**I. Giới thiệu thuật toán chia để trị**

**1.Ý tưởng**

Là một phương pháp được áp dụng rộng rãi

Ý tưởng chung là phân rã bài toán thành bài toán nhỏ hơn "độc lập" với nhau.

Giải các bài toán con theo cùng 1 cách thức

"Tổng hợp" lời các bài toán con để có được kết quả bài toán ban đầu.

Tư tưởng chung của cách tiếp cận Chia để trị

Chia:

Bằng cách nào đó chia tập hợp các đối tượng của bài toán thành bài toán con "độc lập"

Tiếp tục chia các bài toán con cho đến khi có thể giải trực tiếp (không cần, hoặc không thể chia nhỏ nữa)

Trị:

Trên các bài toán con thực hiện cùng một cách thức: Chia nhỏ nếu cần hoặc giải trực tiếp

Tổng hợp:

Khi mỗi bài toán con được giải, tổng hợp để có kết quả bài toán ban đầu.

**2. Mô hình**

Nếu gọi D&C(R) – Với R là miền dữ liệu là hàm thể hiện cách giải bài toán theo phương pháp chia để trị thì ta có thể viết:

Void D&&C(R)

{

If(R đủ nhỏ)

Giải bài toán;

Else

{

Chia R thành R1, R2,…. Rm

For(i=1;i<=m;i++)

D&C(R);

Tổng hợp kết quả bài toán

}

}

**II. Thuật toán Merge theo phương pháp chia để trị**

**1.Bài toán**

Phát biểu bài toán: Cho mảng gồm n phần tử A[1..n], sắp xếp mảng A theo thứ tự tăng dần.

**2.Ý tưởng**

Nếu có hai dãy a và b đã được sắp xếp, tiến hành trộn hai dãy này thành dãy c đã được sắp xếp.

Nếu chia nhỏ mảng cần sắp xếp thành các đoạn 1 phần tử thì nó là đoạn được sắp xếp

Tiến hành ghép các đoạn nhỏ thành các đoạn lớn đã

được sắp xếp

Duyệt trên dãy a tại vị trí i

Duyệt trên dãy b tại vị trí j

Nếu a[i]>b[j] thì thêm b[j] và trong dãy c tăng biến j ngược lại thêm a[i] vào dãy và tăng biến i

Nếu một trong hai dãy hết trước tiến hành đưa toàn bộ dãy còn lại vào trong dãy c

Áp dụng trong trường hợp a, b là hai đoạn của mảng

a[l..t], a[t+1..r]

c[l..r]

Để thuận tiện trong xử lý tiến hành chuyển mảng đã sắp xếp về mảng a

**3.Mô tả thuật toán**

Nếu có hai dãy a và b đã được sắp xếp, tiến hành trộn hai dãy này thành dãy c đã được sắp xếp.

Nếu chia nhỏ mảng cần sắp xếp thành các đoạn 1 phần tử thì nó là đoạn được sắp xếp

Tiến hành ghép các đoạn nhỏ thành các đoạn lớn đã được sắp xếp

4. while (i<=t && j<=r)

a. if(a[i]<a[j])

c[p]=a[i]

i++

b. Else

c[p]=a[j];

j++

c. p++

5. while (i<=t)

c[p]=a[i]

i++

p++

6. while (j<=r)

c[p]=a[j]

j++

p++

7. for (i=l; i<=r ;i++)

a[i]=c[i];

• 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);

**4. Cài đặt**

void merge(int a[100],int l,int r)

{

int c[100];

int t=(l+r)/2;

int i=l;

int j=t+1;

int p=l;

while(i<=t&&j<=r)

{

if(a[i]<a[j])

{

c[p]=a[i];

i++;

}else{

c[p]=a[j];

j++;

}p++;

}

while(i<=t)

{

c[p]=a[i];

i++;

p++;

}

while(j<=r)

{

c[p]=a[j];

j++;

p++;

}

for(int i=l;i<=r;i++)

{

a[i]=c[i];

}

}

int mergesort(int l, int r)

{

if(l>=r)

return 0;

int t=(l+r)/2;

printf("t=%d", t);

printf("\n");

mergesort(l,t);

mergesort(t+1,r);

merge(a,l,r);

}

**5. Thực hiện các bước của thuật toán**

Bộ dữ liệu 1:

Mảng A gồm 10 phân tử

8 4 -7 3 7 -9 3 5 -6 -8

Các bước thức hiện

Thuật toán merge



Thuật toán mergesort



Bộ dữ liệu 2:

Mảng A gồm 12 phân tử

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

Các bước thức hiện

Thuật toán merge



Thuật toán mergesort



Bộ dữ liệu 3:

Mảng A gồm 10 phân tử

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

Các bước thức hiện

Thuật toán merge



Thuật toán mergesort



Bộ dữ liệu 4:

Mảng A gồm 11 phân tử

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

Các bước thức hiện

Thuật toán merge



Thuật toán mergesort



Bộ dữ liệu 5:

Mảng A gồm 11 phân tử

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

Các bước thức hiện

Thuật toán merge



Thuật toán mergesort



**6. Đánh giá độ phức tạp thuật toán:**

• Đánh giá độ phức tạp

Bài làm

*T(n)=2T(n/2)+C2n*

*T(n)=2(2T(n/4)+C2.n/2)+2C2.n*

*T(n)=8T(n/4)+C3.n*

*T(n)=…*

*T(n)=2i\*T(n/2i)+iC2n*

*Khi kết thúc chương trình 2i =n khi đó i=k*

*Suy ra k=log2n*

*T(n)=2k\*T(1)+kC2n*

*Với T(1)=C1 và*

*T(n)=nC1+ log2n.C2n*

*T(n)= nC1+ logn.C2n/log2*

*Độ phức tạp của thuật toán O(n.logn)*