

Môn thi: Cấu trúc dữ liệu và giải thuật
Mã lớp: IT003 – Hệ đại trà, chất lượng cao
Thời gian làm bài: 90 phút
(Sinh viên không được sử dụng tài liệu)

Câu 1: (1.5 điểm)

- Hãy cho biết độ phức tạp của thuật toán Selection Sort theo định nghĩa Big-O (O lớn) (0.25 điểm)
- Viết hàm sắp xếp mảng 1 chiều giảm dần với thuật toán Selection Sort (0.5 điểm)
- Chạy từng bước thuật toán đã viết ở trên với dãy số sau: 5, 8, 9, 10, 3, 6 (0.75 điểm)

Đáp án tham khảo:

- Độ phức tạp của thuật toán Selection Sort theo định nghĩa Big-O cả ba trường hợp tốt nhất, trung bình, xấu nhất là: $O(n^2)$.
- Thuật toán Selection Sort: Tại mỗi giá trị i , tìm giá trị lớn nhất $arr[max_index]$ trong đoạn $[i + 1, n - 1]$ và hoán vị (swap) $arr[max_index]$ với $arr[i]$.

```
void SelectionSort(int arr[], int n) {  
    for (int i = 0; i < n - 1; i++) {  
        max_index = i;  
        for (int j = i + 1; j < n - 1; j++) {  
            if (arr[j] > arr[max_index])  
                max_index = j;  
        }  
        swap(arr[max_index], arr[i]);  
    }  
}
```

- Mảng cần sắp xếp gồm $n = 6$ phần tử:

5	8	9	10	3	6
---	---	---	----	---	---

Bước 1: $i = 0$: khởi tạo $max_index = i = 0 \rightarrow arr[max_index] = arr[0] = 5$. Tìm giá trị $arr[j]$ lớn nhất và lớn hơn $arr[max_index]$ trong đoạn $[1, 5]$
 $\rightarrow arr[j]$ nằm ở vị trí $max_index = 3 \rightarrow$ hoán vị $arr[3]$ với $arr[0]$

10	8	9	5	3	6
----	---	---	---	---	---

Bước 2: $i = 1$: khởi tạo $max_index = i = 1 \rightarrow arr[max_index] = arr[1] = 8$. Tìm giá trị $arr[j]$ lớn nhất và lớn hơn $arr[max_index]$ trong đoạn $[2, 5]$
 $\rightarrow arr[j]$ nằm ở vị trí $max_index = 2 \rightarrow$ hoán vị $arr[2]$ với $arr[1]$

10	9	8	5	3	6
----	---	---	---	---	---

Bước 3: $i = 2$: khởi tạo $max_index = i = 2 \rightarrow arr[max_index] = arr[2] = 8$. Tìm giá trị $arr[j]$ lớn nhất và lớn hơn $arr[max_index]$ trong đoạn $[3, 5]$
 $\rightarrow arr[j]$ nằm ở vị trí $max_index = 2 \rightarrow$ giữ nguyên

10	9	8	5	3	6
----	---	---	---	---	---

Bước 4: $i = 3$: khởi tạo $\text{max_index} = i = 3 \rightarrow \text{arr}[\text{max_index}] = \text{arr}[3] = 5$. Tìm giá trị $\text{arr}[j]$ lớn nhất và lớn hơn $\text{arr}[\text{max_index}]$ trong đoạn $[4, 5]$

$\rightarrow \text{arr}[j]$ nằm ở vị trí $\text{max_index} = 5 \rightarrow$ hoán vị $\text{arr}[5]$ với $\text{arr}[3]$

10	9	8	6	3	5
----	---	---	---	---	---

Bước 5: $i = 4$: khởi tạo $\text{max_index} = i = 4 \rightarrow \text{arr}[\text{max_index}] = \text{arr}[4] = 3$. Tìm giá trị $\text{arr}[j]$ lớn nhất và lớn hơn $\text{arr}[\text{max_index}]$ trong đoạn $[5, 5]$

$\rightarrow \text{arr}[j]$ nằm ở vị trí $\text{max_index} = 5 \rightarrow$ hoán vị $\text{arr}[5]$ với $\text{arr}[4]$

10	9	8	6	5	3
----	---	---	---	---	---

Bước 6: $i = 5$: tới điều kiện dừng vòng lặp, thuật toán kết thúc và ta được mảng sắp xếp giảm dần.

