

ĐỀ 1

Thời gian: 90 phút

(Sinh viên không được sử dụng tài liệu)

HỌ VÀ TÊN SV:	CÁN BỘ COI THI	ĐIỂM SỐ
MSSV:		
STT:		
PHÒNG THI:.....		

PHẦN 1: CÂU TRẢ LỜI NGẮN (6 điểm)

Câu 1 (1.25 điểm) Hãy liên kết các đặc trưng ở cột bên trái với các thuật toán tương ứng ở cột bên phải. Các thuật toán có thể có những đặc trưng giống nhau.

Đặc trưng	
1	Độ phức tạp của thuật toán trong trường hợp xấu nhất là $O(n \log n)$
2	Độ phức tạp của thuật toán trong trường hợp trung bình là $O(n^2)$
3	Thuật toán được phân loại là Online Sorting
4	Thuật toán sắp xếp không dựa trên kết quả so sánh giá trị (khóa) giữa các phần tử trong danh sách
5	Thuật toán có số lượng phép so sánh trong mọi trường hợp (xấu nhất, trung bình, tốt nhất) đều như nhau
6	Thuật toán có số lần hoán vị ít (xấu nhất là $n-1$ lần)
7	Thuật toán được thiết kế theo chiến lược chia để trị

Thuật toán sắp xếp	
A	Selection sort
B	Insertion sort
C	Quick sort
D	Merge sort
E	Heap sort
F	Counting sort
G	Radix sort

Trả lời	Thang điểm (1.25 điểm)
Phần nội dung kiểm tra: 1- D, 1- E 2- A, 2- B 3- B 7- C, 7- D	Phần này có 7 liên kết, đạt 0.25đ cho 1 liên kết đúng với điều kiện như sau: Ứng với mỗi đặc trưng, + nếu câu trả lời bị đư hoặc sai 1 thành phần thì 0đ , tức không tính bất kỳ phần điểm nào dù có liên kết đúng trong số đó +nếu thiếu: vẫn đạt 0.25đ cho liên kết đúng Ví dụ: + nếu SV chỉ ghi duy nhất 1-D hoặc 1-E thì đạt 0.25đ

	+ nếu trả lời 1-D,E,F vì dư F nên 0đ , dù D,E đúng +nếu trả lời 1-D,C vì sai C nên 0đ
Phần mở rộng: 4- F,G Đặc trưng 5 và 6 không chấm	Chỉ cần có liên kết 4-F hoặc 4-G và không có thêm liên kết nào khác thì đạt 0.25đ Tối đa chỉ đạt 0.25đ cho đặc trưng 4, nếu liệt kê cả 2 cũng chỉ đạt 0.25đ
Tổng điểm 2 phần, tối đa chỉ đạt 1.25đ	

Ghi chú: Counting sort và Radix sort là 2 thuật toán được chú thích trong đề cương là “Giới thiệu”, nghĩa là GV không cần giảng dạy chi tiết tại lớp, có thể giới thiệu hoặc đề nghị/khuyến khích SV tự tìm hiểu thêm. Trong slide bài giảng chung gửi cho SV có trình bày 2 thuật toán. Hai thuật toán được nhắc đến trong đề thi như là một cách thức kiểm tra tính tự học, tự nghiên cứu của SV và cho điểm thưởng ở tính tự giác này (một lí do phụ khác là gây nhiễu). Chỉ cần câu trả lời của SV đúng 5 liên kết là đã đạt trọn điểm của câu hỏi là 1.25 điểm. Nếu bỏ phần đáp án liên quan đến 2 thuật toán này thì vẫn còn nhiều hơn 5 liên kết đúng khác.

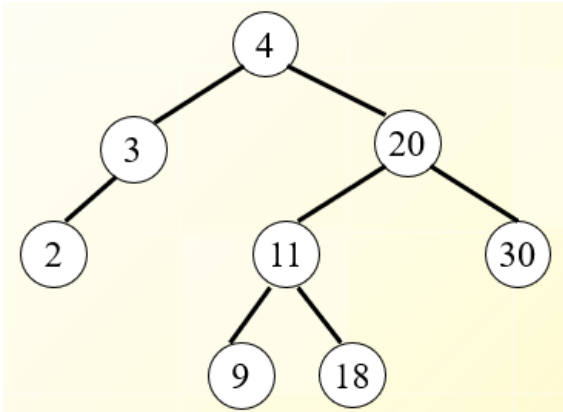
III. CÁC GIẢI THUẬT SẮP XẾP	III. CÁC GIẢI THUẬT SẮP XẾP
❖PHƯƠNG PHÁP ĐẾM <u>Đánh giá:</u> Trong mọi trường hợp, độ phức tạp tính toán của Counting Sort là $O(n+k)$, trong đó k là kích thước của mảng B . Counting Sort là một trong những thuật toán sắp xếp không dựa vào kết quả so sánh giá trị khóa của các phần tử trong danh sách.	❖PHƯƠNG PHÁP CƠ SỐ <u>Đánh giá:</u> - Radix Sort cũng là một trong những thuật toán sắp xếp không dựa trên kết quả so sánh giá trị khóa giữa các phần tử trong danh sách. - Độ phức tạp tính toán là $O(m.n)$ trong đó m là số ký tự lớn nhất của một phần tử trong danh sách. - Radix Sort thích hợp với cấu trúc là danh sách liên kết hơn là dùng cấu trúc mảng

