

(Sinh viên không được sử dụng tài liệu)

HỌ VÀ TÊN SV: MSSV: STT: PHÒNG THI:.....	<u>CÁN BỘ COI THI</u>
---	-----------------------

CÂU HỎI TỰ LUẬN

Câu 1 (1.5 điểm) (CLO1, CLO2)

1.1 Các thuật toán sắp xếp nào, trong số các thuật toán insertion sort, heap sort, merge sort và quick sort, được thiết kế theo chiến lược chia để trị? (0.5 điểm)

Đáp án tham khảo:

có các phương án trả lời bao gồm

- merge sort, quick sort : 0.5 điểm
- merge sort: 0.25 điểm
- quick sort: 0.25 điểm
- các phương án khác không có điểm.

1.2 Cho dãy số $A = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Hãy cho biết:

- Những thuật toán sắp xếp nào, trong số các thuật toán heap sort, quick sort và merge sort, sẽ gặp thuận lợi khi sắp dãy số A theo thứ tự giảm dần? (0.5 điểm)
- Thuận lợi đó, theo từng thuật toán đã trả lời ở câu 1.2 a, là gì? (0.5 điểm)

Đáp án tham khảo:

a) Heap sort: 0.25 điểm, quick sort 0.25 điểm

b) Heap sort vì đã có sẵn heap min 0.25 điểm. Quick sort vì luôn chọn được phần tử trung vị hoặc gần trung vị khi lấy pivot là phần tử giữa mảng 0.25 điểm

Câu 2 (2 điểm) (CLO2, CLO3)

Telegram đã ra đời được 10 năm và được biết đến như là một dịch vụ nhắn tin tức thời miễn phí, đa nền tảng và mã hóa. Màn hình của dịch vụ này chỉ hiển thị tin nhắn của k người sau cùng. Khi có một tin nhắn được gửi đến bạn thì tin nhắn sẽ được hiển thị vào đầu danh sách tin nhắn. Các tin nhắn sẽ được nhóm lại theo số điện thoại (SĐT) của người gửi.

Cho một chuỗi tuần tự các SĐT có gửi tin nhắn cho bạn:

2.1 Hãy mô tả cấu trúc dữ liệu (CTDL) sử dụng để hiển thị tin nhắn của k người sau cùng và các tin nhắn sẽ được nhóm lại theo SĐT của người gửi (xem mô tả ví dụ input, output bên dưới) (0.5 điểm)

Đáp án tham khảo:

Sinh viên trình bày được ý tưởng cơ bản sử dụng ngăn xếp (stack) / queue / mảng. - không cần nêu chi tiết giải thuật - thì được 0.5 điểm, cụ thể:

- Nếu sử dụng Stack: Các SĐT được PUSH_BACK vào stack và lấy TOP/POP ra (xem code minh họa bên dưới)

- Nếu sử dụng Queue: Các SĐT được PUSH_BACK vào queue và lấy FRONT ra
- Nếu sử dụng Mảng: SV tự lưu 2 con trỏ hoặc dồn mảng để thực hiện công việc theo yêu cầu

2.2 Viết chương trình bằng C++ hiện thực hoá yêu cầu, sử dụng CTDL trong câu 2.1 (1.5 điểm)

Ví dụ Input:

- Dòng 1: số k
- Dòng 2: số n: số lượng tin nhắn được gửi đến bạn
- Dòng 3: danh sách SĐT của n tin nhắn theo thứ tự thời gian từ trước đến sau

Lưu ý: Giới hạn SĐT chỉ có tối đa 3 chữ số

Ví dụ Output: Danh sách các SĐT của các tin nhắn hiển thị trên màn hình theo thứ tự SĐT có tin nhắn mới hơn sẽ xuất hiện trước kèm theo số lượng tin nhắn từ SĐT đó.

Input	Output
5 12 903 901 902 904 976 976 973 986 976 904 905 986	986(2) 905 904(2) 976(3) 973

Đáp án tham khảo:

- Nếu chương trình xuất ra được đúng như Output yêu cầu thì được 1.5 điểm
- Nếu chương trình xuất ra nhiều hơn k số điện thoại (nhiều hơn so với Output) nhưng vẫn bao gồm Output và có thứ tự như Output thì được 1 điểm

Ví dụ : 986(2) 905 904(2) 976(3) 973, 902,...

