1. **Nêu bài toán:**

Cần sắp xếp mảng A[1…n]. Thuật toán sắp xếp trộn được phát triển dựa trên chia để trị, bao gồm các thao tác sau:

* Chia (Divede): Chia dãy gồm n phần tử cần sắp xếp ra thành 2 dãy, mỗi dãy có n/2 phần tử
* Trị (Conquer):
  + Sắp xếp mỗi dãy con một cách đệ quy sử dụng sắp xếp trộn.
  + Khi dãy chỉ còn một phần tử thì trả lại phần tử này.
* Tổ hợp (Combine): Trộn (Merge) hai dãy con được sắp xếp để thu được dãy được sắp xếp gồm tất cả các phần tử của cả hai dãy con.

1. **Mô tả chi tiết thuật toán:**

* Chia: Đầu tiên, giải thuật sắp xếp trộn chia toàn bộ mảng thành hai nửa. Tiến trình chia này tiếp tục diễn ra cho đến khi không còn chia được nữa
* Trộn:
  + So sánh hai phần tử đứng đầu của hai danh sách, lấy phần tử nhỏ hơn cho vào danh sách mới. Tiếp tục như vậy cho tới khi một trong hai danh sách là rỗng.
  + Khi một trong hai danh sách là rỗng ta lấy phần còn lại của danh sách kia cho vào cuối danh sách mới.
  + • Thuật toán sắp xếp trộn mergesort
  + • Input: a[l..r]
  + • Ouput: a[l..r] đã được sắp xếp
  + 1. if(l>=r) return ;
  + 2. t=(l+r)/2
  + 3. mergesort(l,t);
  + 4. mergesort(t+1,r);
  + 5. merge(a[l..t],a[t+1..r);

1. **Ví dụ**

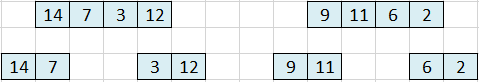
Ví dụ 1:

Mảng ban đầu:

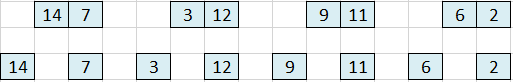


**Chia:**

* Chia mảng thành 2 nửa



* Tiến trình chia này tiếp tục diễn ra cho đến khi không còn chia được nữa



**Trộn:**

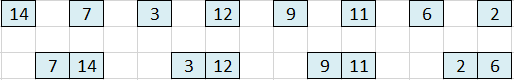
* So sánh hai phần tử đứng đầu của hai danh sách, lấy phần tử nhỏ hơn cho vào danh sách mới. Tiếp tục như vậy cho tới khi một trong hai danh sách là rỗng.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Danh sách trái | Danh sách phải | Danh sách mới |
| Ban đầu | 14 | 7 | Rỗng |
| Bước 1 | 14 | Rỗng | 7 |

* Khi một trong hai danh sách là rỗng ta lấy phần còn lại của danh sách kia cho vào cuối danh sách mới.

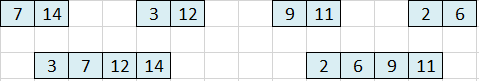
Danh sách mới : 7,14

Tương tự với các danh sách khác ta được kết quả:

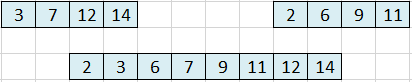


|  |  |  |  |
| --- | --- | --- | --- |
|  | Danh sách trái | Danh sách phải | Danh sách mới |
| Ban đầu | 7,14 | 3,12 | Rỗng |
| Bước 1 | 7,14 | 12 | 3 |
| Bước 2 | 14 | 12 | 3,7 |
| Bước 3 | 14 | Rỗng | 3,7,12 |
| Bước 4 | Rỗng | Rỗng | 3,7,12,14 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Danh sách trái | Danh sách phải | Danh sách mới |
| Ban đầu | 9,11 | 2,6 | Rỗng |
| Bước 1 | 9,11 | 6 | 2 |
| Bước 2 | 9,11 | Rỗng | 2,6 |
| Bước 3 | 11 | Rỗng | 2,6,9 |
| Bước 4 | Rỗng | Rỗng | 2,6,9,11 |



|  |  |  |  |
| --- | --- | --- | --- |
|  | Danh sách trái | Danh sách phải | Danh sách mới |
| Ban đầu | 3,7,12,14 | 2,6,9,11 | Rỗng |
| Bước 1 | 3,7,12,14 | 6,9,11 | 2 |
| Bước 2 | 7,12,14 | 6,9,11 | 2,3 |
| Bước 3 | 7,12,14 | 9,11 | 2,3,6 |
| Bước 4 | 12,14 | 9,11 | 2,3,6,7 |
| Bước 5 | 12,14 | 11 | 2,3,6,7,9 |
| Bước 6 | 12,14 | Rỗng | 2,3,6,7,9,11 |
| Bước 7 | 14 | Rỗng | 2,3,6,7,9,11,12 |
| Bước 8 | Rỗng | Rỗng | 2,3,6,7,9,11,12,14 |



