Kỹ thuật lập trình

Programming Techniques

Ts. Nguyễn Đức Thuần
BM Hệ thống Thông Tin

Giới thiệu môn học

Nội dung môn hoc

- Chương 1: Kỹ thuật tổ chức chương trình
- Chương 2: Lập trình có cấu trúc – Hàm nâng cao
- Chương 3: Đệ qui
- Chương 4: Thử sai quay lui – Nhánh cận
- Chương 5: Chia để trị
- Chương 6:
- Chương 7:

CHIA ĐỂ TRỊ

(Divide & Conquer)

Chương 5: CHIA ĐỂ TRỊ

I. NỘI DUNG

- Giới thiệu
- Phương pháp
- Sơ đồ cài đặt
- Các ví dụ
- Ưu điểm và khuyết điểm

Chương 5: CHIA ĐỂ TRỊ

I. GIỚI THIỆU

Kỹ thuật này có thể chia làm 3 phần:

CHIA (Devide): Phân chia bài toán thành các bài toán nhỏ hơn

TRỊ (Conquer): Các bài toán con được gọi đệ qui đến bao giờ xử lý được.

TỔ HỢP (Combine): Tổ hợp kết quả của các bài toán con để có nghiệm của bài toán.

II. PHƯƠNG PHÁP

Chia để trị là 1 phương pháp áp dụng cho các bài toán có thể giải quyết bằng cách chia nhỏ ra thành các bài toán con từ việc giải quyết các bài toán con này. Sau đó lời giải của các bài toán nhỏ được tổng hợp lại thành lời giải cho bài toán ban đầu.

Chương 5: CHIA ĐỂ TRỊ

III. SƠ ĐỒ CÀI ĐẶT

```
void  chiadetri(A,x)
  { if (A đủ nhỏ) giải (A):x;
        else
            { Phân A thành A1,A2,..An
               for (i=1;i<=n;i++)
                     Chiadetri(Ai,xi);
               Tổ hợp (xi) ; return x;
             }
    }
```

CÁC BÀI TOÁN KINH ĐIỂN

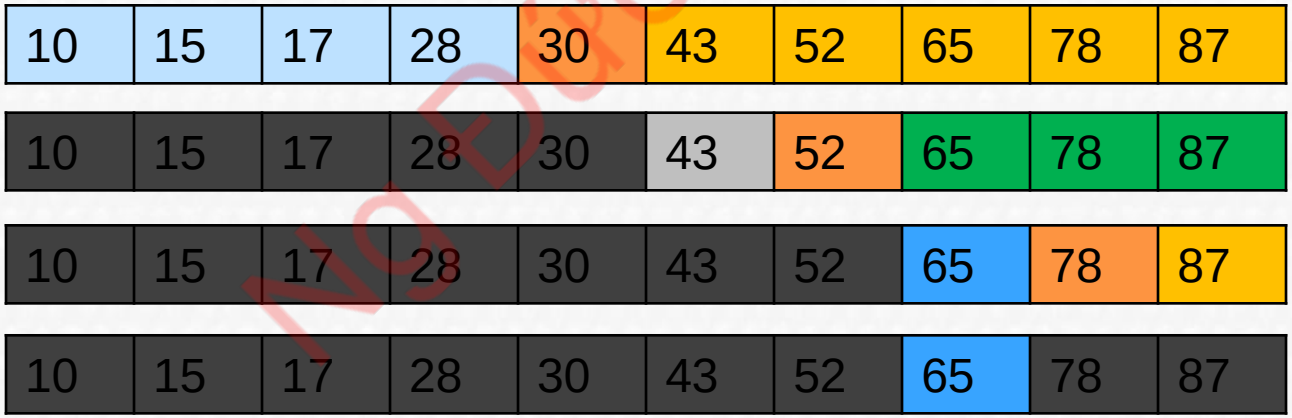
- MỘT SỐ VÍ DỤ

1. Tìm kiếm nhị phân: (Binary Search)

Input: Cho 1 dãy đã được sắp a[n], X là 1 giá trị tùy ý.

Output: Chỉ số phần tử thuộc dãy bằng X, nếu không có giá trị trả về -1.

Ví dụ: X= 65



Chương 5: CHIA ĐỂ TRỊ

MỘT SỐ VÍ DỤ

Tìm kiếm nhị phân: Cho 1 dãy đã được sắp x1, x2,.., xn và 1 số X. Tìm vị trí phần tử mạng giá trị X thuộc dãy, nếu không có trả về giá trị -1.