- Nếu chương trình xuất ra đủ k số điện thoại như Output, nhưng số tin nhắn bị thiếu thì được 1 điểm

Ví dụ : 986(1) 905 904(1) 976(2) 973

- Nếu chương trình xuất kết quả đủ k số điện thoại, nhưng không theo thứ tự như Output hoặc không đủ k số điện thoại thì không tính điểm

Minh hoạ đáp án sử dụng stack:

```
#include<iostream>
#include<stack>
using namespace std;

int main() {
    stack<int> s;
    int SL[1000]={0};
    int n, k,x;
    cin>>k;
    cin>>n;
    for (int i = 0; i < n; i++){
        cin>>x;
        s.push(x);
        SL[x]++;
    }

    while ((k) && !s.empty()){
        x = s.top();
        if (SL[x]!=0){
            cout<<x;
            if (SL[x]!=1)
                cout<<" ("<<SL[x]<<") ";
            cout<<" ";
        }
    }
```

```

        k--;
        SL[x] = 0;
    }
    s.pop();
}
return 0;
}

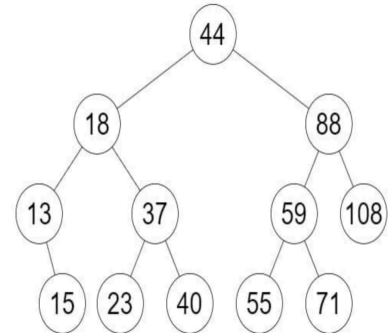
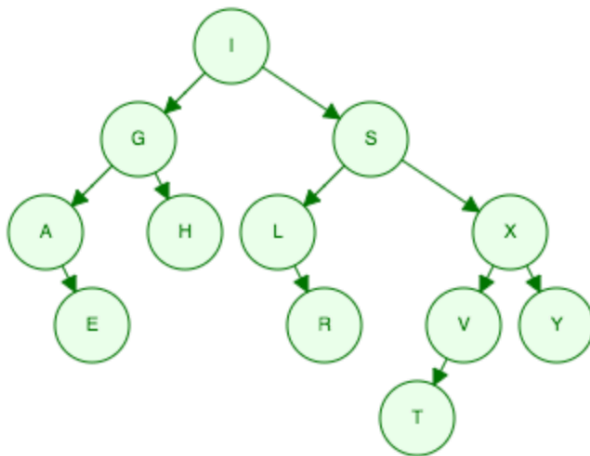
```

Câu 3 (1.5 điểm) (CLO1, CLO3)

3.1 Cho dãy ký tự như sau: I, S, G, L, X, V, A, T, E, R, H, Y, hãy vẽ cây nhị phân tìm kiếm khi thêm từng ký tự vào cây theo thứ tự từ trái qua phải của dãy ký tự, biết rằng giá trị của từng ký tự tương ứng theo thứ tự xuất hiện của ký tự trong từ điển như sau: 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 (0.5 điểm)

Đáp án tham khảo:

Vẽ đúng cây nhị phân tìm kiếm : 0.5 điểm



3.2 Cho biết kết quả duyệt cây nhị phân tìm kiếm (hình bên) theo RNL, NRL (0.5 điểm)

Đáp án tham khảo:

Viết đúng duyệt cây theo RNL : 0.25 điểm

Viết đúng duyệt cây theo NRL : 0.25 điểm

RNL : 108,88,71,59,55,44,40,37,23,18,15,13

NRL : 44,88, 108,59,71,55,18,37,40,23,13,15

3.3 Viết hàm đếm số nút có 2 nút con trên cây nhị phân tìm kiếm (0.5 điểm)

Đáp án tham khảo:

Viết đúng hàm đếm :0.5 điểm

```

int Dem(Tree c)
{
    if (c!=NULL)
    {
        int a = Dem(c->pLeft);
        int b = Dem(c->pRight);
        if (c->pLeft != NULL && c->pRight != NULL)
            return 1 + a + b;
        return a + b;
    }
}

```

```

    }
    return 0;
}

```

Câu 4 (1.5 điểm) (CLO1)

Cho biết B-Tree bậc 5 là một cây đa nhánh thỏa mãn đồng thời tất cả các tính chất sau:

- Tất cả node lá phải nằm trên cùng một mức
- Tất cả các node, trừ node gốc và node lá có tối thiểu là 2 khóa và 3 con.
- Tất cả các node có tối đa là 4 khóa và 5 con
- Một node không phải là lá và có n khóa thì bắt buộc phải có n+1 con.

