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

Đề 1

ĐỀ THI CUỐI HỌC KỲ II (2023-2024)
MÔN: Cấu trúc dữ liệu & Giải thuật
MÃ LỚP: IT003
Thời gian: 90 phút

(***Sinh viên không được sử dụng tài liệu***)

HỌ VÀ TÊN SV: …… **MSSV**: …… **STT**: …… **PHÒNG THI:…..**……	**CÁN BỘ COI THI**

CÂU HỎI ĐIỀN KHUYẾT

Câu 1 (0.5 điểm)

Xét độ phức tạp theo BigO, thuật toán tìm nhị phân tốt hơn thuật toán tìm tuyến tính, vì sao? (0.25đ) .....

.....

.....

.....

Thuật toán tìm nhị phân với thứ tự tăng dần sẽ có kết quả tìm khóa 4 trong mảng gồm các khóa 1, 2, 3, 4, 5, 6, 7, 8, 9 sau bao nhiêu lần **so sánh giá trị khóa**? (0.25đ)

.....

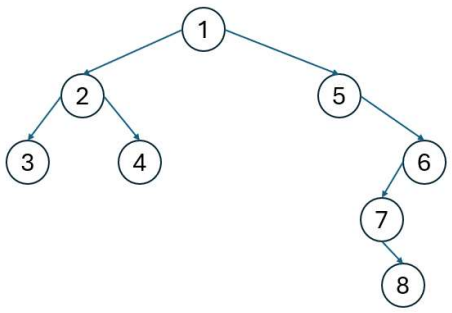
Câu 2 (0.5 điểm)

Nếu dùng thuật toán Insertion Sort (Chèn trực tiếp) để sắp xếp một dãy số tăng dần, thì khi hoàn thành việc sắp xếp mảng A={3,2,4,5,1,5}, số lần dịch chuyển phần tử sang phải một vị trí là bao nhiêu?

.....

Câu 3 (0.5 điểm)

Cho cây nhị phân T_1 như Hình 1.



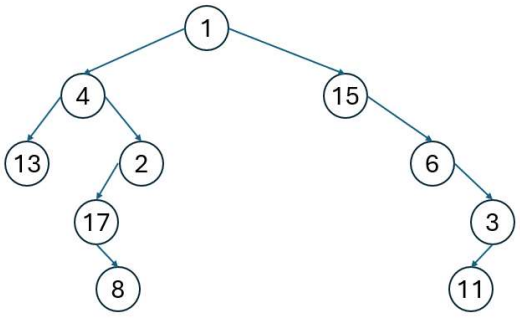
Hình 1. Cây nhị phân T_1

Hỏi khóa 7 xuất hiện ở vị trí thứ mấy (vị trí đầu tiên là 0) trong kết quả duyệt LNR?

.....

Câu 4 (0.5 điểm)

Cho cây nhị phân T_2 như Hình 2.



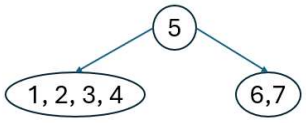
Hình 2. Cây nhị phân T_2

Hỏi mức nào trên cây T_2 có tổng giá trị khóa cao nhất?

.....

Câu 5 (0.5 điểm)

Cho B-Tree bậc 5, T_3, như Hình 3.

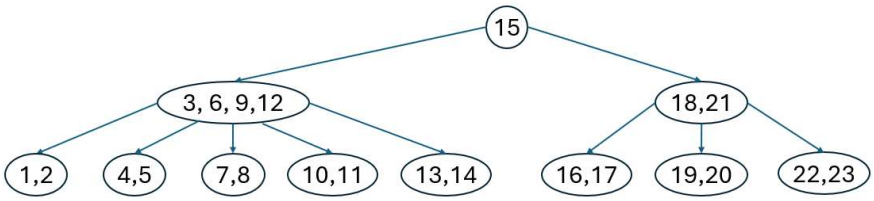


Hình 3. B-Tree bậc 5 T_3

Hãy vẽ cây T_3 sau khi xóa khóa 6 (0.25đ)? Và cho biết tên thao tác đã điều chỉnh các nút trên T_3 (0.25đ)?

Câu 6 (0.5 điểm)

Cho B-Tree bậc 5, T_4, như Hình 4. Biết thao tác tìm khóa trên một nút sử dụng thuật toán tìm tuyến tính.



Hình 4. B-Tree bậc 5 T_4

Hỏi số lần **so sánh giá trị khóa** khi có kết quả tìm khóa 14 trên T_4?

.....

Đề bài bên dưới được sử dụng cho Câu 7, Câu 8 và Câu 9.

Cho bảng băm địa chỉ mở HT_1, có kích thước M=11, hàm băm f và hàm băm lại f' lần lượt là:

$f(key) = key \,\%\, M$

$f'(key, i) = [f(key) + i^2] \,\%\, M$

Trong đó, phép toán % là phép toán lấy phần dư của phép chia nguyên.

Bảng băm HT_1

Chỉ số	0	1	2	3	4	5	6	7	8	9	10
Khóa	11	1			15	5				9	

Câu 7 (0.5 điểm)

Hỏi số lần **so sánh giá trị khóa** khi có kết quả tìm khóa 4 trên HT_1?

