

ĐỀ 1

(Sinh viên không được sử dụng tài liệu)

HỌ VÀ TÊN SV: .....	<b>CÁN BỘ COI THI</b>	<b>ĐIỂM SỐ</b>
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 (Ví dụ về cách trình bày liên kết: 1-A, 1-B, 1-C, 2-B, 3-C)

---

---

---

---

**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:
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);

```

Trả lời (Ví dụ về cách trình bày: In ra cấu hình của mảng sau mỗi lần hoán vị phần tử)

Mảng ban đầu: 4, 9, 5, 6, 10, 2, 3

_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

**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

(a) Kết quả duyệt RLN của cây  $T_1$ : \_\_\_\_\_

(b) Hình ảnh cây  $T_1$  sau khi xóa 2 lần nút gốc:

**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

(a) Hình ảnh cây  $T_2$  được tạo từ dãy số ban đầu:

(b) Hình ảnh cây  $T_2$  sau khi xoá số 12:

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

**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)		
37						
28						
24						
7						
71						

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

Chỉ số (index)	0	1	2												
Khóa (key)															

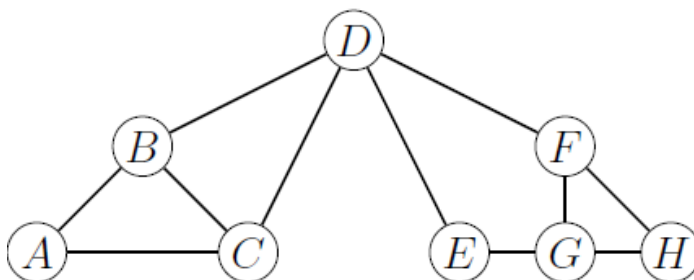
**Câu 6** (1.25 điểm)

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  $G1$  (Hình 2) đã được chuyển đổi dưới đây.



Hình 2: Đồ thị vô hướng  $G1$

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

Kết quả đường đi theo BFS:

---

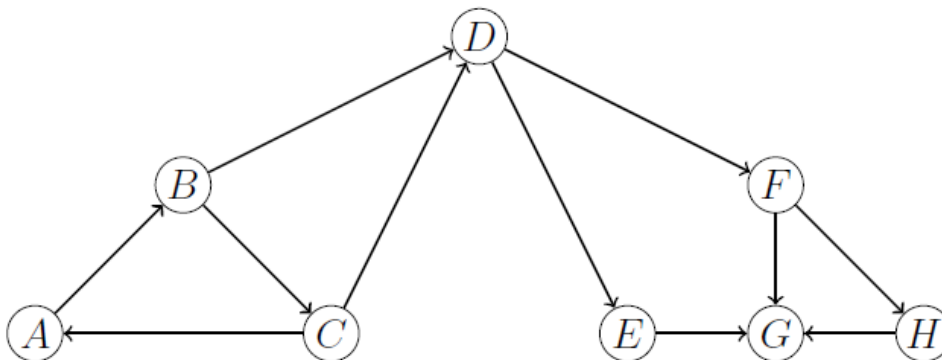
(b) Nếu bạn sử dụng thuật toán tìm kiếm theo chiều sâu (Depth-First Search, DFS) để tìm một đường đi từ đỉnh A đến đỉnh H, hãy **viết lại đường đi kết quả dưới dạng một chuỗi các đỉnh, kèm theo chú thích là bạn chọn cài đặt theo cách thức nào**. Ví dụ, có 2 cách cài đặt DFS phổ biến là sử dụng Ngăn xếp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

Trả lời

Kết quả đường đi theo DFS (sử dụng cách cài đặt-----):

---

(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.



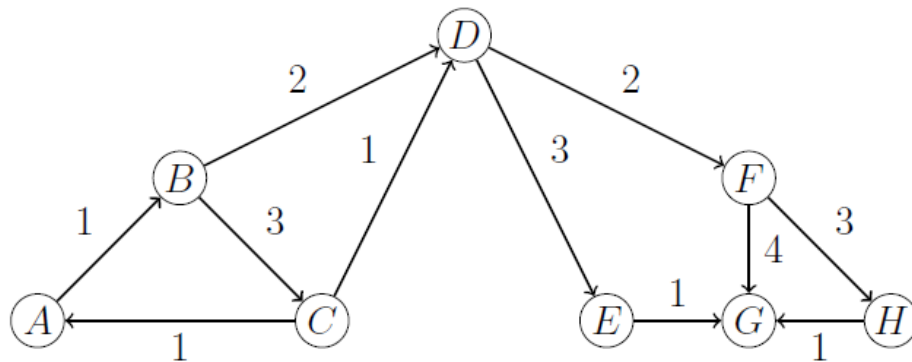
Hình 3: Đồ thị có hướng G2

Trả lời

Kết quả đường đi theo DFS:

---

(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

Thứ tự các đỉnh được loại bỏ khỏi hàng đợi ưu tiên:

---

## 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**.

[illegible]

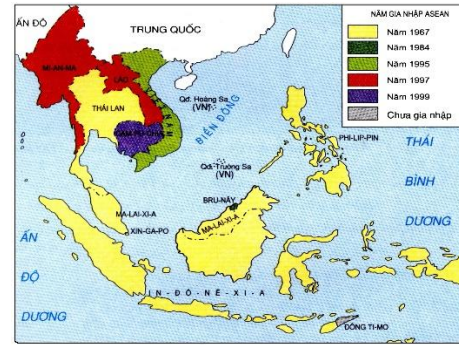
[illegible]



(2.5 điểm)

**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:

- (a) Xác định thông tin đầu vào (input), thông tin đầu ra (output) của bài toán.
- (b) 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.
- (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.

*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.*

- (d) 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

---

---

---

---

---

---

---

---

This image shows a single page of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

This image shows a single page of white paper with horizontal blue or grey ruling lines. The lines are evenly spaced and run across the width of the page, leaving small margins at the top and bottom. There is no handwriting or other markings on the paper.

[illegible]

---HẾT---