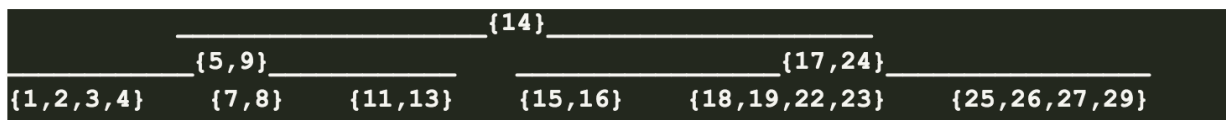
Lần lượt thêm các giá trị sau đây vào cây: 9, 8, 23, 2, 14, 17, 27, 11, 1, 24, 16, 5, 7, 13, 4, 18, 25, 19, 22, 26, 15, 29, 3.

4.1 Vẽ trạng thái của cây sau khi thêm toàn bộ các giá trị trên. (1 điểm)

Đáp án tham khảo:

Đề đã quy định rất rõ các ràng buộc về cấu trúc của cây nên chỉ có thể có 1 đáp án. Chiến lược preemptive splitting hay proactive splitting sẽ không áp dụng được trong trường hợp này.

Cây sau khi thêm tất cả các node sẽ có hình dạng như sau:



Sinh viên vẽ đúng hoàn toàn, không sai sót: được 1 điểm

Sinh viên ghi sai hoặc thiếu 1 số ở 1 node lá bất kỳ: được 0.5 điểm

Sai các node không phải lá hoặc sai nhiều hơn 1 số: 0 điểm

4.2 Cho biết cấu trúc BTree_node và một hàm duyệt cây được định nghĩa như sau:

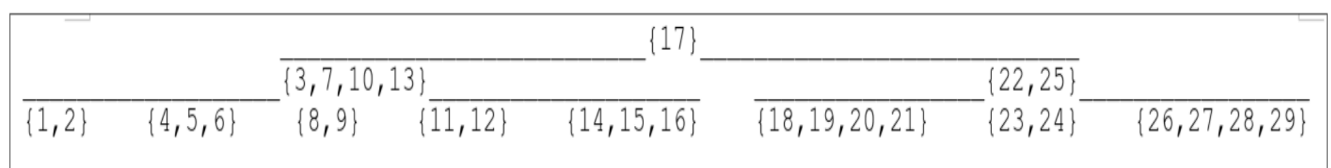
```

class BTree_node{
    vector<int> keys;
    vector<BTree_node*> children;
};

void dfs2(BTree_node* root){
    stack<BTree_node*> q;
    q.push(root);
    while(q.empty() != true){
        BTree_node* x = q.top(); q.pop();
        for (BTree_node* i : x->keys) cout << i << " ";
        cout << endl;
        for(int i: x->children) q.push(i);
    }
}

```

Với cây B-Tree bậc 5 có hình dáng như bên dưới thì hàm duyệt cây trên sẽ cho ra output là gì? (0.5 điểm)



Đáp án tham khảo:

Trường hợp 1:

Sinh viên nhận ra đây là thuật toán duyệt cây theo chiều sâu (phần mã nguồn cũng ghi rõ tên thuật toán là DFS, không đánh đổ) Khi duyệt mỗi node trên cây sẽ xuất ra trên một hàng, thứ tự thêm các node con vào stack cũng như thứ tự in ra các khóa trên mỗi hàng giống với quy ước chung của B-Tree.