.....

Câu 8 (0.5 điểm)

Hỏi hệ số tải của HT_1 (lấy 2 số lẻ sau phần thập phân)?

.....

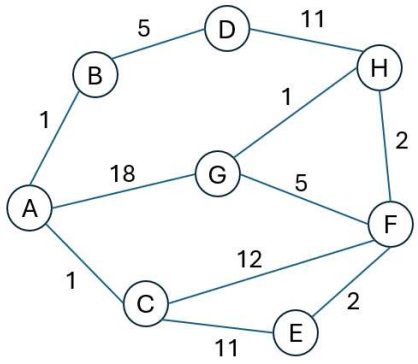
Câu 9 (0.5 điểm)

Hỏi vị trí khi thêm khóa 22?

.....

Câu 10 (0.5 điểm)

Cho đồ thị vô hướng G như Hình 5.



Hình 5. Đồ thị vô hướng G

Hỏi độ dài đường đi ngắn nhất từ A đến các đỉnh còn lại?

Đỉnh	B	C	D	E	F	G	H
Độ dài đường đi ngắn nhất từ A							

CÂU HỎI TỰ LUẬN

Giả sử các đoạn chương trình trong phần này đều có các chỉ thị #include, đảm bảo các kiểu dữ liệu và các hàm được C++ hỗ trợ đều sẵn sàng để sử dụng.

Câu 11 (2 điểm)

Cho các bản đồ thể hiện không quá 10000 giao lộ (ngã ba, ngã tư, ..). Mỗi giao lộ được đánh số từ 0 đến 9999. Các bản đồ này có thể biểu diễn theo đồ thị đơn, vô hướng và được biểu diễn trên máy tính bằng ma trận, với kiểu dữ liệu ***Bando*** như sau:

```
struct Bando {
    int N;
    bool Duong[10000][10000];
};
```

Trong đó, ***N*** là số giao lộ; ***Duong***[***i***][***j***]=true cho biết giao lộ ***i*** có đường đi trực tiếp đến giao lộ ***j***; ***Duong***[***i***][***j***]=false cho biết giao lộ ***i*** không có đường đi trực tiếp đến giao lộ ***j***.

1) Cài đặt hàm ***DemNgaNam*** để đếm số ngã năm trên bản đồ ***m*** (1đ):

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

2) Cài đặt hàm ***CoDuongdi*** trả về ***1*** nếu bản đồ ***m*** có đường đi từ giao lộ ***s*** đến giao lộ ***e***, trả về ***0*** trong các trường hợp còn lại. Yêu cầu phải dùng queue thay cho hàm đệ quy (1đ):

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Câu 12 (1 điểm)

Cho kiểu dữ liệu ***DblStack***, là một stack chứa các giá trị double, và một hàm ***CreateDblItem*** được khai báo như bên dưới.

```
struct DblStack {
    double data;
    DblStack *next;
};
DblStack * CreateDblItem(double x) {
    DblStack r = new DblStack;
    if (r == NULL) exit(1);
    r->data = x;
    r->next = NULL;
    return r;
};
```

Hãy cài đặt hàm ***push*** (0.5đ) và hàm ***pop*** (0.5đ) ở bên dưới, lần lượt để đưa một số double vào stack và xóa bỏ phần tử ở đỉnh stack.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Câu 13 (1 điểm)

Cho kiểu dữ liệu ***TREE***, để biểu diễn cây nhị phân tìm kiếm, được khai báo như bên dưới. Hãy cài đặt hàm ***ChonKhoa*** để trả về một danh sách các khóa ***k*** trên cây nhị phân tìm kiếm sao cho $a < k < b$, với ***a***, ***b*** là các số cho trước. Yêu cầu dùng queue thay cho hàm đệ quy.

```
struct TNode {
    int key;
    TNode *left, *right;
};
typedef TNode *TREE;
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Câu 14 (1 điểm)

Quỳnh được giao công việc cài đặt một hàm nhận một danh sách số nguyên và trả về một tập hợp số nguyên (các phần tử không trùng nhau). Các phần tử trong tập hợp được sắp thứ tự tăng dần.

Ví dụ: Input là 1, 5, 2, 7, 4, 1, 3, 2, 3, 5 thì output là 1, 2, 3, 4, 5, 7.

Quỳnh sử dụng hàm ***sort***, có **độ phức tạp O(nlogn)**, và cài đặt hàm ***ListToSet*** như sau.

```
vector<int> ListToSet(vector<int> A) {
    vector<int> r;
    long long unsigned int i, j;
    for (i = 0; i < A.size(); i++) {
        for (j = 0; j < i; j++) {
            if (A[j] == A[i]) break;
        }
        if (j >= i) r.push_back(A[i]);
    }
    sort(r.begin(), r.end());
    return r;
}
```

Mentor của Quỳnh nói rằng, cách này chưa tốt, phải sắp xếp trước rồi xử lý. Mentor đưa cho Quỳnh hàm ***List2Set*** như bên dưới, và yêu cầu Quỳnh hoàn thành nó sao cho thời gian thực hiện tốt hơn hàm ***ListToSet***.

Bạn cũng phải hoàn thành hàm ***List2Set*** này.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....