Input [ ]={5, 7, 10, 18, 20, 30, 38, 50};

X=30 Output:5; X=9 Output: -1

```
int binarySearch(int a[],int l,int r,int x)
{int mid;
   if (l>r) return -1;
   else
   {  mid=(l+r)/2;
       if (a[mid]==x) return mid;
       else
       if (a[mid]<x)
         return binarySearch(a,mid+1,r,x);
       else
        return binarySearch(a,l,mid-1,x);
                }}
```

2. Lũy thừa a^n: (Exponentiation)

Input: a,n

Output: a^n

Thuật toán:

$$\begin{cases} a^n = a^{n/2}.a^{n/2}, n \text{ chẵn} \\ a^n = a^{n/2}.a^{n/2}.a, n \text{ lẻ} \end{cases}$$

```
long lt(int a, int n)
  { long t;
    if (n==1) return a;
    else
      { t=lt(a,n/2);
        if (n%2==0) return t*t;
        else return t*t>a; }}
```

3. Dãy Fibonacci

Input: fibo(n) thỏa

$$\text{Fibo(n)} = \begin{cases} n, n \leq 2 \\ fibo(n-1) + fibo(n-2), n > 2 \end{cases}$$

Output: fibo(n)

Phương pháp giải:

- Đệ qui (công thức truy hồi)
- Chia để trị:

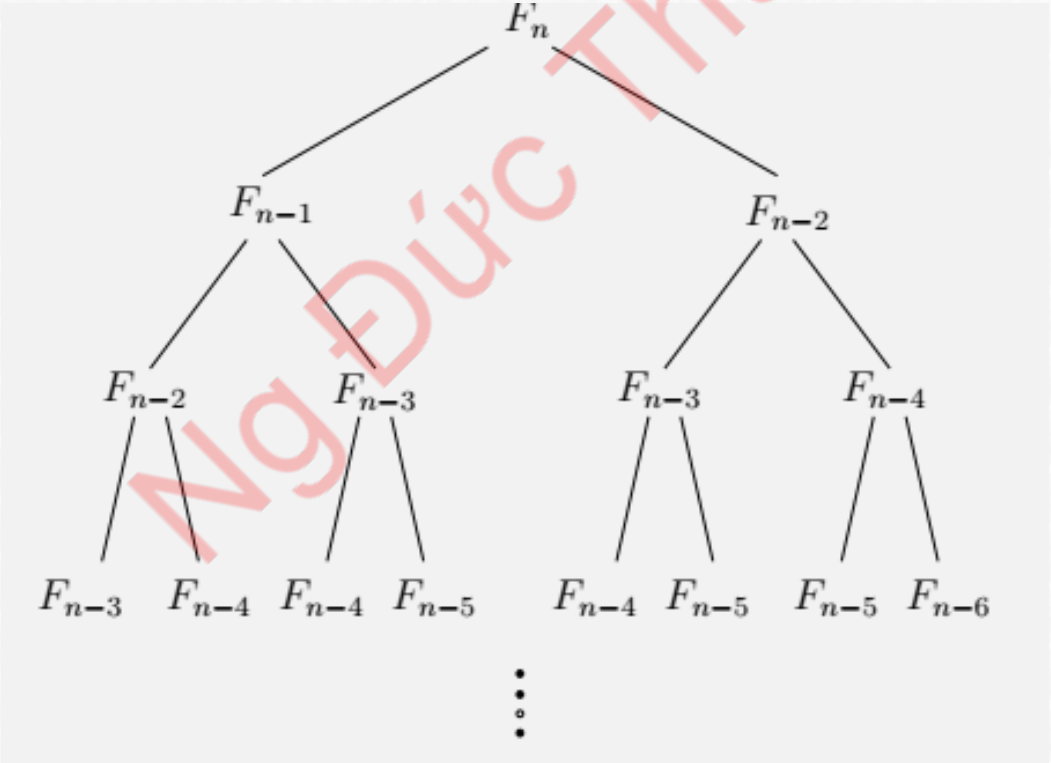
* Lặp

a. (sử dụng 3 biến a, b, c: {c=a+b; a=b; b=c}

b. Dùng mảng F[i]=f[i-1]+f[i-2]