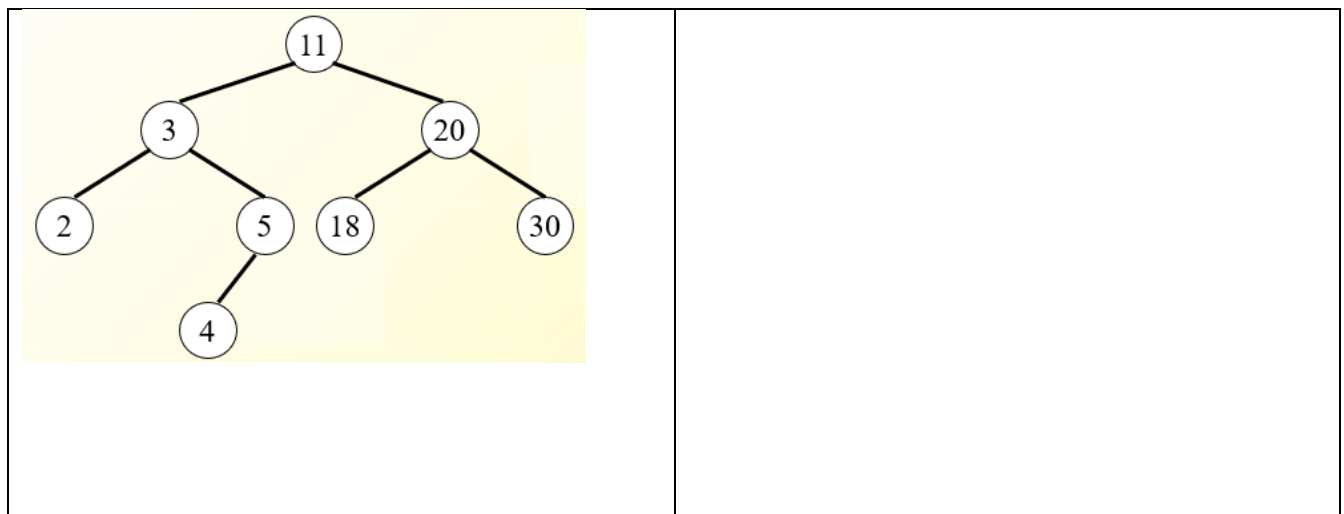
Câu 2 (0.5 điểm) Cho một mảng gồm 7 số nguyên như sau: **4, 9, 5, 6, 10, 2, 3**. Hãy cho biết mảng sẽ thay đổi qua từng bước như thế nào khi áp dụng thuật toán QuickSort theo mã giả bên dưới, để sắp xếp mảng theo thứ tự **giảm dần**.

Mã giả: <pre> void QuickSort(int a[], int left, int right) { int i, j, x; x = a[(left+right)/2]; i = left; j = right; while(i <= j) { while(a[i] > x) i++; while(a[j] < x) j--; if(i <= j) Doicho(a[i],a[j]); i++; j--; } if(left<j) QuickSort(a, left, j); if(i<right) QuickSort(a, i, right); } </pre>
--

Trả lời	Thang điểm (0.5 điểm)
<p>Mảng ban đầu: 4, 9, 5, 6, 10, 2, 3</p> <p><u>10</u>, 9, 5, 6, <u>4</u>, 2, 3</p> <p>10, 9, <u>6</u>, <u>5</u>, 4, 2, 3</p> <p>10, 9, 6, 5, 4, <u>3</u>, <u>2</u></p>	<ul style="list-style-type: none"> - Đúng chính xác 3 lần hoán vị: 0.5đ - Nếu SV ghi nhiều hơn 3 dòng, với những dòng ở giữa, mảng không thay đổi thì vẫn đạt trọn 0.5đ (vì có vài lần hoán vị tại chỗ) - Nếu đúng 2 dòng đầu, thiếu dòng cuối (đổi chỗ 2,3) thì đạt 0.25đ - Dư/thiếu/sai bất kỳ 1 lần cập nhật nào khác: 0đ

Câu 3 (1 điểm) Xét giải thuật tạo cây nhị phân tìm kiếm, thứ tự các giá trị nhập vào cây T_1 như sau: **8, 3, 5, 2, 20, 11, 30, 9, 18, 4**. Yêu cầu: (a) Cho biết kết quả duyệt cây T_1 theo thứ tự Right – Left – Node (RLN); (b) Hãy vẽ lại hình ảnh cây T_1 khi xóa 2 lần nút gốc sao cho cây vẫn là cây nhị phân tìm kiếm.

