

**BÁO CÁO MÔN HỌC**

**PHÂN TÍCH VÀ THIẾT KẾ GIẢI THUẬT**

***Đề 5:***

***Bài toán tìm kiếm phần tử nhỏ thứ k và Select\_k***

*Giáo viên hướng dẫn: Nguyễn Mậu Uyên*

*Sinh viên thực hiện: Trịnh Viết Quốc*

*Lớp: KHMT 14*

Hà Nội, 12/2017

MỤC LỤC

[1. Bài toán 3](#_Toc501120361)

[2. Mô tả chi tiết thuật toán 3](#_Toc501120362)

[3. Ví dụ 4](#_Toc501120363)

[4. Viết chương trình bằng C 10](#_Toc501120364)

5. Đánh giá độ phức tạp thuật toán 11

# **Nêu Bài toán**

Khác với những bài toán tìm kiếm một phần tử có giá trị cụ thể trong mảng bài toán lần này đưa ra là tìm phần tử nhỏ thứ K nào đó trong một mảng, hoặc một danh sách. Thuật toán select\_k được xây dựng để giải quyết bài toán này.

# **Mô tả chi tiết thuật toán**

1. **Ý tưởng**

Có nhiều cách để giải quyết bài toán tìm phần tử nhỏ thứ k của mảng. tư duy thông thường, để tìm ra phần tử nhỏ thứ k của 1 mảng thì ta đi sắp xếp mảng đó theo thứ tự tăng dần, sau dó phần tử lớn thứ k sẽ là phần tử k-1 trong mảng sau khi đã sắp xếp.

Thuật toán select\_k không đi sắp xếp mảng trước, mà với một mảng đầu vào:

* Chọn 1 phần tử chốt(pivot)
* Chia mảng thành 2 phần, phần trái tất cả nhỏ hơn chốt còn lại ở phần bên phải
* Xét chỉ số index của phần tử chốt xem nếu index = k-1 thì array[index] là phần tử cần tìm, index < k-1 thì kết quả nằm ở phần bên phải mảng, còn index>k-1 thì kết quả phải ở bên trái mảng

1. **Mô tả chi tiết thuật toán**

Khi chia mảng bằng phần tử chốt(pivot), mà kết quả index của pivot khác k-1 thì ta sẽ phải gọi đệ quy chính hàm tìm kiếm đó một lần nữa, và pivot sẽ cũng thay đổi. Vì thế ta sẽ xây dựng một hàm để chia mảng ban đầu cũng như các mảng con thành 2 phần thông qua phần tử pivot

Với mảng arr[n]:

* Chọn arr[n-1] làm pivot
* Để index =0, sau khi kết thúc hàm chia mảng thì index = chỉ số của phần tử pivot
* Tăng index thêm 1 khi có 1 phần tử từ 0->length(arr)-1 nhỏ hơn pivot
* Đổi chỗ pivot với phần tử arr[index] khi đó chỉ số pivot bằng indext và mọi phần tử bên trái pivot đều nhỏ hơn nó
* Trường hợp đệ quy kết thúc là khi arr[] con chỉ gồm 2 phần tử kết quả sẽ là phần tử arr[length-1] như trong ý tưởng thuật toán

Từ đó xây dựng được hàm createPartition() để đưa ra index của pivot và Partition() để tìm ra phần tử nhỏ thứ k như sau:

int CreatePartition(int a[], int low, int high){

int pivot, index, i;

index = low;

pivot = high;

// Getting index of pivot.

for(i=low; i < high; i++){

if(a[i] < a[pivot]){

swap(&a[i], &a[index]);

index++;

}

}

// Swapping value at high and at the index obtained.

swap(&a[pivot], &a[index]);

return index;

}

//////////////////////

int Partition(int a[], int low, int high, int k){

int pindex;

if(low < high){

// Partitioning array using last element as a pivot.

// Recursively implementing partitioning in the direction to place the pivot at (k-1)th pivot.

pindex = CreatePartition(a, low, high);

if(pindex == k-1)

return k-1;

else if(pindex > k-1)

Partition(a, low, pindex-1, k);

else

Partition(a, pindex+1, high, k);

}

}

# **Đánh giá độ phức tạp của thuật toán**

Đối với Hàm **int createPartition(int a[], int low, int high){}**

Dễ thấy tồn tại 1 vòng for và các phép gán, so sánh => độ phức tạp O(n)

Hàm **int Partition(int a[], int low, int high, int k){}**

Gồm:

**Partition(a, low, pindex-1, k);**

**và**

**Partition(a, pindex+1, high, k);**

Khi đó T(n)= max(T1, T2) +O(n)

# **Ví dụ**

1. VD1: Tìm phần tử nhỏ thứ 4 của mảng 6 phần tử

**arr[6]= { 42, 68, 35, 1, 70, 25 }**

**kth= 4**

* Bước 1: Chọn chốt pivot = 25, index =0;

arr[]={1, 68, 35, 42, 70, 25}

index tăng 1, đổi chỗ pivot với arr[index]

kêt thúc 1:

index=1

arr[]={1, 25, 35, 42, 70, 68}

* Bước 2: index=1 < kth-1 nên phần tử cần tìm năm trong phần arr[index] đến arr[length]-1

arr[]={1, 25, 35, 42, 70, 68} pivot= 68, index =2

35, 42< pivot nên index tăng => index = 4

Đổi chỗ pivot với arr[index]