Đề thi chỉ có một đáp án:

```
17
22 25
26 27 28 29
23 24
18 19 20 21
3 7 10 13
14 15 16
11 12
8 9
4 5 6
1 2
```

Sinh viên làm đúng hoàn toàn được 0.5 điểm

Sinh viên ghi sai hoặc thiếu không quá 1 số ở 1 hàng bất kỳ được 0.25đ

Các trường hợp sai nhiều hơn: 0 điểm

Trường hợp 2:

Sinh viên ghi rõ được là do class BTree_node khai báo các biến thành viên có access modifiers là private nên hàm dfs2 không phải là thành viên của class sẽ không thể truy cập được biến, dẫn đến lỗi biên dịch. Sau đó không tiến hành làm phần duyệt cây. Do hàm duyệt và cấu trúc node được cung cấp chỉ là minh họa cho thuật toán duyệt chứ không phải chương trình hoàn chỉnh nên sinh viên vẫn phải thực hiện việc duyệt cây mới được đầy đủ điểm câu này

Trường hợp này chỉ được 0.25đ.

Câu 5 (1.5 điểm) (CLO1)

5.1 Cho bảng băm T có các đặc điểm sau:

- Kích thước bảng băm $m=7$.
- Giá trị khóa k là các số nguyên $k \in [1, 50]$.
- Sử dụng hàm băm theo phép chia.
- Xử lý đụng độ (hay va chạm, collision) theo phương pháp thăm dò tuyến tính.

Hãy cho biết công thức của hàm băm $h(k)$ và hàm băm lại $h_i(k)$, hay $h(k, i)$, sao cho xác suất xảy ra đụng độ trong T không vượt quá n/m khi T có n phần tử. (0.5 điểm)

Đáp án tham khảo:

Theo định lý họ hàm băm phổ quát thì hàm băm $h(k) = ((a*k + b) \bmod p) \bmod m$ sẽ có xác suất xảy ra đụng độ không vượt quá n/m khi bảng băm có n phần tử. Trong đó: a là số tự nhiên khác 0, b là số tự nhiên, p là số nguyên tố lớn hơn giá trị lớn nhất của k .

Sinh viên có thể chọn số a, b, p bất kỳ, chẳng hạn:

$$h(k) = ((2*k + 1) \bmod 53) \bmod 7$$

hay

$$h(k) = ((2*k + 1) \% 53) \% 7$$

Trong trường hợp chọn $a = 1$, $b=0$ thì có thể chọn hàm băm:

$$h(k) = k \bmod m$$

hay

$$h(k) = k \% m$$

Nên đáp án tham khảo:

- Nếu sinh viên ghi: Theo định lý họ hàm băm phổ quát thì $h(k) = \dots$. Hàm $h(k)$ đúng dạng (0.25đ). Nếu không nhắc đến định lý họ hàm băm phổ quát thì không tính điểm.
- Hàm băm lại : $h(k, i) = (h(k) + i) \bmod 7$ (0.25đ). Câu này chỉ kiểm tra dạng thăm dò tuyến tính nên nếu $h(k)$ ở bước trước sai cũng không ảnh hưởng đến điểm câu này.

5.2 Cho bảng băm T có các đặc điểm sau:

- Giá trị khóa k là một số nguyên dương.
- Kích thước bảng băm $m=11$.
- Hàm băm $h(k) = \text{floor}(((0.618 * k) \bmod 1) * m)$.
- Hàm băm lại $h(k, i) = \text{floor}(((0.618 * (k + i)) \bmod 1) * m)$.

Cho biết :

- Hàm ***floor(x)*** trả về giá trị nguyên lớn nhất nhỏ hơn hoặc bằng x . Ví dụ $\text{floor}(4.8) = 4$.
- Biểu thức ***x mod 1*** sẽ lấy phần thập phân của x . Ví dụ, $8.492 \bmod 1 = 0.492$.
- Các số thực chỉ lấy đến ba số sau phần thập phân.

Giả sử bảng băm T đã chứa các khóa như sau:

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

Hãy trình bày quá trình:

- Thêm khóa có giá trị 1 vào T. (0.5 điểm)
- Giả sử khóa 2 đã bị xóa (ô có chỉ số 2 trên T chứa ký hiệu DELETED). Tìm khóa có giá trị 1 trên T. (0.5 điểm)

Đáp án tham khảo:

a)

- $h(1) = 6$, $T[6] = 9$ đụng độ \rightarrow băm lại
- $h(1,1) = 2$, $T[2] = 2$ đụng độ \rightarrow băm lại. (đúng tới bước này chấm 0.25 điểm)
- $h(1,2) = 9$, $T[9]$ còn trống \rightarrow gán $T[9] = 1$. (0.25 điểm)

b)

