TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

ĐỀ THI CUỐI KỲ
HỌC KỲ 2 – NĂM HỌC: 2021-2022

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:

a. Hãy cho biết độ phức tạp của thuật toán Insertion Sort (chèn trực tiếp) theo định nghĩa Big-O (O lớn) *(0.25 điểm)*
b. Viết hàm sắp xếp mảng 1 chiều gồm N phần tử giảm dần với thuật toán Insertion Sort *(0.75 điểm)*
c. Hãy cho biết dãy số sẽ thay đổi qua từng bước như thế nào khi áp dụng thuật toán ở câu 1b, biết rằng dãy số cho như sau: 3, 8, 4, 5, 9, 1, 2, 6 *(1 điểm)*

Đáp án tham khảo:

a. Độ phức tạp của thuật toán Insertion Sort theo định nghĩa Big-O:
   - Trường hợp tốt nhất: $O(n^2)$
   - Trường hợp trung bình: $O(n^2)$
   - Trường hợp xấu nhất: $O(n^2)$

b.
```
void InsertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i – 1;
        for (int j = i – 1; j >= 0 && arr[j] < x; j--) {
            arr[j + i] = arr[j];
            arr[j + 1] = key;
        }
    }
}
```

c. Bước 1: chèn 8

8	3	4	5	9	1	2	6

Bước 2: chèn 4

8	4	3	5	9	1	2	6

Bước 3: chèn 5

8	5	4	3	9	1	2	6

Bước 4: chèn 9

9	8	5	4	3	1	2	6

Bước 5: chèn 1

9	8	5	4	3	1	2	6

Bước 6: chèn 2

9	8	5	4	3	2	1	6

Bước 7: chèn 6

9	8	6	5	4	3	2	1

Câu 2:

Cho dãy ký tự như sau: R, E, T, A, V, X, L, G, S, I

Hãy thực hiện các yêu cầu sau:

a. Vẽ cây nhị phân tìm kiếm bằng cách thêm lần lượt 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)*

b. Cho biết kết quả duyệt cây theo RNL, NRL *(1 điểm)*

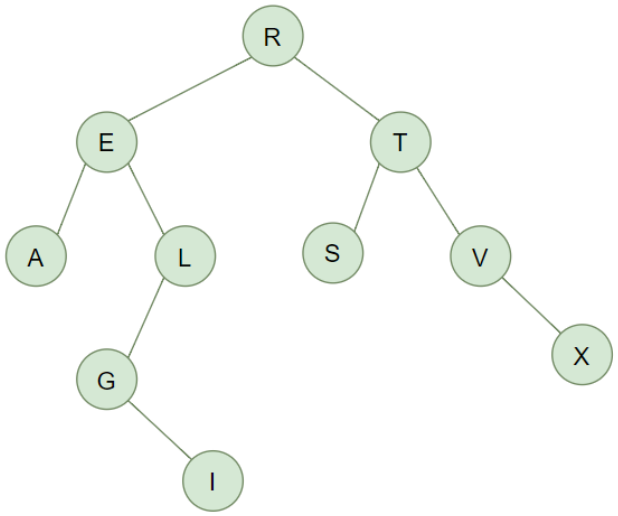
c. Hủy lần lượt từng nút L, T, E, R trên cây, mỗi lần hủy 1 nút vẽ lại cây nối tiếp theo như thứ tự hủy *(1 điểm)*

Đáp án tham khảo:

a. 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.

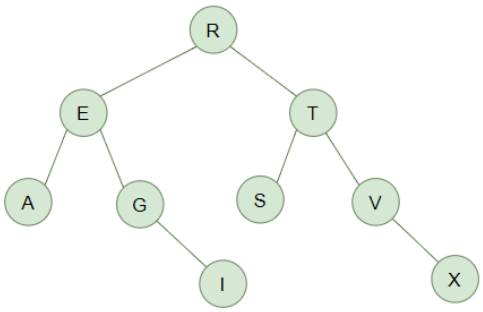
Để vẽ cây nhị phân tìm kiếm bằng các kí tự trên, ta phải xác định được thứ tự của các kí tự trong bảng chữ cái: A < E < G < I < L < R < S < T < V < X