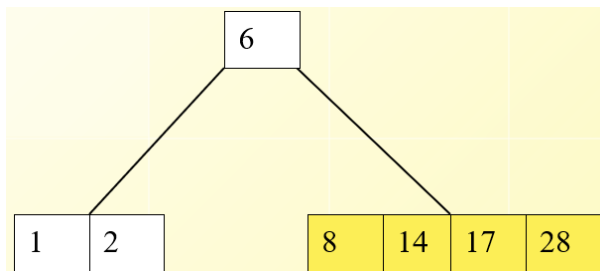
Trả lời	Thang điểm (1 điểm)
<p>(a) Kết quả duyệt RLN của cây T_1:</p> <p>30, 18, 9, 11, 20, 4, 5, 2, 3, 8</p>	<ul style="list-style-type: none"> - Đúng chính xác 10 vị trí: 0.5đ - Chỉ sai thứ tự 1 cặp số liền kề, ví dụ, (30, 18, 9, 11, 20, <u>5</u>, <u>4</u>, 2, 3, 8) chỉ sai cặp (5,4): 0.25đ - Đúng chính xác 9 vị trí đầu, thiếu số 8 (có nhiều SV quên xuất node gốc cuối): chấm chước 0.25đ - Dư/thiếu/sai bất kỳ vị trí nào khác: 0đ
<p>(b) Hình ảnh cây T_1 sau khi xóa 2 lần nút gốc:</p> <p>Chỉ có 2 đáp án đúng:</p> 	<ul style="list-style-type: none"> - Đúng 1 trong 2 cây: 0.5đ - Có nhiều SV vẽ thiếu node 4 ở cây thứ 2: chấm chước 0.25đ - Dư/thiếu/sai bất kỳ vị trí nào khác: 0đ - Chỉ chấm cây kết quả sau 2 lần xóa, không chấm những cây trung gian



Câu 4 (1 điểm) Tạo một cây B-Tree T_2 bậc 5 với thứ tự thêm các giá trị vào cây như sau: **1, 12, 8, 2, 25, 6, 14, 28, 17**. Yêu cầu:(a) Vẽ cây T_2 ; (b) Sau đó, lần lượt xoá 2 giá trị là 12 và 25 ra khỏi cây T_2 , hãy vẽ trạng thái cây sau mỗi lượt xoá. Quy ước: Trong quá trình xoá, ưu tiên thực hiện thủ tục nhường khóa (underflow) trước thủ tục gộp (catenation).

Trả lời	Thang điểm (1 điểm)
<p>(a) Hình ảnh cây T_2 được tạo từ dãy số ban đầu:</p>	<p>-Đúng tất cả các vị trí: 0.5đ</p> <p>-Sai bất kỳ 1 vị trí nào: 0đ</p> <p>-Nếu câu a sai thì không chấm câu b. Nhiều bài làm câu a sai nhưng vô hình các cây ở câu b đúng, vẫn 0đ</p>
<p>(b) Hình ảnh cây T_2 sau khi xoá số 12:</p>	<p>0.25đ</p> <p>Sai bất kỳ 1 vị trí nào: 0đ</p>

(b) Hình ảnh cây T_2 sau khi xóa số 25 (đã xóa số 12 trước đó):



0.25đ

Sai bất kỳ 1 vị trí nào: **0đ**

Câu 5 (1 điểm) Thêm các khoá **37, 28, 24, 7, 71** vào một bảng băm địa chỉ mở HT, có kích thước $M = 13$, sử dụng hàm băm $h_1(\text{key}) = \text{key} \% M$. Biết rằng, phương pháp giải quyết xung đột là băm kép (double hashing), với hàm băm phụ $h_2(\text{key}) = 11 - (\text{key} \% 11)$, và hàm băm lại $h(\text{key}, i) = (h_1(\text{key}) + i * h_2(\text{key})) \% M$.

Yêu cầu: (a) Hãy trình bày từng bước việc thêm các khóa vào HT bằng cách điền kết quả tính toán vào Bảng 1; (b) Cho biết vị trí thêm các khóa ở Bảng 2.