- $h(1) = 6$, $T[6] = 9$ đụng độ \rightarrow băm lại
- $h(1,1) = 2$, $T[2] = \text{DELETED}$ đụng độ \rightarrow băm lại. (đúng tới bước này chấm 0.25 điểm)
- $h(1,2) = 9$, $T[9] = 1 \rightarrow$ Tìm thấy. (0.25 điểm)

Lưu ý : Trường hợp sinh viên làm câu b) không theo kết quả câu a):

- $h(1) = 6$, $T[6] = 9$ đụng độ \rightarrow băm lại

- $h(1,1) = 2$, $T[2] = 2$ đụng độ \rightarrow bấm lại. (đúng tới bước này chấm 0.25 điểm)
- $h(1,2) = 9$, $T[9]$ là ô trống \rightarrow Không tìm thấy. (0.25 điểm)

Câu 6 (2 điểm) (CLO2, CLO3)

Bài toán tìm đường đi ngắn nhất giữa hai đỉnh trên đồ thị có thể được 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 và các cạnh đều có trọng số, hãy tìm một đường đi ngắn nhất (không có đỉnh lặp lại) từ đỉnh xuất phát s thuộc V đến đỉnh đích g thuộc V . Đường đi ngắn nhất giữa hai đỉnh có tổng các trọng số của các cạnh tạo nên đường đi đó là nhỏ nhất.

Giả sử thông tin đầu vào của bài toán (Input) được nhập vào chương trình và kết quả đầu ra (Output) bao gồm:

Input	Giải thích
8 10 A B C D E H I K A C 9 A D 14 A E 3 C H 7 D H 1 E I 10 E K 15 H K 2 I K 11 K B 6 A K	- Dòng đầu tiên chứa hai số nguyên dương v và e , lần lượt là số đỉnh và số cạnh của đồ thị - Dòng tiếp theo chứa v chuỗi (chuỗi không có khoảng trắng) là danh sách tên các đỉnh - 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 g , đây là đỉnh bắt đầu và đỉnh kết thúc của đường đi cần tìm <u>Lưu ý: thông tin nhập tương ứng với đồ thị có hình như bên dưới</u>
Output	Giải thích
A D H K	Danh sách các đỉnh trên đường đi (cả đỉnh xuất phát và đỉnh đích)

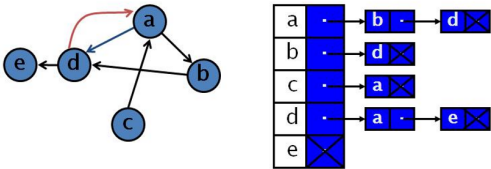
Hãy thực hiện các yêu cầu sau:

6.1 Xây dựng các CTDL 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.

Đáp án tham khảo:

Cách biểu diễn	Thiết kế CTDL	Thang điểm và cách đánh giá
Danh sách kề	<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>Đạt trọn 0.5 điểm nếu đáp ứng đầy đủ các yêu cầu sau:</p> <p>SV phải gọi được đúng tên phương pháp biểu diễn là “Danh sách kề” và mô tả được danh sách kề có cấu trúc thế nào, tức</p>