Câu 2: (3.5 điểm)

Cho dãy ký tự như sau: F, D, B, A, C, E, H, G, I

Hãy thực hiện các yêu cầu sau:

- Vẽ cây nhị phân tìm kiếm bằng cách thêm lần lượt từng ký tự vào cây theo thứ tự từ trái qua phải của dãy ký tự trên, biết rằng giá trị của từng ký tự tương ứng theo thứ tự xuất hiện của ký tự trong từ điển. (1 điểm)
- Cho biết kết quả duyệt cây theo RNL, NRL. (0.5 điểm)
- Hủy lần lượt từng nút D, E, F, H trên cây, mỗi lần hủy 1 nút vẽ lại cây nối tiếp theo thứ tự hủy. (1 điểm)
- Viết hàm đếm số lượng nút có một nút con trên cây, nếu cây rỗng thì in ra giá trị -1. (1 điểm)



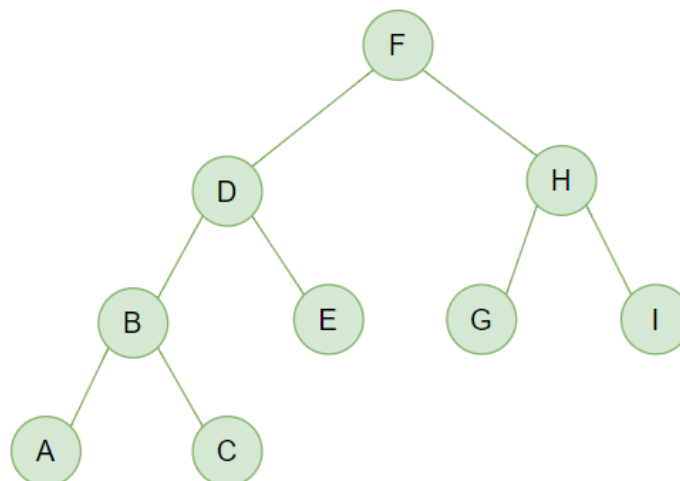
Đáp án tham khảo:

- Cây nhị phân tìm kiếm là cây nhị phân đảm bảo nguyên tắc bố trí khóa tại mỗi nút, sao cho:

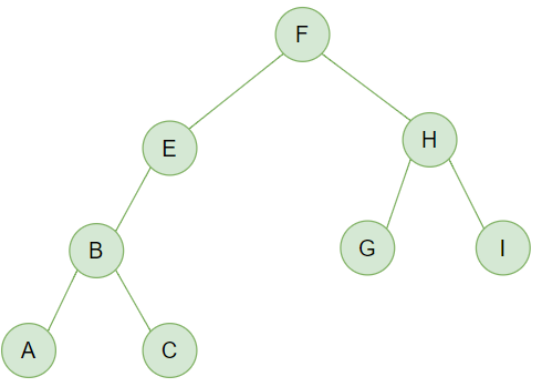
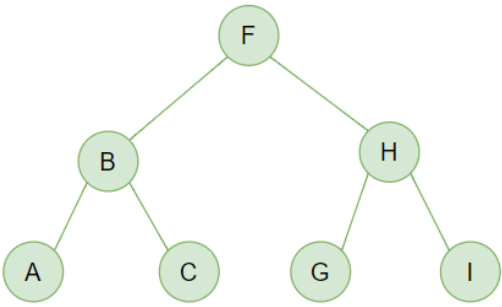
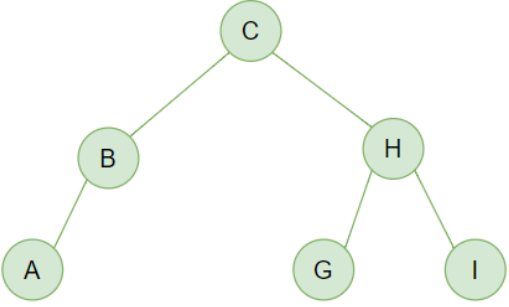
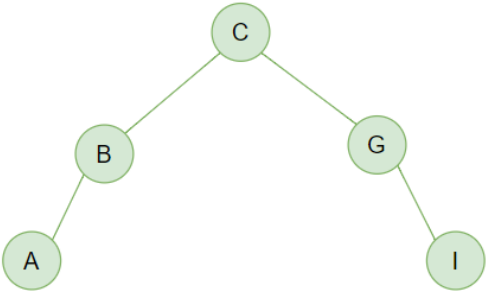
- Các nút trong cây con bên trái nhỏ hơn nút hiện hành.

