Thời gian còn lại 0:14:18

Câu hỏi 9

Chính xác

Điểm 1,00 của 1,00

Research **queue** which is implemented in C library at http://www.cplusplus.com/reference/queue/queue/. You can use library **queue** in c++ for this question.

Using **queue**, complete function **bool isBipartite(vector<vector<int>> graph)** to determine if a graph is bipartite or not (the graph can be disconnected). In caat https://en.wikipedia.org/wiki/Bipartite_graph.

You can use below liberaries in this question.

```
#include <iostream>
#include <vector>
#include <queue>
```

For example:

Test	Result

`int G[6][6] = { {0, 1, 0, 0, 0, 1},`
`                {1, 0, 1, 0, 0, 0},`
`                {0, 1, 0, 1, 0, 0},`
`                {0, 0, 1, 0, 1, 0},`
`                {0, 0, 0, 1, 0, 1},`
`                {1, 0, 0, 0, 1, 0} };`
`int n = 6;`
 	
`vector<vector<int>> graph(n, vector<int>());`
`        for (int i = 0; i < n; ++i) {`
`                for (int j = 0; j < n; ++j) {`
`                        if (G[i][j]) graph[i].push_back(j);`
`                }`
`        }`
 	
`isBipartite(graph) ? cout << "Yes" : cout << "No";`
	Yes

Answer: (penalty regime: 0 %)

Reset answer

```
bool isBipartite(vector<vector<int>> graph) {
    int n=graph.size();
    vector<int> colors(n,0);
    for (int i=0;i<n;i++)
    {
        if (colors[i] == 0)
        {
            std::queue<int> q;
            q.push(i);
            colors[i] = 1; // Color the starting vertex as 1

            while (!q.empty()) {
                int u = q.front();
                q.pop();

                for (int v : graph[u]) {
                    if (colors[v] == colors[u]) {
                        // Two adjacent vertices have the same color, not bipartite
                        return false;
                    }
                    if (colors[v] == 0) {
                        // Uncolored vertex, color it with the opposite color
                        colors[v] = -colors[u];
                        q.push(v);
                    }
                }
            }
        }
    }
    return true;
}
```

Kiểm tra

	Test	Expected	Got	

✓	
	```
	int G[6][6] = { {0, 1, 0, 0, 0, 1},
	                {1, 0, 1, 0, 0, 0},
	                {0, 1, 0, 1, 0, 0},
	                {0, 0, 1, 0, 1, 0},
	                {0, 0, 0, 1, 0, 1},
	                {1, 0, 0, 0, 1, 0} };
	int n = 6;

	vector<vector<int>> graph(n, vector<int>());
	\tfor (int i = 0; i < n; ++i) {
	\t\tfor (int j = 0; j < n; ++j) {
	\t\t\tif (G[i][j]) graph[i].push_back(j);
	\t\t}
	\t}

	isBipartite(graph) ? cout << "Yes" : cout << "No";
	```
		Yes	Yes	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 10

Chính xác

Điểm 1,00 của 1,00

Research **queue** which is implemented in C library at: http://www.cplusplus.com/reference/queue/queue/. You can use library **queue** in c++ for this question.

Using **queue**, complete function **void bfs(vector<vector<int>> graph, int start)** to traverse all the nodes of the graph from given start node using Breadth First Search algorithm and data structure **queue**, and print the order of visited nodes.

You can use below liberaries in this question.

```
#include <iostream>
#include <vector>
#include <queue>
```

For example:

Test	Result
`int init_graph[10][10] = {  {0, 1, 1, 0, 1, 0, 1, 0, 1, 0},`	0 1 2 4 6 8 3 7 5 9
`{0, 0, 1, 1, 0, 0, 0, 1, 0, 0},`	
`{0, 1, 0, 0, 0, 1, 1, 0, 1, 1},`	
`{1, 0, 0, 0, 0, 0, 0, 1, 0, 0},`	
`{0, 1, 0, 0, 0, 0, 0, 1, 0, 0},`	
`{1, 0, 1, 0, 1, 0, 0, 0, 1, 0},`	
`{0, 0, 1, 1, 0, 1, 0, 0, 0, 0},`	
`{1, 0, 0, 0, 0, 1, 1, 0, 1, 0},`	
`{0, 0, 0, 0, 0, 1, 0, 1, 0, 1},`	
`{1, 0, 1, 0, 1, 0, 0, 0, 1, 0} };`	
`int n = 10;`	
`vector<vector<int>> graph(n, vector<int>());`	
`for (int i = 0; i < n; ++i) {`	
`for (int j = 0; j < n; ++j) {`	
`if (init_graph[i][j]) graph[i].push_back(j);`	
`}`	
`}`	
`bfs(graph, 0);`	

Answer: (penalty regime: 0 %)