	<pre>map<string,set<pair<string,int>>> adj_list</pre> <p>Có thể dùng vector, list, tree thay cho set cũng được.</p> <p style="text-align: center;"> $A \rightarrow \{(B, 1), (D, 7)\}$ $B \rightarrow \{(E, 3)\}$ </p> <p>Hoặc</p> <pre>map<string,map<string,int>> adj_list</pre> <p> $A \rightarrow [B \rightarrow 10]$ $[O \rightarrow 45]$ $[C \rightarrow 12]$ $B \rightarrow [C \rightarrow 4]$ $[L \rightarrow 8]$ </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. Struct node chứa thông tin tên đỉnh và trọng số.</p>	<p>là nói được ý “ứng với mỗi đỉnh i, ta cần lưu trữ một tập hợp (danh sách) gồm các đỉnh kề với đỉnh i...” hoặc có ví dụ thể hiện được ý này.</p> <p>Nếu chỉ khai báo cấu trúc, như <code>map<string,set<pair<string,int>> adj_list</code> mà không kèm theo những mô tả/giải thích cụ thể, rõ ràng nhằm cung cấp thông tin chi tiết về cấu trúc cho người đọc dễ hiểu thì chỉ đạt 0.25 điểm</p>
Ma trận trọng số	<p>1. <code>vector<vector<int> matrix; // ma trận trọng số của đồ thị</code></p> <p>- Giá trị của mỗi ô trong ma trận nếu khác 0 thì cho biết trọng số của cạnh nối giữa 2 đỉnh tương ứng</p> <p>Hoặc <code>vector<vector<int> matrix (v,</code></p>	<p>Đạt trọn 0.5 điểm nếu đáp ứng đầy đủ các yêu cầu sau:</p> <p>Gọi tên phương pháp biểu diễn được chọn là “Ma trận trọng số/ma trận kề”, có mô tả ý nghĩa, vai trò của từng cấu trúc, và xử lý được vấn đề ánh xạ từ tên đỉnh là chuỗi sang index trong</p>

	<pre>vector<int>(v,0);</pre> <p>2. <code>vector<string> vnames; //</code> Cấu trúc lưu danh sách các tên (chuỗi) đỉnh, có thể thay vector là list/tree....</p> <p>3. <code>map<string, int> vindex;</code> - 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>vector/matrix</p> <p>Một số lỗi cần phải trừ điểm</p> <ul style="list-style-type: none"> - Cấp phát tĩnh hoặc kích thước không phù hợp: trừ 0.25 điểm <p>Nếu chỉ define MAX 2000 thì kích thước không phù hợp</p> <pre>vector <vector<int> > matrix (MAX, vector<int> (MAX, 0)); hoặc int **matrix; hoặc int matrix[MAX][MAX]</pre> <ul style="list-style-type: none"> - Không xử lý ánh xạ từ tên đỉnh là chuỗi sang index: trừ 0.25 điểm
--	---	---

6.2 Dưới đây là một ví dụ về mã giả của thuật toán Dijkstra để tìm một đường đi ngắn nhất từ đỉnh xuất phát s đến đỉnh đích g trong đồ thị. Hãy minh họa các bước thực hiện của thuật toán theo mã giả đã cho để **“Tìm đường đi ngắn nhất từ đỉnh A đến đỉnh H”** trong đồ thị bên dưới (0.75 điểm).

<p>Gọi Open: tập các đỉnh có thể xem xét lựa chọn cho bước đi tiếp theo, các đỉnh có thể được xem xét lại, đỉnh chờ duyệt</p> <p>Close : tập các đỉnh đã xét/đã duyệt, không xem xét lại</p> <p>s : đỉnh xuất phát</p> <p>g : đỉnh đích</p> <p>p : đỉnh đang xét, đỉnh hiện hành</p> <p>Hàm d(u) dùng để lưu trữ độ dài đường đi ngắn nhất từ đỉnh nguồn s đến đỉnh u</p> <p>w(u,i) : trọng số của cạnh nối từ đỉnh u tới đỉnh i</p> <p>parent(q)=p: lưu thông tin cha con, đỉnh p là cha của đỉnh q</p> <p>Các bước thực hiện chính:</p> <p>Bước 1: Khởi tạo</p> <p>Open = {s} Close = {} d(s)=0</p> <p>Bước 2: While (Open ≠ {})</p> <p>2.1 Chọn p thuộc Open có d(p) nhỏ nhất (xóa p ra khỏi Open). Nếu có nhiều đỉnh cùng giá trị d(p) nhỏ nhất thì chọn đỉnh nhập vào sau trong danh sách đỉnh</p> <p>2.2 Nếu p là đỉnh đích thì xuất đường đi, kết thúc thuật toán</p>
--

2.3 Nếu p đã duyệt rồi thì bỏ qua, không xem xét lại, trở lại đầu vòng lặp

2.4 Đánh dấu p đã duyệt qua rồi (tức thêm p vào Close)

