HỌC VIỆN KỸ THUẬT QUÂN SỰ

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÀI TẬP LỚN**

**PHÂN TÍCH VÀ THIẾT KẾ GIẢI THUẬT**

**Bài 8: Tìm Kiếm Các Thành Phần Chính (Majority Element Problem)**

Giáo viên hướng dẫn: Nguyễn Mậu Uyên

Sinh viên thực hiện: Hoàng Thị Hằng

Lớp: MMT14

MSV: 15150126

**Majority Element Problem**

1. **Bài toán**

Bài toán tìm thành phần chính trong mảng là một bài toán xác định có hay không một phần tử trong mảng xuất hiện nhiều lần hơn so với tất cả các phần tử khác trong mảng kết hợp lại. Nếu tồn tại phần tử này thì đưa ra nó là phần tử nào, bằng không thông báo không có thành phần chính trong mảng.

VD: Cho một mảng a[], với n số phần tử trong mảng .Viết một hàm từ mảng đã cho và kết quả in ra là thành phần chính (nếu tồn tại). Nếu không tìm được thành phần chính thì in ra kết quả là NONE

* I / P: 3 3 4 2 4 4 2 4 4

O / P: 4

* I / P: 3 3 4 2 4 4 2 4

O / P: NONE

1. **Mô tả chi tiết thuật toán:**

**Ý tưởng:** Thuật toán sử dụng là ***thuật toán bầu cử của Moore*** ( Moore Voting Algorithm).

Thuật toán này xử lý bài toán theo hai bước:

* Bước 1: Đưa ra phần tử có khả năng là thành phần chính trong mảng. Nếu có một thành phần chính trong mảng, bước sau đó sẽ đưa ra thành phần chính đó, bằng không nó sẽ trả về một thành phần bất kì nào khác.
* Bước 2: Kiểm tra phần tử được đưa ra ở trên có phải là thành phần chính không. Bước này là cần thiết bởi chúng ta thường không chắc chắn phần tử đưa ra ở bước trên là thành phần chính của mảng.

1. Tìm một phần tử có khả năng là thành phần chính

Ta sử dụng thuật toán bầu cử của Moore để thực hiện công việc này, nó có độ phức tạp là O(n). Ý tưởng của thuật toán là nếu ta có thể giảm số lần xuất hiện các của một phần tử *e* đã duyệt trước đó so với các phần tử khác *e* cho đến cuối cùng của mảng mà vẫn tồn tại số lượng phần tử *e* thì *e* là thành phần chính.

|  |
| --- |
| findCandidate(a[], n)  1. Initialize index and count of majority element  maj\_index = 0, count = 1  2. Loop for i = 1 to n – 1  (a) If a[maj\_index] == a[i]  count++  (b) Else  count--;  (c) If count == 0  maj\_index = i;  count = 1  3. Return a[maj\_index] |

Trong thuật toán trên, ta sẽ lặp qua mỗi phần tử và duy trì một bộ đếm cho phần tử chính a[maj\_index]. Nếu phần tử tiếp theo mà giống phần tử trước đó thì tăng bộ đếm, ngược lại sẽ giảm bộ đếm và nếu giảm đến khi bằng 0 thì thay đổi maj\_index bằng phần tử hiện tại và cho bộ đếm count=1.

Trong pha đầu tiên, thuật toán đưa ra phần tử có khả năng là thành phần chính. Trong pha thứ hai, ta phải kiểm tra xem nó có phải là thành phần chính hay không. Ta chỉ cần kiểm tra xem nếu số lần xuất hiện của ứng cử viên cho thành phần chính lớn hơn n/2

1. Kiểm tra xem có phải phần tử đó là thành phần chính

|  |
| --- |
| printMajority (a[], n)  1. Find the candidate for majority  2. If candidate is majority. i.e., appears more than n/2 times.  Print the candidate  3. Else  Print "NONE" |

1. **Độ phức tạp của thuật toán:**

* Thuật toán có tốc độ tăng tuyến tính với độ phức tạp O(n)

int maj\_index = 0, count = 1;

int i;

for (i = 1; i < n; i++) (1)

{

if (a[maj\_index] == a[i]) (2)

count++; (3)

else

count--; (4)

if (count == 0) (5)

{

maj\_index = i; (6)

count = 1; (7)

}

}

return a[maj\_index];

* Toàn bộ chương trình chỉ có 1 vòng lặp for (1), lồng trong vòng lặp for (1) là các lệnh (2), (3), (4), (5), (6), (7).Tiến hành tính độ phức tạp theo thứ tự từ trong ra ngoài.
* Lệnh (3), (4), (5), (6), (7) đều tốn O(1) thời gian, việc so sánh a[maj\_index] == a[i] cũng tốn O(1) thời gian, suy ra lệnh (2) tốn O(1) thời gian.
* Vòng lặp for (1) thực hiện n lần , mỗi lần tốn O(1) do đó tốn O(n.1) = O(n). Vòng lặp (1) với i chạy từ 1 đến n nên thời gian thực hiện của vòng lặp (1) và cũng là độ phức tạp của giải thuật là:

T(n) =

Vậy độ phức tạp của thuật toán là O(n)

1. **Thực hiện thuật toán**

***Bộ dữ liệu 1:***

Cho A [] ={2, 2, 3, 5, 2, 2, 6 }

n =7

* Tìm thành phần chính

+/ Khởi tạo:

maj\_index = 0, count = 1 -> ứng cử viên cho thành phần chính là 2

2, 2, 3, 5, 2, 2, 6

+/Tương tự như a[maj\_index] => count = 2

2, 2, 3, 5, 2, 2, 6

+/ Khác với a[maj\_index] => count = 1

2, 2, 3, 5, 2, 2, 6

+/ Khác với a[maj\_index] => count = 0

Vì count = 0, thay đổi ứng cử viên cho thành phần chính thành 5

=> maj\_index = 3, count = 1

2, 2, 3, 5, 2, 2, 6

+/ Khác với a[maj\_index] => count = 0

Vì count = 0, thay đổi ứng cử viên cho thành phần chính thành 2

=> maj\_index = 4

2, 2, 3, 5, 2, 2, 6

+/ Tương tự như a[maj\_index] => count = 2

2, 2, 3, 5, 2, 2, 6

+/ Khác với a[maj\_index] => count = 1

Cuối cùng ứng cử viên cho thành phần chính là 2.

