**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 Quicksort 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**

Cho một dãy, chọn một phần tử ở giữa, chia đoạn thành 2 phần

Chuyển các phần tử nhỏ, hoặc bằng đến trước, các phần tử lớn hơn về sau

Sẽ được nửa đầu bé hơn nửa sau

Lặp lại việc chuyển đổi cho các phần tử nửa đầu, và nửa sau đến lúc số phần tử là 1

Thuật toán ban đầu là chia: cố gắng chia thành hai đoạn khác nhau

Trị: thực hiện các thuật toán sắp xếp trên các đoạn con

Thực hiện kết hợp: thuật toán tự kết hợp kết quả

**3.Mô tả thuật toán**

• Thuật toán: partition

• Input: A[l..r], l,r: đoạn cần phân chia

• Ouput: A[l..r], i chỉ số phân chia

1. X=a[l]

2. i=l+1;

3. j=r;

4. While (i<j)

a. While (i<j && a[i]<x) i++

b. While (j>=i && a[j]>=x) j –

c. If(i<j) swap(a[i],a[j])

5. Swap(a[l],a[j])

6. Return j;

• Thuật toán: quicksort

• Input: A[l..r]: đoạn cần sắp xếp

• Ouput: A[l..r] đã sắp xếp

1. If(l>=r) return;

2. i=partition(A,l,r)

3. quicksort(A,l,i-1)

4. quicksort(A,i+1,r)

**4.Cài đặt**

void swap(int &a, int &b)

{

int x;

x=a;

a=b;

b=x;

}

int partition(int a[100], int l, int r)

{

int x=a[l];

int i=l+1;

int j=r;

while(i<=j)

{

while(i<=j&&a[i]<=x)i++;

while(j>i&&a[j]>x)j--;

if(i<j)swap(a[i],a[j]);

}swap(a[l],a[j]);

return j;

}

int quicksort(int a[100], int l, int r)

{

if(l>=r ) return 0 ;

int i=partition(a,l,r);

quicksort(a,l,i-1);

quicksort(a,i+1,r);

}

**5. Thực hiện các bước thuật toán**

Bộ dữ liệu 1:

Mảng A gồm 13 phần tử:



Các bước thực hiện:

Hàm partition:



Hàm Quicksort



Bộ dữ liệu 2:

Mảng A gồm 10 phần tử:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 10 | 9 | 3 | 2 | 8 | -1 | 6 | 7 | 20 |

Các bước thực hiện:

Hàm partition



Hàm Quicksort



Bộ dữ liệu 3:

Mảng A gồm 10 phần tử:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 9 | 39 | -23 | 7 | 90 | 5 | 7 | -3 | 4 |

Các bước thực hiện:

Hàm partition



Hàm quicksort:



Bộ dữ liệu 4:

Mảng A gồm 12 phần tử:



Các bước thực hiện:

Hàm partition



Hàm quicksort:



Bộ dữ liệu 5:

Mảng A gồm 11 phần tử:



Các bước thực hiện:

Hàm partition



Hàm quicksort:



**6. Đánh giá dộ phức tạp của thuật toán**

Đối với trường hợp tốt nhất

• Đánh giá độ phức tạp

Bài làm

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

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

*T(n)=8T(n/4)+3/2.Cn*

*T(n)=…*

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

*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)*

Đối với trường hợp xấu nhất

*Bài làm*

*T(n)=T(n-1)+C1+Cn*

*T(n)=T(n-2)+2C1+C(n+n-1)*

*T(n)=nC1+C(n+…+1)*

*T(n)=nC1+Cn.(n+1)/2*

*T(n)=C(n2/2+3/2.n)*

*Độ phức tạp thuật toán O(n2)*