2.5 Với mỗi đỉnh q kề với p, nếu q không thuộc Close:

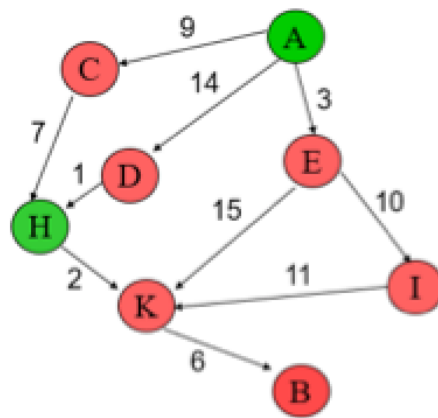
2.5.1 Nếu q đã có trong Open và $d(q) > d(p) + w(p,q)$ thì cập nhật các thông tin:

$$d(q) = d(p) + w(p,q) \quad \text{parent}(q)=p$$

2.5.2 Nếu q chưa có trong Open:

$$d(q) = d(p) + w(p,q) \quad \text{parent}(q)=p \quad \text{Thêm q vào Open}$$

Bước 3: Kết luận “Không tìm thấy đường đi”



Đáp án tham khảo:

Nội dung cần minh họa					Thang điểm và cách đánh giá
Quá trình duyệt các đỉnh	p	$\Gamma(p)$	Open	Close	Đúng 2 lần lặp đầu (tới dòng thông tin khi chọn đỉnh E) đạt 0.25 điểm
			{(A,0)}	{}	
	A	{C,D,E}	{(C,9), (D,14), (E,3)}	{A}	Nếu chỉ đúng tới dòng A thì chỉ đạt 0.125 điểm
	E	{I,K}	{(C,9), (D,14), (I,13), (K,18)}	{A,E}	
Yêu cầu: SV biết cách chọn đỉnh tốt nhất để đi tiếp, mở rộng các đỉnh kề với p bỏ vào Open và tính đúng các giá trị d(u)					
	C	{H}	{(D,14), (I,13), (K,18), (H,16)}	{A,E,C}	Đúng 4 lần lặp sau đạt 0.25 điểm
	I	{K}	{(D,14), (K,18), (H,16), (K,24)}	{A,E,C,I}	
	D	{H}	{(K,18), (H,16), (H,15)}	{A,E,C,I,D}	Nếu đúng tới dòng I thì đạt 0.125 điểm , đúng tới C thì cũng không có điểm
	H	(Kết thúc)			
Yêu cầu: khi gặp lại đỉnh trong Open thì phải tính lại khoảng cách, cập nhật khoảng cách mới nếu tìm thấy đường đi tốt hơn					
Đỉnh nào có gạch ngang là không có trong Open, SV không ghi trong bài làm cũng được					

Truy xuất tìm đường đi	parent	son	<p>Có kết luận đường đi và cho biết cách thức để tìm đường đi đó đạt 0.25 điểm</p> <p>Nếu chỉ kết luận đường đi A->D ->H mà không giải thích gì thêm thì chỉ đạt 0.125 điểm</p>
	A	C	
	A	D	
	A	E	
	E	I	
	E	K	
	E	H	
	I	K	
	D	H	
<p>Kết luận đường đi bằng cách truy ngược theo quan hệ cha – con: cha (H)=D, cha(D)=A, ta có đường đi là A->D->H</p> <p>Cần cho biết cách để tìm ra đường đi, SV không cần lập bảng như trên nhưng phải thể hiện được cách thức để truy xuất</p>			

6.3 Hãy thiết kế CTDL cho các đối tượng được đề cập trong thuật toán 6.2 (như Open, Close, hàm d(u), parent) và viết hoàn chỉnh hàm Dijkstra để giải quyết bài toán nêu trên (0.75 điểm).

Đáp án tham khảo:

Đoạn mã	Chi tiết các câu lệnh	Thang điểm và cách đánh giá
<p>Nếu SV không khai báo và mô tả đầy đủ, chi tiết các CTDL cho các đối tượng được đề cập trong thuật toán (như Open, Close, hàm d(u), parent) thì KHÔNG CHẤM PHẦN SAU, XEM NHƯ 0 điểm. Khi không thiết kế được CTDL phù hợp thì các câu lệnh sau đó không có ý nghĩa.</p> <p>Vì có nhiều cách cài đặt/viết hàm Dijkstra nên đoạn mã bên dưới chỉ mang tính chất tham khảo, tùy theo cách tổ chức CTDL mà cách viết câu lệnh theo từng bước của mã giả sẽ khác nhau, chỉ cần câu lệnh viết đúng cú pháp và xử lý đầy đủ các bước là có điểm tương ứng theo từng phần.</p>		
<p>Liên quan đến việc chọn đỉnh trong Open để đi tiếp</p> <p>0.25 điểm</p>	<pre> 1 void Dijkstra (int s, int g) // index của 2 đỉnh, 2 tham số kiểu string cũng được 2 { 3 //Tổ chức cấu trúc dữ liệu, vì có nhiều cách thiết kế nên GV linh hoạt trong cách chấm 4 priority_queue<...>open; // lưu các đỉnh chờ duyệt, nếu SV viết không đúng cú pháp <...> cũng không sao 5 vector<bool> close(v, 0); // lưu thông tin đỉnh nào đã duyệt qua rồi 6 map<string, string> parent; // lưu thông tin cha con, parent[u]=v nghĩa là cha của u và v 7 vector<int> d(v,INF); // lưu khoảng cách (ngắn nhất) từ s đến các đỉnh khác 8 9 Bước 1: Khởi tạo 10 open.push({0,s}); d[s] = 0; // close đã khởi tạo lúc khai báo rồi 11 Bước 2: while (!open.empty()) 12 <2.1> pair<int, int> top = open.top(); open.pop(); // SV không xử lý trường hợp có nhiều đỉnh cùng giá trị d(p) 13 // nhỏ nhất cũng không sao 14 int p=top.second; 15 16 <2.2> if (p == g) 17 { found=true; break; } // Có thể gọi hàm findpath (s,g) hoặc xử lý xuất đường đi luôn tại đây 18 19 <2.3> if(close[p]==1) continue;// p đã duyệt rồi thì bỏ qua 20 21 <2.4> close[p] = 1; // đánh dấu p đã duyệt rồi 22 -----ĐÚNG TỚI ĐÂY ĐƯỢC 0.25 ĐIỂM ----- </pre>	
<p>Liên quan đến việc mở rộng các đỉnh kề với p và cập nhật Open</p> <p>0.25 điểm</p>	<pre> <2.5> Xử lý các đỉnh kề với p for (int i = 0; i < v; i++) if (matrix[p][i]>0 && close[i]== 0) // tìm đỉnh kề mà chưa đi qua if (d[i]>d[p]+ matrix[p][i]) // tìm thấy con đường khác tốt hơn { d[i]=d[p]+ matrix[p][i] ; // cập nhật thông tin đường đi tốt nhất open.push({d[i],i}); parent[i] = p; } -----ĐÚNG TỚI ĐÂY ĐƯỢC THÊM 0.25 ĐIỂM NỮA ----- </pre>	

	<p>Ví dụ: trường hợp có sai nhẹ nhưng có thể thông cảm bỏ qua map<string, string> parent; // lưu thông tin cha con, parent[u]=v nghĩa là cha của u và v, u và v kiểu string ... câu lệnh cập nhật quan hệ cha con: parent[i] = p; // với i, p là số nguyên thì có sai do không tương thích với cách khai báo map bên trên Câu lệnh đúng phải là: parentp[vnames[i]] = vnames[p]; phải chuyển sang tên đỉnh để xử lý map (theo cách biểu diễn bằng Ma trận trọng số) hoặc đổi lại khai báo map<int, int> parent --> TUY NHIÊN, LỖI NÀY KHÔNG TRỪ ĐIỂM</p>
Liên quan với việc truy xuất đường đi 0.25 điểm	<p>Bước 3: if(found == false) //kết luận không tìm thấy else // Xử lý xuất đường đi -----CÁCH XỬ LÝ TÌM ĐƯỜNG ĐI CHẤP NHẬN ĐƯỢC THÌ MỚI CÓ 0.25 ĐIỂM, KHÔNG TÌM ĐƯỢC ĐƯỜNG ĐI ĐÚNG THÌ DÙ CÓ VIẾT NHIỀU CÂU LỆNH CŨNG KHÔNG CÓ XẤU ĐIỂM NÀO -----</p>

Hết

Duyệt đề Khoa/Bộ Môn

Đại diện nhóm Giáo viên ra đề