- Các nút trong cây con bên phải lớn hơn nút hiện hành.

Để vẽ cây nhị phân tìm kiếm bằng các ký tự trên, ta phải xác định được thứ tự của các ký tự trong bảng chữ cái: $A < B < C < D < E < F < G < H < I$.



- Duyệt cây RNL: I, H, G, F, D, C, E, B, A
– Duyệt cây NRL: F, H, I, G, B, E, C, D, A
- Xóa D, E, F, H:

Nút cần xóa	Dạng nút	Vẽ lại cây
D	nút có đầy đủ 2 cây con (chọn nút thể mạng là nút E)	 <pre> graph TD F((F)) --- B((B)) F --- H((H)) B --- A((A)) B --- C((C)) H --- G((G)) H --- I((I)) </pre>
E	nút có 1 cây con	 <pre> graph TD F((F)) --- B((B)) F --- H((H)) B --- A((A)) B --- C((C)) H --- G((G)) H --- I((I)) </pre>
F	nút có đầy đủ 2 cây con (chọn nút thể mạng là nút C)	 <pre> graph TD C((C)) --- B((B)) C --- H((H)) B --- A((A)) B --- C2((C)) H --- G((G)) H --- I((I)) </pre>
H	nút có đầy đủ 2 cây con (chọn nút thể mạng là nút G)	 <pre> graph TD C((C)) --- B((B)) C --- G((G)) B --- A((A)) B --- C2((C)) G --- I((I)) </pre>

```

d. int DemNut(TREE t) {
    if (t == NULL)
        return -1;    // cây rỗng
    if (t->pLeft == NULL && t->pRight == NULL)

```

```

        return 0;        // nút lá
    int count = 0;
    if (t->pLeft != NULL && t->pRight == NULL)
        count++;
    if (t->pLeft == NULL && t->pRight != NULL)
        count++;
    count += DemNut(t->pLeft);
    count += DemNut(t->pRight);
    return count;
}

```

Câu 3: (2 điểm)

Cho biết cây B-Tree bậc 3 là một cây thỏa mãn các tính chất sau:

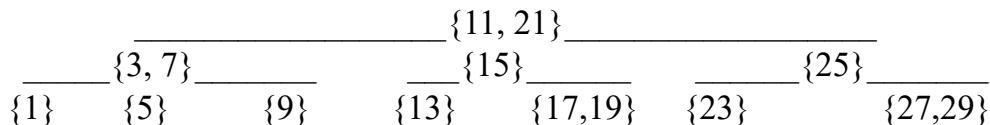
- Tất cả node lá nằm trên cùng một mức
- Tất cả các node, trừ node gốc và node lá, có ***tối thiểu*** 2 node con.
- Tất cả các node có ***tối đa*** 3 con
- Tất cả các node, trừ node gốc, có từ 1 cho đến 2 khóa (keys)
- Một node không phải lá và có n khóa thì phải có n + 1 node con.

Hãy thực hiện các yêu cầu sau:

3.1 Cho dãy số: 27, 19, 23, 9, 1, 3, 11, 21, 5, 13, 17, 15, 29, 25. Hỏi khi lần lượt thêm các số trong dãy theo thứ tự từ trái qua phải vào một cây B-Tree bậc 3 rồi thì:

- Các khóa nào khi thêm vào cây sẽ làm phát sinh thao tác split node? (0.5 điểm)
- Vẽ cây B-Tree trước và sau khi thêm các khóa ở câu a (1 điểm)

3.2 Cho cây B-Tree bậc 3 như hình sau:



Hãy lần lượt tiến hành xóa các khóa sau khỏi cây: 11, 21, 13 và vẽ cây B-Tree trước và sau khi xóa mỗi khóa trên (0.5 điểm)

Lưu ý khi xóa:

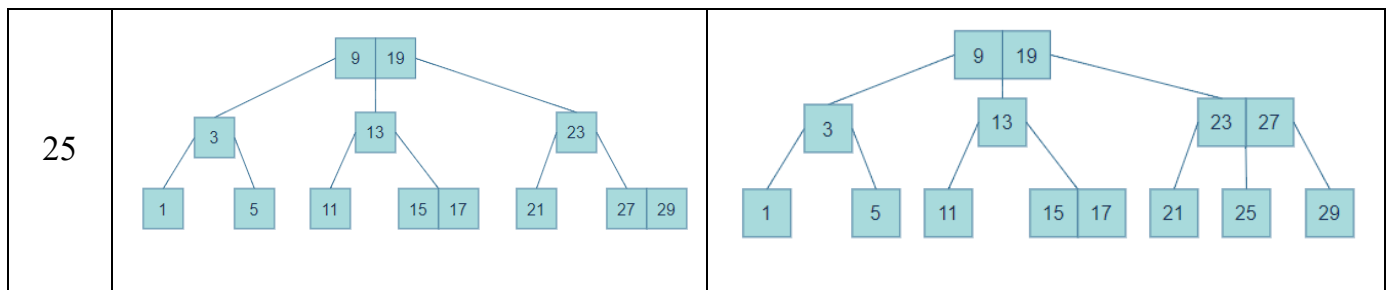
- Khi khóa cần xóa (gọi là x) không nằm ở node lá, chọn khóa thế mạng là khóa có giá trị lớn nhất mà nhỏ hơn x.
- Thao tác nhường khóa (underflow) sẽ được thực hiện khi hai node liền kề có tổng số khóa ≥ 2 . Khi có một node không còn đáp ứng đủ số lượng khóa tối thiểu, ưu tiên thực hiện underflow thay cho catenation (hợp) vì thao tác này không làm thay đổi số khóa của node cha.
- Khi có 02 lựa chọn node liền kề để thực hiện catenation, ưu tiên chọn catenation giữa node bị thiếu khóa với node liền trước.

Đáp án tham khảo:

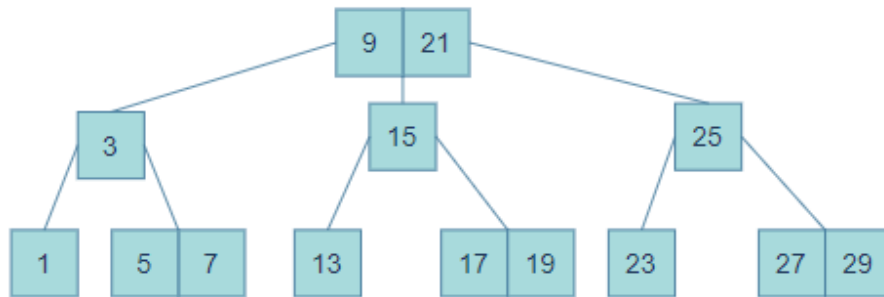
- Các khóa làm phát sinh thao tác split node: 23, 1, 21, 5, 17, 25
- Vẽ cây sau khi thêm các khóa:

(Lưu ý: Sinh viên không cần vẽ chi tiết từng bước thực hiện thao tác split, chỉ cần vẽ trạng thái cây trước khi tiến hành thêm khóa và cây sau khi thêm khóa vào hoàn tất. Tổng cộng có 12 hình cần vẽ. Ở câu này sinh viên bắt buộc phải làm theo thứ tự các khóa sẽ được thêm. Khi vẽ sai ở bước nào thì các bước sau sẽ không được điểm)

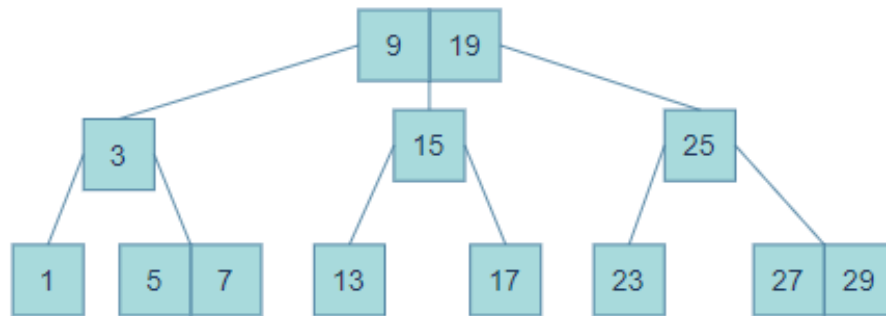
Khóa	Trước khi thêm	Sau khi thêm
23	<pre> graph TD Root["19 27"] </pre>	<pre> graph TD Root["23"] Root --- L["19"] Root --- R["27"] </pre>
1	<pre> graph TD Root["23"] Root --- L["9 19"] Root --- R["27"] </pre>	<pre> graph TD Root["19"] Root --- L["9"] Root --- R["23"] L --- L1["1 3"] L --- L2["11"] R --- R1["21"] R --- R2["27"] </pre>
21	<pre> graph TD Root["9 23"] Root --- L["1 3"] Root --- M["11 19"] Root --- R["27"] </pre>	<pre> graph TD Root["19"] Root --- L["9"] Root --- R["23"] L --- L1["1 3"] L --- L2["11"] R --- R1["21"] R --- R2["27"] </pre>
5	<pre> graph TD Root["19"] Root --- L["9"] Root --- R["23"] L --- L1["1 3"] L --- L2["11"] R --- R1["21"] R --- R2["27"] </pre>	<pre> graph TD Root["19"] Root --- L["3 9"] Root --- R["23"] L --- L1["1"] L --- L2["5"] L --- L3["11"] R --- R1["21"] R --- R2["27"] </pre>
17	<pre> graph TD Root["19"] Root --- L["3 9"] Root --- R["23"] L --- L1["1"] L --- L2["5"] L --- L3["11"] L --- L4["13"] R --- R1["21"] R --- R2["27"] </pre>	<pre> graph TD Root["9 19"] Root --- L["3"] Root --- M["13"] Root --- R["23"] L --- L1["1"] L --- L2["5"] M --- M1["11"] M --- M2["17"] R --- R1["21"] R --- R2["27"] </pre>