Với số lượng phần tử có trong mảng là 7 và số lần xuất hiện của ứng cử viên cho thành phần chính là 4 lần.Vậy kết quả trả về thành phần chính là 2

***Bộ dữ liệu 2:***

Cho A[] = {1, 3, 1, 1, 2, 4, 6, 7, 3, 2}

n = 10

* Tìm thành phần chính.

Khởi tạo:

+/maj\_index = 0, count = 1 -> ứng cử viên cho thành phần chính là 1

1, 3, 1, 1, 2, 4, 6, 7, 3, 2

+/3 khác với 1 => count = 0

Vì count = 0, thay đổi ứng cử viên cho thành phần chính thành 3

maj\_index = 2, count = 1

1, 3, 1, 1, 2, 4, 6, 7, 3, 2

+/ 1 khác với 3 => count = 0

Vì count = 0, thay đổi ứng cử viên cho thành phần chính thành 1

maj\_index = 3, count = 1

1, 3, 1, 1, 2, 4, 6, 7, 3, 2

+/ 1 và 1 giống nhau => count = 2

1, 3, 1, 1, 2, 4, 6, 7, 3, 2

+/ 2 khác 1 => count = 1

1, 3, 1, 1, 2, 4, 6, 7, 3, 2

+/ 4 khác 1 => count = 0

Vì count = 0, thay đổi ứng cử viên cho thành phần chính thành 4

maj\_index = 6, count = 1

1, 3, 1, 1, 2, 4, 6, 7, 3, 2

+/ 6 khác 4 => count = 0

Vì count = 0, thay đổi ứng cử viên cho thành phần chính thành 6

maj\_index = 7, count = 1

1, 3, 1, 1, 2, 4, 6, 7, 3, 2

+/7 khác 6 => count = 0

Vì count = 0, thay đổi ứng cử viên cho thành phần chính thành 7

maj\_index = 8, count = 1

1, 3, 1, 1, 2, 4, 6, 7, 3, 2

+/ 3 khác 7 => count = 0

Vì count = 0, thay đổi ứng cử viên cho thành phần chính thành 3

maj\_index = 9, count = 1

1, 3, 1, 1, 2, 4, 6, 7, 3, 2

+/ 2 khác 3 => count = 0

Vì count = 0, thay đổi ứng cử viên cho thành phần chính thành 2

maj\_index = 10, count = 1

Cuối cùng ứng cử viên cho thành phần chính là 2

Với số lượng phần tử có trong mảng là 10, số lần xuất hiện của ứng cử viên cho thành phần chính là 2 lần => không thỏa mãn điều kiện , vậy kết quả trả về là không tìm được thành phần chính .

1. **Cài đặt**

#include<stdio.h>

#include <stdlib.h>

#define bool int

const char FILENAME[] = "Majority\_element.txt";

int findCandidate(int \*, int);

bool isMajority(int \*, int, int);

void printMajority(int a[], int n) // ham in ra ket qua thanh phan chinh or khong co thanh phan chinh

{

/\* Find the candidate for Majority\*/

int cand = findCandidate(a, n); // khai bao bien cand va gan no bang phan tu duoc coi la ung cu vien cho thanh phan chinh

/\* Print the candidate if it is Majority\*/

if (isMajority(a, n, cand))

printf("Majority Element: %d\n", cand);

else

printf("No Majority Element\n");

}

/\* Function to find the candidate for Majority \*/

int findCandidate(int a[], int n) // Buoc 1: ham tim ung cu vien cho thanh phan chinh

{

int maj\_index = 0, count = 1;

int i;

for (i = 1; i < n; i++)

{

if (a[maj\_index] == a[i])

count++;

else

count--;

if (count == 0)

{

maj\_index = i;

count = 1;

}

}

return a[maj\_index];

}

bool isMajority(int a[], int n, int cand) // Buoc 2: kiem tra xem ung cu vien co du dieu kien la thanh phan chinh hay khong

{

int i, count = 0;

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

if (a[i] == cand)

count++;

if (count > n/2)

return 1;

else

return 0;

}

void ReadFile(int a[],const char filename[],int &i) // doc file

{

i=0;

int u;

FILE \*f=fopen(filename,"r"); // mo file

if(f==NULL)printf("\nKhong mo duoc file ");

else

{

while(!feof(f)) // neu chua den khoang trong

{

fscanf(f,"%d",&u); // lay gia tro bien u cho vao mang a

a[i]=u;

i+=1; // tang bien i them 1

}

printMajority(a, i-1); // in ra gia tri

}

fclose(f);

}

/\* Ham main \*/

int main()

{

int a[50],n;

//int size = (sizeof(a))/sizeof(a[0]);

ReadFile(a,FILENAME,n);

//printMajority(a, n-1);

getchar();

return 0;

}