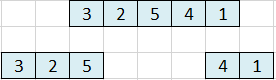
Ví dụ 2:

Mảng ban đầu:

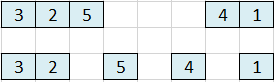


**Chia:**

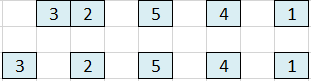
Bước 1:



Bước 2:

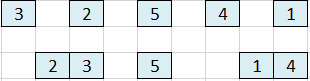


Bước 3:

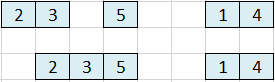


Trộn:

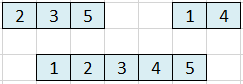
Bước 1:



Bước 2:



Bước 4:



-Bộ dữ liệu 2:

Mảng A gồm 12 phân tử

-23 34 65 3 -5 45 7 14 5 1 -10 -28

Kết quả sau khi sắp xếp:

-28 -23 -10 -5 1 3 5 7 14 34 45 65

Bộ dữ liệu 3:

Mảng A gồm 10 phân tử

5 6 9 0 -2 5 45 2 -4 12

Kết quả:

-4 -2 0 2 5 5 6 9 12 45

Bộ dữ liệu 4:

Mảng A gồm 11 phân tử

7 0 3 -6 -5 45 2 0 -5 1 3

-Kết quả sau khi sắp xếp:

-6 -5 -5 0 0 1 2 3 3 7 45

Bộ dữ liệu 5:

Mảng A gồm 11 phân tử

-3 5 7 4 8 -5 34 23 12 3 -2

-Kết quả sau khi sắp xếp:

-5 -3 -2 3 4 5 7 8 12 23 34

**4. viết chương trình**

// gop hai mang con arr[l...m] và arr[m+1..r]

void merge(int arr[], int l, int m, int r)

{ // l chi so left, r la right, m la middle

int i, j, k; // i la chi so bat dau cua mang dau tien,j la mang thu hai

int n1 = m - l + 1; // so pt mang ben trai

int n2 = r - m;// so pt mang ben pahi

// tao cac mang tam

int L[n1], R[n2];

// coppy dl sang mang tam

for (i = 0; i < n1; i++)

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

for (j = 0; j < n2; j++)

R[j] = arr[m + 1+ j];

// gom hai mang tam vao mang arr

i = 0; // khoi tao chi so bat dau cho mang dau tien

j = 0; //khoi tao chi so bat dau cho mang thu hai

k = l; // khoi tao chi so bat dau cho mang luu ket qua

while (i < n1 && j < n2)

{

if (L[i] <= R[j])

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[j];

j++;

}

k++;

}

// coppy cac pt con lai cua mang R vao arr neu co

while (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

// coppy cac pt con lai cua mang R vao arr neu co

while (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

}

// l la chi so trai,r la phai cua mang can dc sap xep

void mergeSort(int arr[], int l, int r)

{

if (l < r)

{

// Tuong tu (l+r)/2, nhung cách này tránh tràn so khi l và r lon

int m = l+(r-l)/2;

//goi de quy tiep tuc chia doi tung nua mang

mergeSort(arr, l, m);

mergeSort(arr, m+1, r);

merge(arr, l, m, r);

}

}

**5.Độ phức tạp:**

Gọi T(n) là thời gian sắp xếp mảng với n phần tử

Ta có T( n ) = t( n / 2 ) + t ( n / 2 ) + cn

Trong đó:

* t ( n / 2 ) là thời gian sắp xếp mảng với n/2 phần tử
* cn là thời gian sắp xếp lại mảng

T(n) = 2t(n/2) + cn

= 2( 2t(n/4) + c(n/2) ) + cn

= 22 \* t(n/4) + 2cn

= 22 \* ( 2t(n/8) + c(n/22) ) + 2cn

= 23 \* t(n/8) + 3cn

…

= 2k  \* t(n/2k) + kcn (1)

1. sẽ dừng lại khi t(1) ⬄ n = 2k ⬄ log2n = k

* T(n) = 2log2n t(1) + cn \* log2n

= O(n) + O(n \* log2n)

= O(n \* log2n)