3. Dãy Fibonacci

Chú ý: Phương pháp đệ qui theo công thức truy hồi, độ phức tạp thuật toán là hàm mũ, để thực hiện tính Fibo(n) có 2^n lời gọi hàm Fibo



3. Dãy Fibonacci

```
long Fi (int n)
{ int I;
  if (n<2) return n;
     else
       {f[0]=0; f[1]=1;
          for (i=2;i<=n;i++) f[i]=f[i-1]+f[i-2];
               return f[n];} }
```

0	1	2	3	..	i-2	i-1	i	..	n
0	1	2	3	..	f[i-2]	f[i-1]			

Chương 5: CHIA ĐỂ TRỊ

4. Sắp xếp Quick_sort

Input: cho dãy a[n]

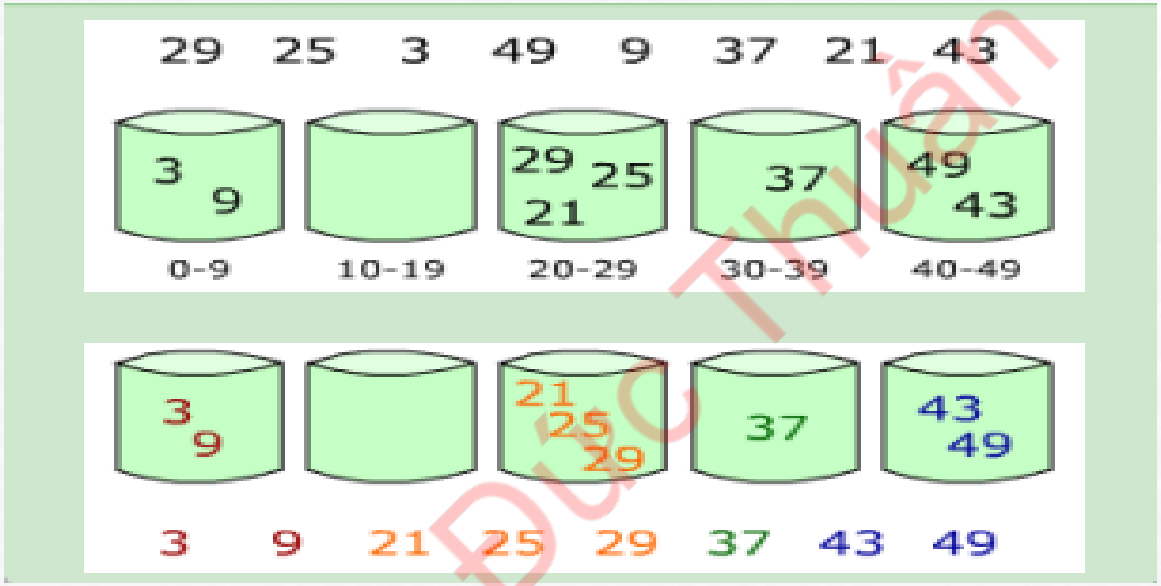
Output: a[n] là dãy được sắp theo thứ tự tăng dần.

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
20	75	42	25	19	50	40	13	80	15	35	45
20	35	42	25	19	50	40	13	80	15	75	45
20	35	15	25	19	50	40	13	80	42	75	45
20	35	15	25	19	13	40	50	80	42	75	45
13	35	15	25	19	20	40	50	45	42	75	80
13	15	35	25	19	20		42	45	50	75	
13	15	20	25	19	35				50	75	
		20	19	25	35						
		19	20	25	35						

4. Sắp xếp Quick_sort

```
 void quick_sort(int a[],int l,int r)
{ int k; int j;int m;int tam;
  int key;
    if (l<r)   { m=(l+r)/2; key=a[m];  k= l; j=r;
            while (k<j)
                {while (a[k]<key) k=k+1;
                 while (a[j]>key) j=j-1;
                       if (k<j) {tam=a[j]; a[j]=a[k];a[k]=tam;}
                        k=k+1;j=j-1;}
                          if (k<r) quick_sort(a,k,r);
                          if (j>l) quick_sort(a,l,j);
                         }}
```