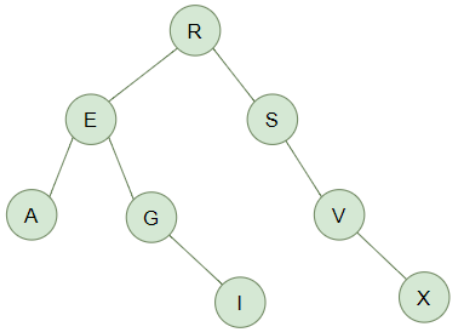
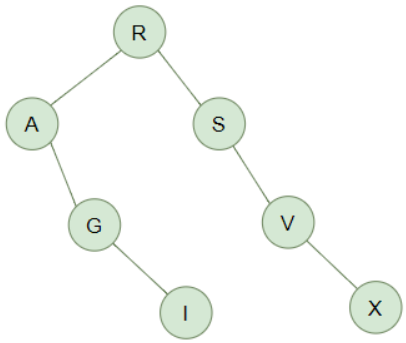
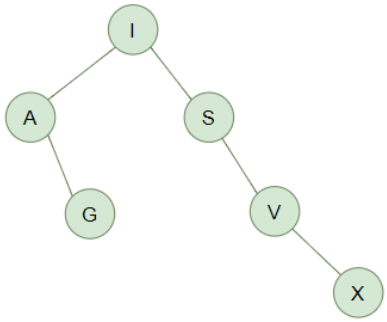


b. Duyệt cây theo RNL: X, V, T, S, R, L, I, G, E, A

Duyệt cây theo NRL: R, T, V, X, S, E, L, G, I, A

c. Hủy lần lượt từng nút L, T, E, R trên cây:

Nút cần xóa	Dạng nút	Vẽ lại cây
L	nút có 1 cây con	

T	nút có đầy đủ 2 cây con (chọn nút thế mạng là nút S)	
E	nút có đầy đủ 2 cây con (chọn nút thế mạng là nút A)	
R	nút có đầy đủ 2 cây con (chọn nút thế mạng là nút I)	

Câu 3:

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.

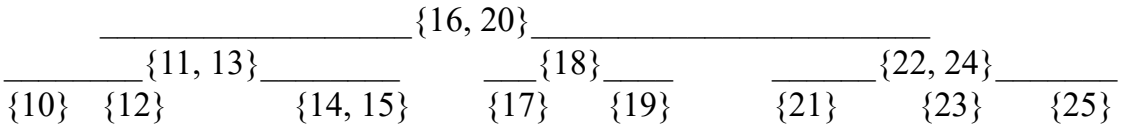
Hãy thực hiện các yêu cầu sau:

3.1 Cho dãy số: 12, 17, 20, 23, 15, 11, 24, 13, 19, 22, 18, 21, 16. Hỏi khi lần lượt thêm các số trong dãy theo thứ tự từ trái qua phải vào một cây B-Tree bậc 3 rỗng thì:

a. Các khóa nào khi thêm vào cây sẽ làm phát sinh thao tác tách (split) node? *(0.5 điểm)*

b. 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:



c. Hãy lần lượt tiến hành xóa các khóa sau khỏi cây: 13, 24, 19 và vẽ câu B-Tree trước và sau khi xóa mỗi khóa trên *(0.5 điểm)*

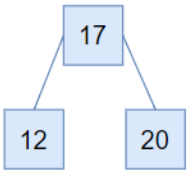
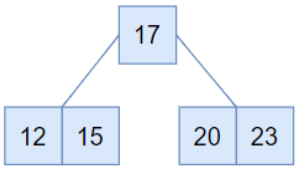
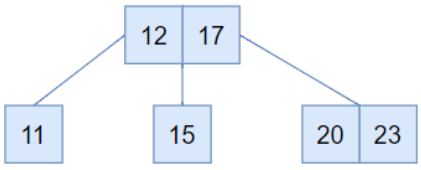
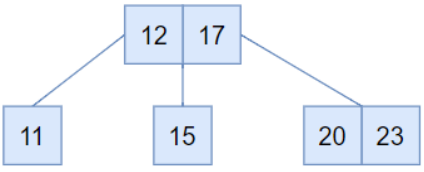
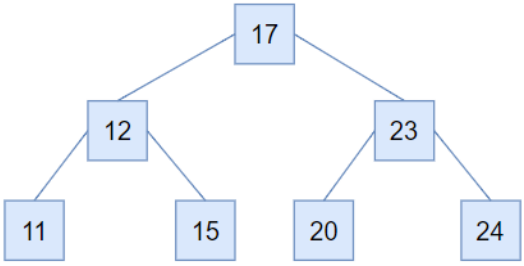
Lưu ý khi xóa:

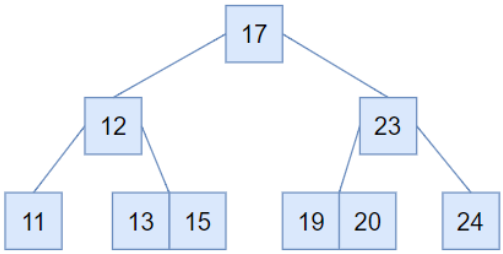
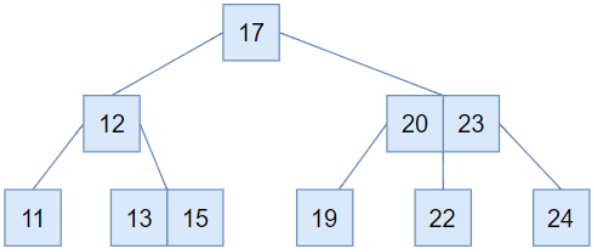
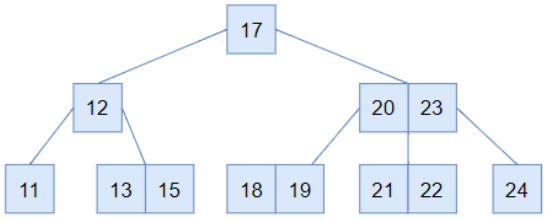
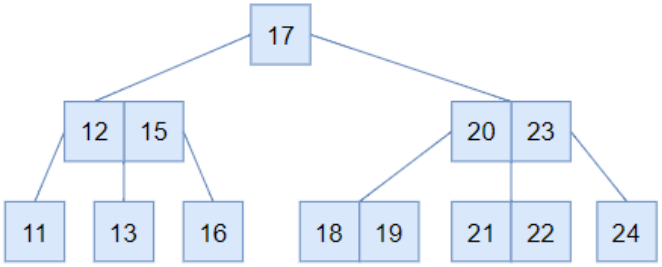
- Khi khóa cần xóa (gọi là x) không nằm ở node lá, chọn khóa thế mạng là khóa có giá trị lớn nhất mà nhỏ hơn x.
- Thao tác nhường khóa (underflow) sẽ được thực hiện khi hai node liền kề có tổng số khóa >= 2. Khi có một node không còn đáp ứng đủ số lượng khóa tối tiểu, ưu tiên thực hiện underflow thay 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*:**

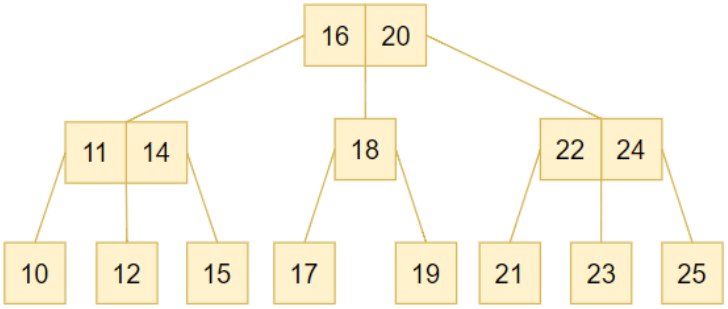
a. Các khóa làm phát sinh thao tác slipt node: 20, 11, 24, 22, 16

b.