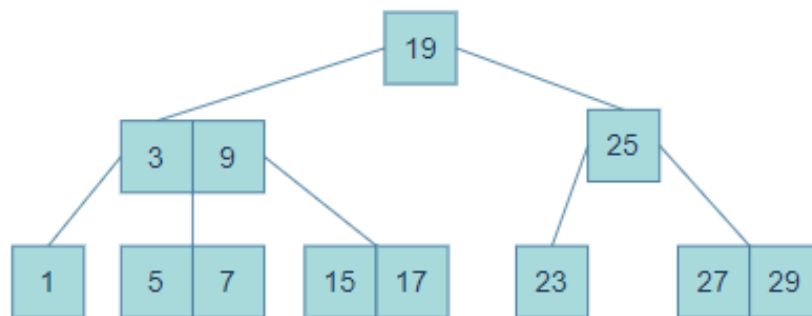
c. **Xóa 11:** swap 11 và 9 sau đó xóa lá và thực hiện catenate (chỉ có 1 lựa chọn để catenate)



Xóa 21: swap 21 và 19, thực hiện xóa tại lá và không cần điều chỉnh



Xóa 13: Xóa trực tiếp tại lá, thực hiện catenate, lan truyền quá trình catenate lên node con ở giữa của gốc, tại mức này ưu tiên catenate với node liền trước là {3} node gốc làm gốc bị giảm số khóa.



Câu 4: (2 điểm)

Cho một bảng băm theo phương pháp thăm dò bậc 2 với hàm băm $h(\text{key})$ và hàm băm lại (hay hàm thăm dò) $\text{prob}(\text{key}, i)$ như sau:

$$h(\text{key}) = (\text{key} \% M) \quad \text{prob}(\text{key}, i) = (h(\text{key}) + i * i) \% M$$

Trong đó:

- key là giá trị khóa.
- i là một số nguyên cho biết lần thăm dò thứ i.

- M là kích thước bảng băm.

Cho $M = 7$ và trên bảng băm đã chứa các mục dữ liệu như bên dưới. Biết EMP và DEL lần lượt là ký hiệu để đánh dấu vị trí còn trống hoặc bị xóa trong bảng băm.

	Key
0	EMP
1	EMP
2	2
3	EMP
4	4
5	EMP
6	EMP

- a. Trình bày từng bước việc thêm các khóa **Key** trong danh sách bên dưới vào bảng băm theo đúng thứ tự trong danh sách. (1 điểm)

STT	Key
1	6
2	16
3	10

- b. Trình bày từng bước việc xóa giá trị **Key** = 16 trong bảng băm khi hoàn thành yêu cầu ở câu a. (0.5 điểm)
- c. Trình bày từng bước việc xóa giá trị **Key** = 10 trong bảng băm khi hoàn thành yêu cầu ở câu b. (0.5 điểm)

 **Đáp án tham khảo:**

- a. Các key: 6, 16, 10

Thêm 6: $h(6) = (6 \bmod 7) = 6$. Vị trí 6 còn trống \rightarrow thêm 6 vào vị trí 6.