Kết thúc 2:

index = 4

arr[]={1, 25, 35, 42, 68, 70}

* Bước 3: index=4 > kth-1 nên phần tử cần tìm nằm trong arr[2] đếm arr[index-1]

arr[]={1, 25, 35, 42, 68, 70} pivot= 42, index=2

35<42 nên index tăng 1 => index=3

Đổi chỗ pivot với arr[index] tức là giữ nguyên vị trí arr[3]

Kết thúc 3:

index = 3

arr[]={1, 25, 35, 42, 68, 70}

index = kth-1 => **arr[index]= arr[3]= 42 là phần tử nhỏ thứ 4 của mảng**

1. VD2: Tìm phần tử nhỏ thứ 3 của mảng 9 phần tử

**arr[6]= { 42, 68, 35, 1, 70, 25, 79, 59, 63}**

**kth=3**

* Bước 1: Chọn chốt pivot = 63, index =0;

arr[]={42, 68, 35, 1, 70, 25, 79, 59, 63}

42, 35, 1, 25, 59 < pivot =>

index tăng 5, đổi chỗ pivot với arr[index]

kêt thúc 1:

index=5

arr[]={42, 35, 1, 25, 59, 63, 79, 70, 68 }

* Bước 2: index=1 > kth-1 nên phần tử cần tìm năm trong phần arr[0] đến arr[index-1] tức là a[4]

arr[]={42, 35, 1, 25, 59, 63, 79, 70, 68 } pivot= 59, index =0

42, 35, 1, 25< pivot nên index tăng 4=> index = 4

arr[] không có gì thay đổi

Kết thúc 2:

index = 4

arr[]={42, 35, 1, 25, 59, 63, 79, 70, 68 }

* Bước 3: index=4 > kth-1 nên phần tử cần tìm nằm trong arr[0] đếm arr[index-1] tức là arr[3]

arr[]={42, 35, 1, 25, 59, 63, 79, 70, 68 } pivot= 25, index= 0

1<25 => index tăng 1

Đổi chỗ pivot với arr[index]

Kết thúc 3:

index = 1

arr[]={1, 25, 42, 35, 59, 63, 79, 70, 68 }

* Bước 4: index=1< kth -1 nên phần tử cần tìm nằm trong khoảng arr[2] đến arr[3]

arr[]={1, 25, 42, 35, 59, 63, 79, 70, 68 }pivot= 35, index= 2

Đổi chỗ pivot với arr[index]

Kết thúc 4:

index = 2

arr[]={1, 25, 35, 42, 59, 63, 79, 70, 68 }

index = kth-1 => **arr[index]= arr[2]= 35 là phần tử nhỏ thứ 3 của mảng**

# **5 .Thực hiện chương trình trên C và kết quả**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#define MAX 100

#define MIN 1

FILE \*f;

void swap(int\*, int \*);

void inItNumber(int\*, int\*);

void creatFile(char\*, int\*, int);

void inFile(char\*, int\*, int\*);

void out(int\*, int);

int CreatePartition(int[], int, int);

int Partition(int[], int, int, int);

int main(){

int a[100000],n,left,right,kth, result;

printf("Size of list that you want: ");

scanf("%d",&n);

inItNumber(a,&n);

creatFile("input.txt",a,n);

inFile("input.txt",a,&n);

printf("\nSuccess!\n\nData:\n");

out(a,n);

printf("\nEnter the Kth smallest element that you want: ");

scanf("%d",&kth);

result= Partition(a,0,n-1,kth);

printf("\nThe %dth smallest element is: %d",kth,a[result]);

getch();

}

void inItNumber(int \*a, int \*n){

for(int i=0; i<\*n; i++){

a[i]= MIN + rand()%(MAX- MIN +1);

}

}

void creatFile(char \*fname, int \*a, int n){

f= fopen(fname, "w");

fprintf(f,"%d\n",n);

for(int i=0; i<n; i++){

fprintf(f,"%d ",a[i]);

}

fclose(f);

}

void inFile(char \*fname, int \*a, int \*n){

f=fopen(fname,"r");

if(f==NULL)

printf("Khong tim thay file!");

else{

fscanf(f,"%d",n);

for(int i=0;i< \*n; i++)

fscanf(f,"%d", &a[i]);

}

fclose(f);

}

void out( int \*a, int n){

for(int i= 0; i<n; i++) {

printf("%d ",a[i]);

}

}

void swap(int \*a,int \*b){

int temp=\*a;

\*a=\*b;

\*b=temp;

}

// Partitioning the array on the basis of values at high as pivot value.

int CreatePartition(int a[], int low, int high){

int pivot, index, i;

index = low;

pivot = high;

// Getting index of pivot.

for(i=low; i < high; i++){

if(a[i] < a[pivot]){

swap(&a[i], &a[index]);

index++;

}

}

// Swapping value at high and at the index obtained.

swap(&a[pivot], &a[index]);

return index;

}

// Implementing Partition.

int Partition(int a[], int low, int high, int k){

int pindex;

if(low < high){

// Partitioning array using last element as a pivot.

// Recursively implementing partitioning in the direction to place the pivot at (k-1)th pivot.

pindex = CreatePartition(a, low, high);

if(pindex == k-1)

return k-1;

else if(pindex > k-1)

Partition(a, low, pindex-1, k);

else

Partition(a, pindex+1, high, k);

}

**Kết quả thực hiện chương trình:**