5. Sắp xếp Bucket (Bucket Sort)

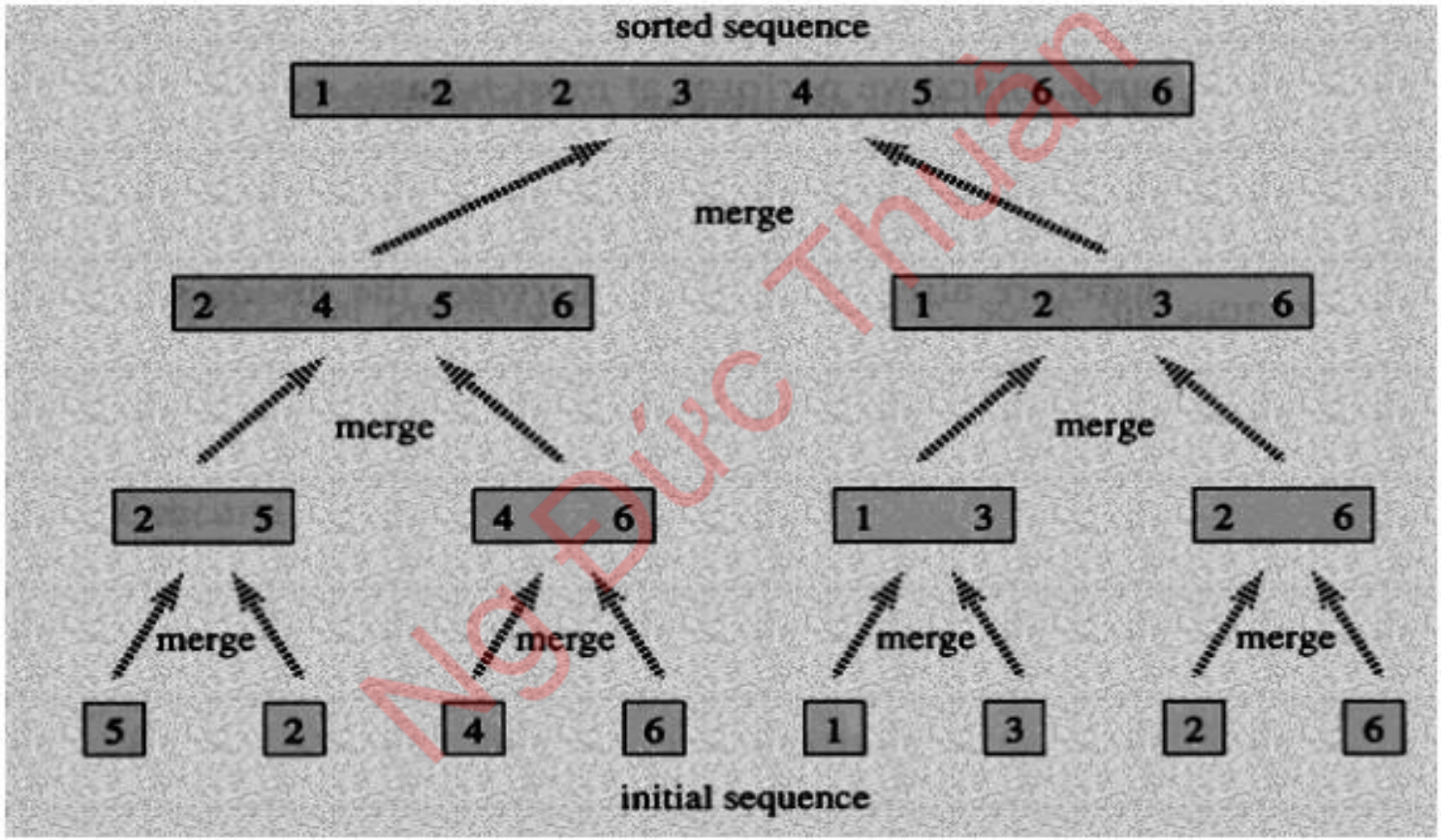


6. Sắp xếp phương pháp đếm (Counting Sort)

- Đây là trường hợp đặc biệt của thuật toán Bucket Sort, với độ rộng mỗi bucket là 1.

Chương 5: CHIA ĐỂ TRỊ

8. Sắp xếp trộn (Merge sort)



* Tìm hiểu và cài đặt 3 thuật toán 5, 6, 7

9. Tìm phần tử nhỏ thứ k (kth – smalleast element)

Input: Cho 1 dãy a[n] (chưa được sắp)

Output: Hiển thị phần tử nhỏ thứ k thuộc dãy – Độ phức tạp thuật toán O(n)

Method:

Có nhiều kỹ thuật (tham khảo geeksforgeeks). Ở đây sử dụng phương pháp Quickselect.