	Key
0	EMP
1	EMP
2	2
3	EMP
4	4
5	EMP
6	6

Thêm 16: $h(16) = (16 \bmod 7) = 2$. Vị trí 2 xảy ra đụng độ \rightarrow băm lại.

+ Băm lại lần 1: $prob(16, 1) = (2 + 1.1) \bmod 7 = 3$. Vị trí 3 còn trống \rightarrow thêm 16 vào vị trí 3.

	Key
0	EMP
1	EMP
2	2
3	16
4	4
5	EMP
6	6

Thêm 10: $h(10) = (10 \bmod 7) = 3$. Vị trí 3 xảy ra đụng độ \rightarrow băm lại.

+ Băm lại lần 1: $\text{prob}(10, 1) = (3 + 1.1) \bmod 7 = 4$. Vị trí 4 xảy ra đụng độ \rightarrow băm lại

+ Băm lại lần 2: $\text{prob}(10, 2) = (3 + 2.2) \bmod 7 = 0$. Vị trí 0 còn trống \rightarrow thêm 10 vào vị trí 0.

	Key
0	10
1	EMP
2	2
3	16
4	4
5	EMP
6	6

b. Xóa 16: $h(16) = (16 \bmod 7) = 2$. Vị trí 2 chứa giá trị khác \rightarrow băm lại.

+ Băm lại lần 1: $\text{prob}(16, 1) = (2 + 1.1) \bmod 7 = 3$. Vị trí 3 chứa 16, xóa 16.

	Key
0	10
1	EMP
2	2
3	DEL
4	4
5	EMP
6	6

c. Xóa 10: $h(10) = (10 \bmod 7) = 3$. Vị trí 3 chứa giá trị khác \rightarrow băm lại.

+ Băm lại lần 1: $\text{prob}(10, 1) = (3 + 1.1) \bmod 7 = 4$. Vị trí 4 chứa giá trị khác \rightarrow băm lại

+ Băm lại lần 2: $\text{prob}(10, 2) = (3 + 2.2) \bmod 7 = 0$. Vị trí 0 chứa 10, xóa 10.

	Key
0	DEL
1	EMP
2	2
3	DEL
4	4
5	EMP
6	6

Câu 5: (1 điểm)

Cho bài toán “Tô màu bản đồ” được đặt ra như sau: Có một bản đồ các quốc gia trên thế giới, ta muốn tô màu các quốc gia này sao cho hai nước có cùng ranh giới được tô khác màu nhau. Yêu cầu tìm cách tô sao cho số màu sử dụng là ít nhất. Bài toán có thể được mô hình hóa thành một bài toán trên đồ thị, khi đó mỗi nước trên bản đồ là một đỉnh của đồ thị, hai nước láng giềng tương ứng với hai đỉnh kề nhau được nối với nhau bằng một cạnh, bài toán trở thành: tô màu các đỉnh của đồ thị sao cho mỗi đỉnh chỉ được tô một màu, hai đỉnh kề nhau có màu khác nhau và số màu sử dụng là ít nhất.

Giả sử cho thông tin đầu vào của bài toán được nhập vào chương trình sau:

Input	Giải thích
15	- Dòng đầu tiên chứa một số e là số cạnh của đồ thị
Viet_Nam Lao	- Với e dòng tiếp theo, mỗi dòng chứa 02 chuỗi u, i thể hiện thông tin có một cạnh nối từ đỉnh u sang đỉnh i trong đồ thị
Viet_Nam Trung_Quoc	
Thai_Lan Lao	
....	
Campuchia Thai_Lan	Lưu ý: không biết trước số đỉnh và danh sách các đỉnh.

Hãy thực hiện các yêu cầu sau:

- Xây dựng cấu trúc dữ liệu thích hợp để biểu diễn đồ thị nhằm lưu trữ các thông tin cần thiết trên bản đồ. (0.5 điểm)
- Viết hàm nhập đồ thị (bằng cách nhập số cạnh và danh sách các cạnh như ví dụ ở trên) và lưu trữ thông tin của đồ thị vào cấu trúc dữ liệu đã đề xuất ở câu a. (0.5 điểm)



Đáp án tham khảo:

```
a. map<string, set<string>> listCountry;  
b. void Input() {  
    for(int j = 0; j < e; j++) {  
        string u, i;  
        cin >> u >> i;  
        listCountry[u].insert(i);  
        listCountry[i].insert(u);  
    }  
}
```

HẾT