Bảng 1: Minh họa các bước tính toán trong quá trình thêm 5 khóa trên

key	$h_1(\text{key})$	$h_2(\text{key})$	Kết quả băm lại lần 1 (nếu có)	Kết quả băm lại lần 2 (nếu có)	Lần 3	Thang điểm
37	11					0.25đ
28	2					
24	11	9	7			0.25đ
7	7	4	11	2	6	0.25đ
71	6	6	12			0.25đ

- Đúng từng dòng tính toán trong Bảng 1 và vị trí thêm phần tử tương ứng trong Bảng 2: **0.25đ**/mỗi kết quả đúng, trừ (37, 28) đúng cả 2 số chỉ đạt **0.25đ**
- Nếu vị trí thêm trong Bảng 2 và kết quả tính toán trong Bảng 1 không khớp nhau thì không tính điểm (vì SV copy nhau)
- Nếu các vị trí trong Bảng 2 đều sai nhưng đúng hết 5 số ở cột $h_1(\text{key})$ trong Bảng 1 thì chấm chước cho **0.25đ**

Bảng 2: Bảng băm HT sau khi thêm 5 khóa trên

Chỉ số (index)	0	1	2	3	4	5	6	7	8	9	10	11	12		
Khóa (key)			28				7	24				37	71		

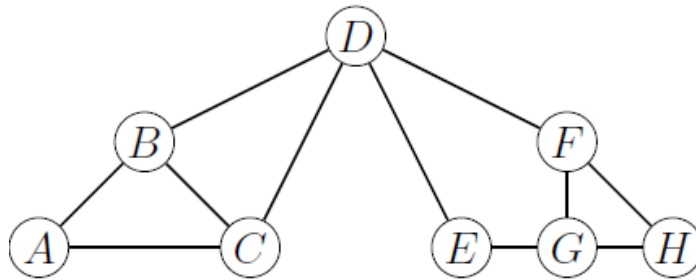
Câu 6 (1.25 điểm)

Bạn đang chơi trò chơi MazeCraft, có giao diện như Hình 1. Nhận thấy rằng, bạn có thể chuyển đổi mê cung thành một đồ thị, trong đó các đỉnh đại diện cho các ô và các cạnh đại diện cho các lối đi, bạn muốn sử dụng các thuật toán tìm kiếm trên đồ thị vừa học để tìm đường đi qua mê cung.



Hình 1: Ví dụ về một mê cung trong game MazeCraft

Hãy xem xét đồ thị G_1 (Hình 2) đã được chuyển đổi dưới đây.



Hình 2: Đồ thị vô hướng G_1

Giả sử rằng, đồ thị được biểu diễn bằng danh sách kề (adjacency list) và tất cả các danh sách kề được sắp xếp, nghĩa là ứng với mỗi đỉnh i , ta cần lưu trữ một danh sách có thứ tự gồm các đỉnh kề với đỉnh i . Các đỉnh kề này được sắp xếp theo thứ tự bảng chữ cái. Thứ tự bảng chữ cái là A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z.

Hãy thực hiện các yêu cầu sau:

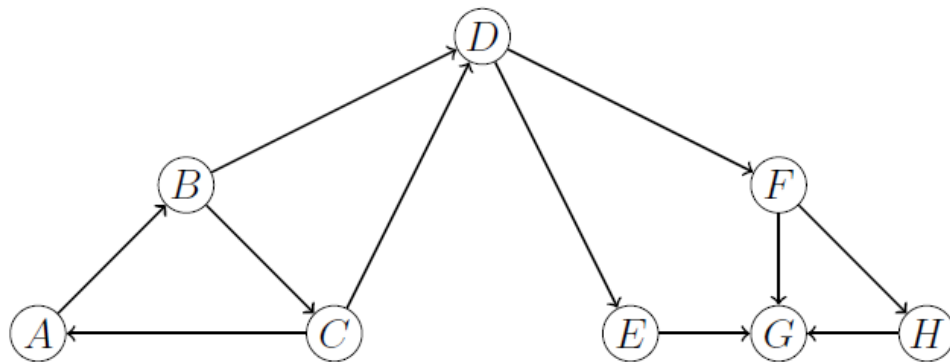
(a) Giả sử bạn muốn tìm một đường đi từ đỉnh A đến đỉnh H. Nếu bạn sử dụng thuật toán tìm kiếm theo chiều rộng (Breadth-First Search, BFS), hãy **viết lại đường đi kết quả dưới dạng một chuỗi các đỉnh**.

Trả lời	Thang điểm
Kết quả đường đi theo BFS: chỉ có duy nhất 1 đáp án A, B, D, F, H	- Đúng chính xác chuỗi đáp án: 0.25đ - Dư/thiếu/sai bất kỳ vị trí nào: 0đ

(b) Nếu bạn sử dụng thuật toán tìm kiếm theo chiều sâu (Depth-First Search, DFS) để tìm một đường đi từ đỉnh A đến đỉnh H, hãy **viết lại đường đi kết quả dưới dạng một chuỗi các đỉnh, kèm theo chú thích là bạn chọn cài đặt theo cách thức nào**. Ví dụ, có 2 cách cài đặt DFS phổ biến là sử dụng Ngăn xếp để lưu các đỉnh chờ duyệt (Stack-based implementation) hoặc dùng kỹ thuật đệ quy quay lui (Recursion-based implementation, Backtracking).