Ý tưởng:1. Chia dãy a[l,r] thành 2 dãy con với giá trị khóa K.

Dãy con a[l,j] và dãy a[k,r] (tương tự như Quick Sort)

2. Nếu (j-1)==(k-1) -> a[j] là phần tử cần tìm

Nếu j-1 > k tìm trong dãy con a[r,j-1]

Nếu j-1 < k tìm trong dãy con a[j-1,l]

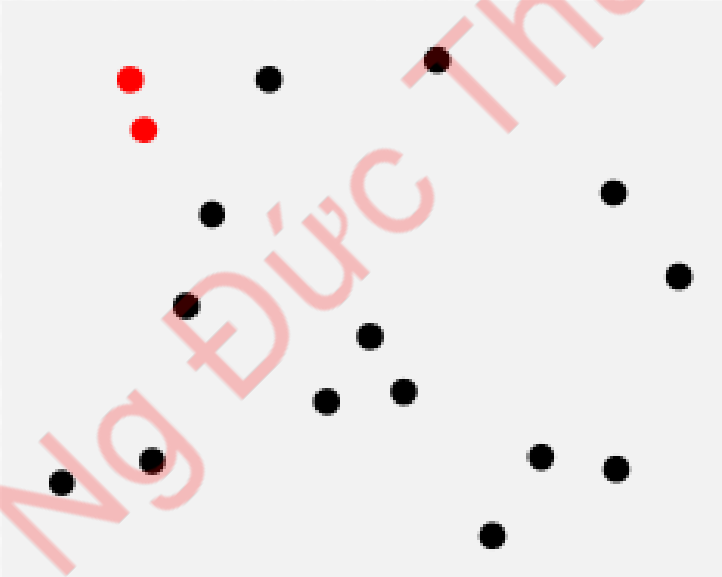
9. Tìm phần tử nhỏ thứ k (kth – smalleast element)

```
int Chiadoi(int a[],int l,int r)
{ int k; int j;int m;int tam;
  int key;
     if (l<r)
          { m=(l+r)/2;
            key=a[m];
          k= l; j=r;
while (k<j)
  {while (a[k]<key) k=k+1;
    while (a[j]>key) j=j-1;
    if (k<j) {tam=a[j];
a[j]=a[k];a[k]=tam;}
k=k+1;j=j-1;}
             return j;
             } }
```

```
int kthSmallest(int a[],int l,int r,int k)
{   int pos; // vi tri phan tu lon thu k
    if (k > 0 && k <= r - l + 1)  //
    {  pos = Chiadoi(a,l,r);
if (pos - l == k - 1)  return a[pos];
if (pos - l > k - 1)
   return kthSmallest(a, l, pos - 1, k);
return
kthSmallest(a, pos + 1, r, k - pos + l -1);
        // Neu khong co
        return -100000000;
}}
```

10. Tìm cặp điểm gần nhất (Finding closest pair of points)

Bài toán: Cho một tập hợp n điểm trong không gian d chiều, xác định hai điểm có khoảng cách Euclide nhỏ nhất.



Bài toán có nhiều ứng dụng trong: Kỹ thuật đồ họa, Thị giác (vision), điều khiển hàng không, chỉ đường,..

10. Tìm cặp điểm gần nhất (Finding closest pair of points)

- Chia (Divide): Xác định k sao cho đường thẳng x = k chia đều các điểm.
  (Tính toán trung vị, mất thời gian O (n).)
- Trị (Conquer): Thực hiện đệ qui: Tìm cặp gần nhất trong mỗi nửa.
- Tổ hợp (Combine): Có thể không cần xem xét các cặp gần nhất mà mỗi điểm nằm 1 nửa.
  Cần xem xét các cặp trên đường phân chia –
  (thời gian O (n^2))

Chương 5: CHIA ĐỂ TRỊ

10. Tìm cặp điểm gần nhất (Finding closest pair of points)

Nhận xét:

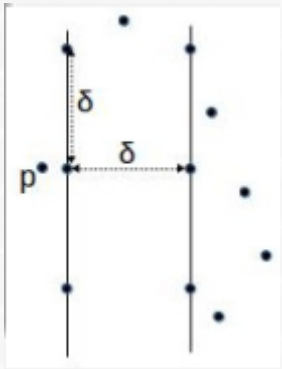
1. Đặt δ1 và δ2 là khoảng cách tối thiểu trong mỗi nửa.