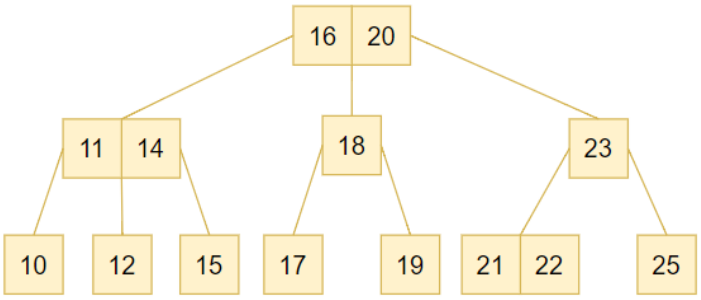
Khóa	Trước khi thêm	Sau khi thêm
20		
11		
24		

22		
16		

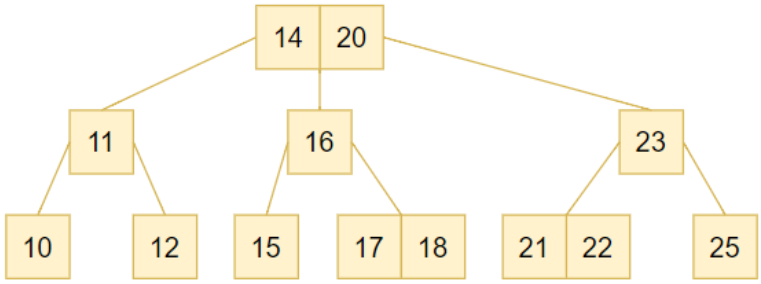
c. ***Xóa 13:*** swap 13 và 12 sau đó xóa. Ưu tiên thực hiện underflow giữa node {14, 15} và node vừa bị thiếu khóa thay vì catenate nó với node {10}.



Xóa 24: swap 24 với 23 và thực hiện catenate, ưu tiên catenate với {21} thay vì {25}.



Xóa 19: trực tiếp tại lá sau đó xóa và catenate node bị xóa với {17} làm {18} bị thiếu khóa và lan truyền xử lý này lên khiến node {11, 14} phải underflow cho node bị thiếu khóa.



Câu 4:

Để việc tìm kiếm thông tin mặt hàng được nhanh chóng, người ta dùng một bảng băm theo phương pháp thăm dò, làm việc trên mã quản lý của mặt hàng. Mã quản lý này là một con số nguyên. Bảng băm có:

- Hàm băm: $h(key) = (key \% M)$
- Hàm băm lại (hàm thăm dò): $prob(key, i) = (h(key) + i*i + i) \% M$

Trong đó:

- key là giá trị khóa.
- i là một số nguyên cho biết lần băm lại (thăm dò) thứ i.
- M là kích thước bảng băm.

Giả sử M = 7, cho trường hợp ***T*** của bảng băm đã chứa dữ liệu như bên dưới. Biết "-" là ký hiệu vị trí trống trong bảng băm.

Bảng băm ***T***

0	-
1	-
2	16
3	-
4	-
5	12
6	13

a. Trình bày từng bước việc tìm mã quản lý 23 trong bảng băm ***T***. *(0.5 điểm)*

b. Trình bày từng bước việc thêm các mã quản lý sau vào bảng băm ***T*** theo đúng thứ tự liệt kê là 11, 20, 27 *(1.5 điểm)*.

Đáp án tham khảo:

a. h(23) = 23 mod 7 = 2. Vị trí 2 đang chứa "16" → băm lại
Băm lại lần 1: prob(23, 1) = (2 + 1.1 + 1) mod 7 = 4. Vị trí 4 đang chứa "-" (rỗng) → không tìm thấy mã quản lý 23.

b. ***Thêm 11:*** h(11) = 11 mod 7 = 4. Vị trí 4 còn trống, thêm 11 vào vị trí 4.

Bảng băm ***T***

0	-
1	-
2	16
3	-
4	11
5	12
6	13

Thêm 20: h(20) = 20 mod 7 = 6. Vị trí 6 đang chứa '13' → xảy ra đụng độ

Băm lại lần 1: h(20, 1) = (6 + 1.1 + 1) mod 7 = 1. Vị trí 1 còn trống, thêm 20 vào vị trí 1.