Trả lời	Thang điểm
<p>Kết quả đường đi theo DFS: có 2 đáp án đúng</p> <p>1. Dừng stack: (A, C, D, F, H)</p> <p>2. Dừng đệ quy: (A, B, C, D, E, G, F, H) hoặc (A, C, D, F, H) khi duyệt danh sách kề ngược</p>	<p>- Đúng 1 trong 2 đáp án: 0.25đ</p> <p>- Dư/thiếu/sai bất kỳ vị trí nào: 0đ (nhiều bài làm chỉ bị dư 1 đỉnh cũng là sai)</p> <p>- Nếu SV không chú thích là dùng cách cài đặt nào mà đúng 1 trong 2 đáp án thì cũng cho đạt 0.25đ điểm</p>

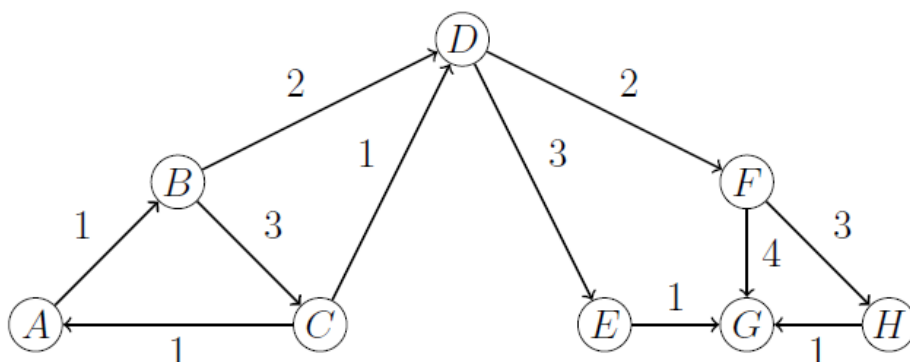
(c) Bây giờ giả sử rằng, các lối đi trong mê cung có hướng. Chạy lại thuật toán tìm kiếm theo chiều sâu (DFS) trên đồ thị có hướng G2 (Hình 3) bên dưới.



Hình 3: Đồ thị có hướng G2

Trả lời	Thang điểm
<p>Kết quả đường đi theo DFS: có 2 đáp án đúng</p> <p>1. Nếu câu trả lời ở câu b là (A, C, D, F, H) thì kết quả câu c là (A, B, D, F, H)</p> <p>2. Nếu câu trả lời ở câu b là (A, B, C, D, E, G, F, H) thì kết quả câu c là (A, B, C, D, F, H)</p>	<p>- Đúng 1 trong 2 đáp án: 0.25đ</p> <p>- Dư/thiếu/sai bất kỳ vị trí nào: 0đ</p> <p>- Nếu câu b sai thì chỉ cần đúng 1 trong 2 đáp án là đạt 0.25đ, nhưng nếu câu b đúng, kết quả phải khớp theo cách cài đặt thì mới tính điểm</p>

(d) Giả sử mỗi lối đi trong mê cung gây ra một lượng sát thương khác nhau cho bạn trong trò chơi. Đồ thị được bổ sung thêm các trọng số để biểu diễn sát thương gây ra bởi mỗi cạnh. Sử dụng thuật toán Dijkstra để tìm đường đi từ đỉnh A đến đỉnh H trong đồ thị G3 (Hình 4) với sát thương thấp nhất có thể. Hãy ghi lại thứ tự các đỉnh được loại bỏ khỏi hàng đợi ưu tiên (priority queue) khi chạy thuật toán Dijkstra.



Hình 4: Đồ thị có trọng số G3

Trả lời	Thang điểm
<p>Thứ tự các đỉnh được loại bỏ khỏi hàng đợi ưu tiên:</p> <p>A, B, D, C, F, E, G, H</p>	<p>- Đúng chính xác chuỗi đáp án: 0.5đ</p> <p>- (A, B, D, C, F, E, H, G) sai thứ tự cặp đỉnh cuối, hoặc (A, B, D, C, F, E, G) thiếu H: 0.25đ</p> <p>- Dư/thiếu/sai bất kỳ vị trí nào khác: 0đ (nhiều bài làm chỉ bị ngược vị trí giữa D,C vẫn 0đ)</p>

PHẦN 2: TỰ LUẬN (4 điểm)

Câu 7 (1.5 điểm) Hãy khai báo kiểu dữ liệu danh sách liên kết đơn mà mỗi phần tử chứa thông tin về một quốc gia gồm tên quốc gia, dân số, diện tích. Với các kiểu dữ liệu vừa khai báo, hãy xây dựng các hàm sau:

- Hàm nhập danh sách các quốc gia bằng cách thêm từng quốc gia vào **đầu** danh sách.
- Hàm sắp xếp danh sách các quốc gia theo tên quốc gia dùng thuật toán **Selection Sort**.