- Chỉ cần xem xét các điểm trong phạm vi δ= min (δ1; δ2) từ đường thẳng phân cách 2 nửa.

(*Trong trường hợp xấu nhất có thể tất cả các điểm của 2 nửa nằm trong dải này*)

2. Xét một điểm p trên dải left bên trái. Có bao nhiêu điểm q1;..; qr trên dải right bên phải có thể nằm trong δ từ p?

- q1;..; qr tất cả phải nằm trong một hình chữ nhật 2δ × δ như hình
- *r có thể không quá lớn: q1; .. ; qr tạo thành 1 cụm gần hơn δ*
- Định lý: r ≤ 6



- Chỉ cần xem xét 6n cặp từ 2 nửa.

10. Tìm cặp điểm gần nhất (Finding closest pair of points)

Lời giải xem: https://www.geeksforgeeks.org/closest-pair-of-points-using-divide-and-conquer-algorithm/

(Cài đặt như bài tập)

11. Nhân 2 ma trận (Thuật toán Strassen's Matrix Multiplication)

Bài toán: Tìm ma trân X,Y với X, Y là 2 ma trận vuông nxn

Ý tưởng:

1. Biểu diễn X, Y thành:

- (Chia – Devide)

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \text{ and } Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

11. Nhân 2 ma trận (Matrix Multiplication)

2. Tính toán

- Trị (Conquer):

$$XY = \begin{bmatrix} P_6 + P_5 + P_4 - P_2 & P_1 + P_2 \\ P_3 + P_4 & P_1 - P_3 + P_5 - P_7 \end{bmatrix}$$

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$

$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$

$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$

$$P_4 = D(G - E)$$

- Cài đặt thuật toán như bài tập.

V. BÀI TẬP (bắt buộc)

1. Sàn trong mảng được sắp (floor in sort Array):

Cho 1 mảng đã được sắp và 1 giá trị X, sàn của X là phần tử lớn nhất bé hơn bằng X thuộc dãy.

Input: *arr[] = {1, 2, 3, 8, 10,12, 19}, X=5*
Output: *2*

Input: *arr[] = {1, 2, 3, 8, 10,12, 19}, X=20*
Output: *19*

Input: *arr[] = {1, 2, 3, 8, 10,12, 19}, X=0*
Output: *-1*

2. (Tìm số lượng số 0)

Cho 1 mảng chứa các ký số 0, 1. Các ký số 1 nằm đầu dãy, ký số 0 nằm cuối dãy. Đếm số lượng số 0 thuộc dãy.

Input: arr[] = {1, 1, 1, 1, 0, 0}	Output: 2
Input: arr[] = {1, 0, 0, 0, 0}	Output: 4
Input: arr[] = {0, 0, 0}	Output: 3
Input: arr[] = {1, 1, 1, 1}	Output: 0

3. Tìm điểm cố định (Fix Point)

Điểm cố định của mảng a[ ] là chỉ số i, sao cho a[i]=i;

Cho 1 mảng, tìm điểm cố định của mảng.

Input: arr[] = {-10, -5, 0, 3, 7} Output: 3 // arr[3] == 3

Input: arr[] = {0, 2, 5, 8, 17} Output: 0 // arr[0] == 0

Input: arr[] = {-10, -5, 3, 4, 7, 9} Output: -1 // No Fixed Point

Chương 5: CHIA ĐỂ TRỊ

V. BÀI TẬP LÀM THÊM (trong website GeeksforGeeks.org)

- Karatsuba algorithm for fast multiplication using Divide and Conquer algorithm
- Convex Hull using Divide and Conquer Algorithm
- Tiling Problem using Divide and Conquer algorithm
- The Skyline Problem using Divide and Conquer algorithm
- Maximum Subarray Sum using Divide and Conquer algorithm
- Longest Common Prefix using Divide and Conquer Algorithm
- Closest Pair of Points using Divide and Conquer algorithm
- Search in a Row-wise and Column-wise Sorted 2D Array using Divide and Conquer algorithm
- Dynamic Programming vs Divide-and-Conquer
- Maximum Sum SubArray using Divide and Conquer | Set 2
- Sum of maximum of all subarrays | Divide and Conquer
- Frequency of an integer in the given array using Divide and Conquer
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Merge K sorted arrays | Set 3 (Using Divide and Conquer Approach)

Cám ơn đã theo dõi