Bảng băm ***T***

0	-
1	20
2	16
3	-
4	11
5	12
6	13

Thêm 27: h(27) = 27 mod 7 = 6. Vị trí 6 đang chứa '13' → xảy ra đụng độ
Băm lại:
- Lần 1: h(27, 1) = (6 + 1.1 + 1) mod 7 = 1. Vị trí 1 chứa '20' → xảy ra đụng độ.
- Lần 2: h(27, 2) = (6 + 2.2 + 2) mod 7 = 5. Vị trí 5 chứa '12' → xảy ra đụng độ.
- Lần 3: h(27, 3) = (6 + 3.3 + 3) mod 7 = 4. Vị trí 4 chứa '11' → xảy ra đụng độ.
- Lần 4: h(27, 4) = (6 + 4.4 + 4) mod 7 = 5. Vị trí 5 chứa '12' → xảy ra đụng độ.
- Lần 5: h(27, 5) = (6 + 5.5 + 5) mod 7 = 1. Vị trí 1 chứa '20' → xảy ra đụng độ.
- Lần 6: h(27, 6) = (6 + 6.6 + 6) mod 7 = 6. Vị trí 6 chứa '13' → xảy ra đụng độ.

Đã băm lại M – 1 = 6 lần, kết thúc việc băm lại → không tìm được chỗ trống để thêm '25'.

Câu 5:

Trong các ứng dụng thực tế, chẳng hạn trong mạng lưới giao thông đường bộ, đường thủy hoặc đường hàng không, người ta không chỉ quan tâm đến việc tìm đường đi giữa hai địa điểm mà còn phải lựa chọn một hành trình tiết kiệm nhất (theo tiêu chuẩn không gian, thời gian hay chi phí). Vấn đề này có thể được mô hình hóa thành một bài toán trên đồ thị, trong đó mỗi địa điểm được biểu diễn bởi một đỉnh, cạnh nối hai đỉnh biểu diễn cho "đường đi trực tiếp" giữa hai địa điểm (tức không đi qua địa điểm trung gian) và trọng số của cạnh là khoảng cách giữa hai địa điểm.

Bài toán có thể phát biểu dưới dạng tổng quát như sau: Cho một đơn đồ thị có hướng và có trọng số dương G = (V, E), trong đó V là tập đỉnh, E là tập cạnh (cung) và các cạnh đều có trọng số, hãy tìm một đường đi (không có đỉnh lặp lại) ngắn nhất từ đỉnh xuất phát S thuộc V đến đỉnh đích F thuộc V.

Giả sử thông tin đầu vào của bài toán (Input) được nhập vào chương trình như sau:

Input	**Giải thích**
7 A B 1 B E 3 E D 3 C B 4 A D 7 E C 2 C D 1 A E	- Dòng đầu tiên chứa một số nguyên dương ***e*** cho biết số cạnh của đồ thị - Với ***e*** dòng tiếp theo, mỗi dòng chứa hai chuỗi ***u, i*** và một số nguyên dương **x,** thể hiện thông tin có một cạnh nối từ đỉnh ***u*** sang đỉnh ***i*** trong đồ thị với độ dài (trọng số) là x - Dòng cuối cùng chứa hai chuỗi s và f, đây là đỉnh bắt đầu và đỉnh kết thúc của đường đi cần tìm **Lưu ý:** không biết trước số đỉnh và danh sách các đỉnh.

Hãy thực hiện các yêu cầu sau:

a. Xây dựng các cấu trúc dữ liệu phù hợp nhất có thể để biểu diễn đồ thị trên máy tính theo input đã cho. *(0.5 điểm)*
   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.
b. Viết hàm nhập theo Input ở đầu bài 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, map<string, int>> DIEM;
b.
```
void Input() {
        int e;
        for (int i = 0; i < e; i++) {
                string u, i;
                cin >> u >> i;
                int x;
                cin >> x;
                DIEM[u][v] = x;
                DIEM[v][u] = x;
        }
}
```

***** KHÔNG YÊU CẦU tìm cách giải cho bài toán này. 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.**

HẾT