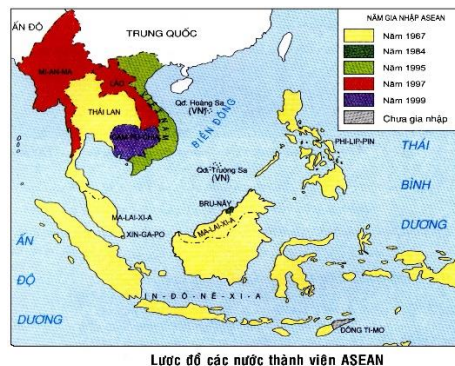
Code tham khảo	Thang điểm (1.5 điểm)
<pre>typedef struct Country { string name; int pop; float area; }Co; struct node { Co info; node *next; };</pre>	<p>- Đủ thông tin của 1 node (bắt buộc có con trỏ next) và quản lý được DSLK (bắt buộc có con trỏ trỏ tới node đầu): 0.5đ</p> <p>- Phải định nghĩa được cấu trúc của 1 node thì</p>

<pre>struct List { node *head,*tail; };</pre> <p>- Không có *tail cũng được</p> <p>- Không tạo struct List mà dùng node* thay thế cho List khi khai báo tham số của hàm cũng được</p> <p>- Nếu không tạo struct Country riêng mà gộp name, pop, area vào trong struct node thì cũng được (mặc dù thiết kế như vậy không tốt vì phải khai báo nhiều biến hoặc khi truyền tham số cho các hàm phải đưa vào nhiều tham số)</p>	<p>mới có điểm, nếu chỉ có Country với 3 thuộc tính (name, pop, area) thì 0đ</p>
<pre>void Init (List &L) { L.head=L.tail=NULL;} void inputCountry(Co &x) { cin.ignore(); getline(cin, x.name); cin>>x.pop>>x.area; } node* getNode(Co x) { node *p=new node; if (p) { p->info=x; p->next=NULL; } return p; }</pre>	<p>Không tính điểm phần này</p>
<pre>void addHead(List &L, Co x) { node* p = getNode(x); if (!L.head) //(L.head==NULL) L.head=L.tail=p; else { p->next=L.head; //liên kết 1 L.head=p; //liên kết 2 } } //Không có nhánh if cũng được, khi DS rỗng thì hàm vẫn xử lý đúng void inputList(List &L,int n) { Co x; for (int i=0;i<n;i++) { inputCountry(x); addHead (L,x); } } -----Cách khác: void addHead(node* &L, Co x) // không tạo struct List</pre>	<p>- Chạy được và cho kết quả đúng: 0.5đ</p> <p>Ví dụ 1, chỉ có addHead và đúng, không nhập danh sách: vẫn chậm chước cho đạt 0.5đ (vì đây là xử lý trọng tâm nhất)</p> <p>Ví dụ 2, xử lý addHead sai LỚN nhưng có nhập danh sách (nhầm rõ yêu cầu của đề): 0.25đ</p> <p>Ví dụ 3, xử lý addHead sai do thiếu 1 trong 2 liên kết (có đánh dấu) nhưng có một số đoạn code chấp nhận được như tạo node mới/nhập</p>

<p>hoặc</p> <pre>void addHead(List &L, Co x)// có typedef node* List { node* p = getNode(x); p->next=L; //liên kết 1 L=p; //liên kết 2 // Khi DS rỗng thì hàm vẫn xử lý đúng }</pre>	<p>thông tin một quốc gia/khởi tạo List/xử lý trường hợp List rỗng...(GV đánh giá là SV có hiểu biết về những xử lý cơ bản trên DSLK): 0.25đ</p> <p>- Đề yêu cầu thêm đầu, nếu SV thêm cuối là sai đề: 0đ (vì có khả năng SV copy nhau)</p>
<pre>void SelectionSort(List &L) { node *i=L.head, *j=NULL, *pmin; while(i->next)//hoặc while(i != L.tail) { pmin=i; j=i->next; while(j) { if(j->info.name < pmin->info.name) pmin=j; j=j->next; } swap(pmin->info,i->info); i=i->next; } } ----- void SelectionSort(List &L) { node *i,*j, *pmin; for(i=L.head; i->next;i=i->next) { pmin=i; for(j=i->next;j;j=j->next) { if(j->info.name < pmin->info.name) pmin=j; } swap(pmin->info,i->info); } }</pre>	<p>Ghi chú: đây là câu hỏi phân loại SV khá, giới nên chấm gắt.</p> <p>- Đúng kỹ thuật Selection Sort, chạy được và cho kết quả đúng: 0.5đ</p> <p>- Thuật toán sắp xếp khác Selection Sort, dù đúng cũng 0đ</p> <p>- Sắp xếp theo kiểu duyệt mảng (tức i,j chạy từ 1 đến n, i++), không phải duyệt trên DSLK, dù là Selection Sort cũng 0đ</p> <p>- Có lỗi: PHẢI trừ điểm, không đạt mức 0.5đ</p>

Câu 8 (2.5 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 (Hình 5).



