều kiện: b[h..t] và b[t+1..k] đã sắp xếp

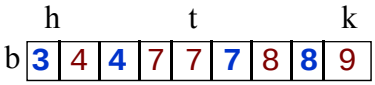
```
void merge(int b[], int h, int t, int k) {
}
```



Hòa nhập hai dãy đã sắp xếp

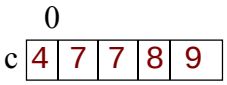


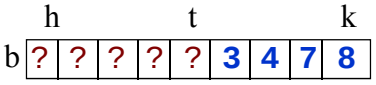




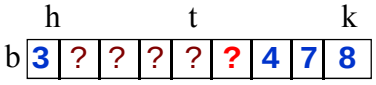
- Chép b[h..t] vào mảng mới c;
- Hòa nhập c và b[t+1..k] thành b[h..k]: Đặt các giá trị trong c[0..t] và b[t+1..k] vào b[h..k] theo thứ tự sắp xếp

Hòa nhập hai dãy đã sắp xếp

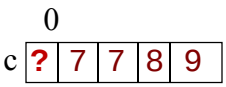


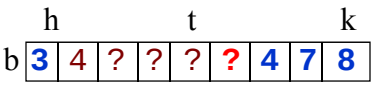


Bước 1. Chuyển 3 từ b[t+1] vào b[h]

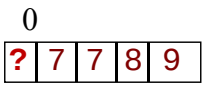


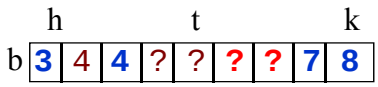
Bước 2. Chuyển 4 từ c[0] vào b[h+1]





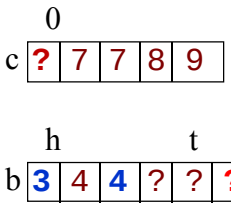
Bước 3. Chuyển 4 từ b[t+2] vào b[h+2]



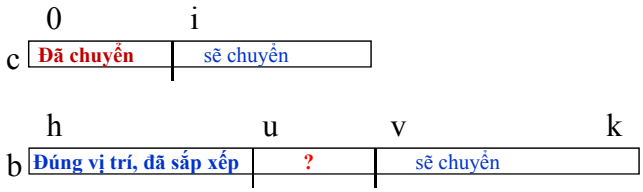


Hòa nhập hai dãy đã sắp xếp

Bước 3. Chuyển **4** từ b[t+2] vào b[h+2]



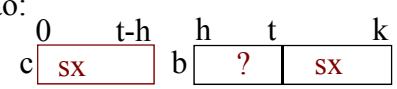
Ràng buộc:



Merge

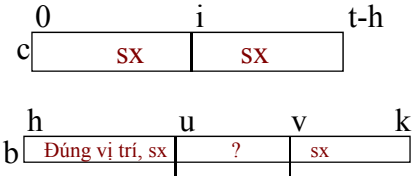
```
void Merge(int b[], int h, int t, int k) {
    i = 0;  u = h;  v = t+1;
    while (i <= t-h) {
        if (v <= k && b[v] < c[i]) {
            b[u] = b[v];
            u++; v++;
        } else {
            b[u] = c[i];
            u++; i++;
        }
    }

}
```

Vào:



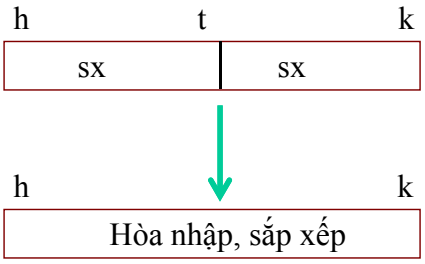
Kết quả: b

Ràng buộc:



MergeSort

```
/** Sort b[h..k] */
void MergeSort(int b[], int h, int k) {
   if (k – h < 1)  return;
   int t = (h+k) / 2;
   MergeSort(b, h, t);
   MergeSort(b, t+1, k);
   Merge(b, h, t, k);
}
```



So sánh QuickSort và MergeSort

```
/** Sort b[h..k] */
void QuickSort(int b[], int h, int k)
{
   if (k – h < 1) return;
   int j =  partition(b, h, k);
   QuickSort(b, h, j-1);
   QuickSort(b, j+1, k);
}
```

```
/** Sort b[h..k] */
void MergeSort(int b[], int h, int k) {
   if (k – h < 1) return;
   MergeSort(b, h, (h+k)/2);
   MergeSort(b, (h+k)/2 + 1, k);
   Merge(b, h, (h+k)/2, k);
}
```

QuickSort xử lý mảng (partition), rồi gọi đệ quy.
MergeSort gọi đệ quy, rồi xử lý mảng (merge).

Độ phức tạp

Algorithm	Ave time. Worst-case time	Space
Insertion sort	$O(n^2)$. $O(n^2)$	$O(1)$
Selection sort	$O(n^2)$. $O(n^2)$	$O(1)$
Quick sort	$O(n \log n)$. $O(n^2)$	O(log n)*
Merge sort	$O(n \log n)$. $O(n \log n)$	O(n)

11

Cấu trúc dữ liệu và giải thuật

- Nội dung bài giảng được biên soạn bởi TS. Phạm Tuấn Minh.

1-12

- 40, 66, 22, 55, 11, 88, 77

1-13