Hình 5: Ví dụ về một bản đồ

Hãy thực hiện các yêu cầu sau:

- Xác định thông tin đầu vào (input), thông tin đầu ra (output) của bài toán.
- Hãy mô hình hóa bài toán trên thành một bài toán trên đồ thị, tức cho biết đỉnh và cạnh của đồ thị biểu diễn thông tin gì, đồ thị có trọng số hay không, yêu cầu xử lý gì với đồ thị đó.
- Xây dựng các cấu trúc dữ liệu phù hợp nhất có thể để biểu diễn đồ thị trên máy tính theo input đã cho ở Câu a.

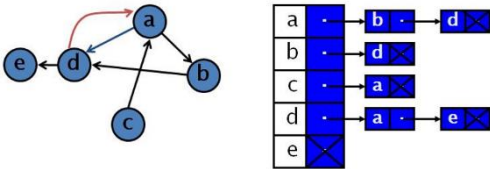
Cấu trúc được xem là tốt nếu đạt được các tiêu chuẩn sau: Tiết kiệm tài nguyên; Hỗ trợ một số thao tác cơ bản như “Kiểm tra hai đỉnh có kề nhau không”, “Tìm danh sách các đỉnh kề với một đỉnh cho trước” với ràng buộc là không phải duyệt qua danh sách tất cả các cạnh của đồ thị.

- Giả sử chương trình đã có sẵn các hàm hỗ trợ nhập thông tin đầu vào và xuất thông tin đầu ra theo cấu trúc dữ liệu ở Câu c, cùng với các chỉ thị `#include` cần thiết. Hãy cài đặt một (hoặc một số) hàm cho biết **tên của quốc gia có đường biên giới chung với nhiều quốc gia khác nhất** trong bản đồ.

Sinh viên được phép sử dụng Standard Template Library-STL với những cấu trúc dữ liệu (vector, stack, queue, list, map, set, pair, ...) cũng như giải thuật được xây dựng sẵn.

Bài làm tham khảo		Thang điểm (2.5 điểm)
(a)Xác định thông tin đầu vào (input), thông tin đầu ra (output) của bài toán		0.5đ
Chỉ là phân tích sơ bộ nên không cần quá hình thức Input: 1.Thông tin của bản đồ gồm: - Số lượng các quốc gia trong bản đồ. - Danh sách tên của các quốc gia. - Thông tin mỗi quốc gia có giáp với những quốc gia nào: thể hiện dưới dạng ma trận hay danh sách các cặp quốc gia có giáp nhau.	Cách trình bày khác hình thức hơn: Input: - Hai số nguyên v,e lần lượt là số quốc gia và số đường ranh giới giữa các quốc gia. - v chuỗi là danh tên các quốc gia. Chuỗi không có khoảng trắng, các từ trong tên quốc gia nối nhau bởi dấu <code>_</code> .	Ví dụ 1, nếu SV chỉ ghi chung chung như bên dưới thì không có điểm: 0đ <i>“Input: bản đồ các quốc gia. Output: kết quả tô màu”</i> (không đạt)

<p>2. Thông tin về bảng màu dùng để tô gồm số lượng màu và mã hay tên của các màu (không có cũng được).</p> <p>Output:</p> <ul style="list-style-type: none"> - Số lượng màu được dùng (không có cũng được) - Một phương án tô màu cho biết mã màu đã tô cho mỗi quốc gia. <p>Ghi chú</p> <ul style="list-style-type: none"> - Do chưa cần đề cập đến đồ thị lúc này nên không cần ghi rõ nhập số đỉnh, số cạnh, tên các đỉnh, ... - Không có dữ kiện về bảng màu cũng được. Nếu không cần nhập bảng màu thì xem như quy ước các màu được đánh số từ 0 (màu đen #000000) đến 16777215 (màu trắng #FFFFFF) trong hệ màu hệ màu RGB 24 bit. 	<ul style="list-style-type: none"> - e dòng tiếp theo, mỗi dòng chứa hai chuỗi a, b, thể hiện thông tin quốc gia a giáp với quốc gia b trong bản đồ. (hoặc cách khác là nhập dưới dạng ma trận gồm v dòng, mỗi dòng chứa v số nguyên) - Bảng màu dùng để tô gồm số lượng màu và mã hay tên của các màu (không có dữ kiện này cũng được). <p>Output:</p> <ul style="list-style-type: none"> - v số nguyên/chuỗi là mã/tên màu đã tô cho các quốc gia. Giá trị đầu tiên cho biết màu tô cho quốc gia đầu tiên trong danh sách tên quốc gia đã nhập (hoặc một cấu trúc nào khác cho biết thông tin mỗi quốc gia được tô màu nào) - một số nguyên cho biết số lượng màu được dùng (không có cũng được) 	<p>Ví dụ 2: bài làm như sau vẫn được chấm 0.5đ.</p> <p><i>“Input: số quốc gia, tên các quốc gia, danh sách các quốc gia liền kề của từng quốc gia. Output: màu của từng quốc gia”</i> (đạt)</p> <p>Ghi chú: đề thi các HK trước có cung cấp sẵn thông tin này, nếu SV ôn tập các đề cũ thì sẽ có lợi thế hơn.</p>
(b) Hãy mô hình hóa bài toán trên thành một bài toán trên đồ thị		0.5đ
<ul style="list-style-type: none"> - Đỉnh tương ứng cho một quốc gia trên bản đồ - Cạnh nối giữa 2 đỉnh cho biết 2 quốc gia tương ứng giáp với nhau. - Không cần trọng số, đồ thị vô hướng. <p>Bài toán trở thành: Tìm cách tô màu cho các đỉnh sao cho 2 đỉnh kề nhau có màu khác nhau, số màu sử dụng là ít nhất</p>		<p>Chỉ ra được đỉnh và cạnh biểu diễn thông tin gì là đạt 0.5đ</p>
(c) Xây dựng các cấu trúc dữ liệu phù hợp nhất có thể để biểu diễn đồ thị trên máy tính theo input đã cho ở Câu a		0.5đ
Cách 1: Dùng danh sách kề	<p>Cách 2: Dùng ma trận kề</p> <ul style="list-style-type: none"> - Giá trị của mỗi ô trong ma trận nếu khác 0 thì cho biết có 	<p>Ví dụ 1, nếu chỉ khai báo cấu trúc, như map<...></p>

<p>Ứng với mỗi đỉnh i, ta cần lưu trữ một tập hợp (danh sách) gồm các đỉnh kề với đỉnh i.</p> <p>Có nhiều cách cài đặt</p> <p>Ví dụ: Cài đặt Danh sách kề dùng map trong STL</p> <p>map<string,set< string >> adj_list</p> <p>Có thể dùng vector, list, tree thay cho set cũng được.</p> <p>A → {B, D}</p> <p>B → {E}</p> <p>Ví dụ: Cài đặt Danh sách kề dùng các danh sách liên kết</p>  <p>vector<list<node>> adj_list và 1 cấu trúc để tra cứu từ chuỗi sang index.</p>	<p>cạnh nối giữa 2 đỉnh tương ứng</p> <p>- Do đỉnh có tên là chuỗi nên cần có cách ánh xạ từ chuỗi sang số nguyên cho biết index dòng/cột tương ứng trong ma trận.</p> <p>Ví dụ: cách ánh xạ từ chuỗi sang index</p> <p>1. map<string, int> name_to_index;</p> <p>hoặc</p> <p>2. dùng mảng lưu các đỉnh (khi muốn biết đỉnh tương ứng với index nào thì phải search)</p>	<p>adj_list hoặc mảng 2 chiều vector<...>, mà không kèm theo mô tả hay chú thích nào khác nhằm cung cấp thêm thông tin về cấu trúc cho người đọc dễ hiểu thì chỉ đạt 0.25 điểm</p> <p>Ví dụ 2, nhiều SV chỉ khai báo mảng 2 chiều mà không diễn giải gì thêm, tuy nhiên sau đó viết đoạn code nhập dữ liệu rồi cập nhật phần tử mảng là 1, vẫn chấm đạt 0.5đ</p> <p>Ghi chú: dạng câu hỏi này đã xuất hiện nhiều lần trong các đề thi cũ.</p>
<p>(d) Cài đặt một (hoặc một số) hàm cho biết tên của quốc gia có đường biên giới chung với nhiều quốc gia khác nhất trong bản đồ</p>		<p>1đ</p>
<p>Các hàm xử lý tìm đỉnh có bậc lớn nhất trong đồ thị</p>		<p>Ghi chú: đây là câu hỏi phân loại SV khá, giỏi nên chấm gắt.</p> <p>- Hàm hoàn chỉnh, trả về đúng tên của quốc gia</p>

		<p>(không phải chỉ trả về index): 1đ</p> <p>-Nhiều lỗi sai, có ý tưởng, xem như mã giả: 0.5đ</p> <p>- Không thể hiện rõ ràng các bước xử lý, dù có viết nhiều câu lệnh cũng không có điểm: 0đ</p>
--	--	--