Reset answer

```
bool visited[100];
int path[100];

void bfs(vector<vector<int>> graph, int start) {
    for (int i=0;i<graph.size();i++)
    {
        visited[i]=0;
        path[i] = -1;
    }
    queue<int>q;
    visited[start] = true;
    q.push(start);
    while(!q.empty())
    {
        int u=q.front();
        q.pop();
        for (int i=0;i<graph[u].size();i++)
        {
            int v = graph[u][i];
            if (!visited[v])
            {
                visited[v]= true;
                q.push(v);
                path[v]=u;
            }
        }
        cout << u <<" ";
    }
    return;
}
```

Kiểm tra

	Test	Expected	Got	

✓	`\tint init_graph[10][10] = {  {0, 1, 1, 0, 1, 0, 1, 0, 1, 0},`
	`\t\t\t\t    {0, 0, 1, 1, 0, 0, 0, 1, 0, 0},`
	`\t\t\t\t    {0, 1, 0, 0, 0, 1, 1, 0, 1, 1},`
	`\t\t\t\t    {1, 0, 0, 0, 0, 0, 0, 1, 0, 0},`
	`\t\t\t\t    {0, 1, 0, 0, 0, 0, 0, 1, 0, 0},`
	`\t\t\t\t    {1, 0, 1, 0, 1, 0, 0, 0, 1, 0},`
	`\t\t\t\t    {0, 0, 1, 1, 0, 1, 0, 0, 0, 0},`
	`\t\t\t\t    {1, 0, 0, 0, 0, 1, 1, 0, 1, 0},`
	`\t\t\t\t    {0, 0, 0, 0, 0, 1, 0, 1, 0, 1},`
	`\t\t\t\t    {1, 0, 1, 0, 1, 0, 0, 0, 1, 0} };`
	`\tint n = 10;`
	`\tvector<vector<int>> graph(n, vector<int>());`
	`\tfor (int i = 0; i < n; ++i) {`
	`\t\tfor (int j = 0; j < n; ++j) {`
	`\t\t\tif (init_graph[i][j]) graph[i].push_back(j);`
	`\t\t}`
	`\t}`
	
	`\tbfs(graph, 0);`
		`0 1 2 4 6 8 3 7 5 9`	`0 1 2 4 6 8 3 7 5 9`	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 11

Chính xác

Điểm 1,00 của 1,00

Implement all methods in class **Queue** with template type **T**. The description of each method is written as comment in frame code.

```
#ifndef QUEUE_H
#define QUEUE_H
#include "DLinkedList.h"
template<class T>
class Queue {
protected:
    DLinkedList<T> list;
public:
    Queue() {}
    void push(T item) ;
    T pop() ;
    T top() ;
    bool empty() ;
    int size() ;
    void clear() ;
};

#endif /* QUEUE_H */
```

You can use all methods in class **DLinkedList** without implementing them again. The description of class **DLinkedList** is written as comment in frame code.

```
template <class T>
class DLinkedList
{
public:
    class Node;     //forward declaration
protected:
    Node* head;
    Node* tail;
    int count;
public:
    DLinkedList() ;
    ~DLinkedList();
    void add(const T& e);
    void add(int index, const T& e);
    T removeAt(int index);
    bool removeItem(const T& removeItem);
    bool empty();
    int size();
    void clear();
    T get(int index);
    void set(int index, const T& e);
    int indexOf(const T& item);
    bool contains(const T& item);
};
```

For example:

Test	Result

`Queue<int> queue;`
`    assert(queue.empty());`
`    assert(queue.size() == 0);`	

Answer: (penalty regime: 0 %)

Reset answer

```
void push(T item) {
    // TODO: Push new element into the end of the queue
    list.add(item);
}

T pop() {
    // TODO: Remove an element in the head of the queue
    T temp = list.removeAt(0);
    return temp;
}

T top() {
    // TODO: Get value of the element in the head of the queue
    return list.get(0);
}

bool empty() {
    // TODO: Determine if the queue is empty
    if (list.size() ==0) return true;
    else return false;
}

int size() {
    // TODO: Get the size of the queue
    return list.size();
}

void clear() {
    // TODO: Clear all elements of the queue
    list.clear();
}
```

Precheck Kiểm tra

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 12

Chính xác

Điểm 1,00 của 1,00

A nice number is a positive integer that contains only 2's and 5's.
Some nice numbers are: 2, 5, 22, 25, 52, 55, ...
Number 2 is the first nice number.
Given an integer N, return the Nth nice number.
Note: iostream, vector, queue are already included for you.

Constraint:
1 <= n <= 10^6

Example 1:
Input:
n = 5
Output:
52

Explanation:
The sequence of nice numbers is 2, 5, 22, 25, 52, 55, ...
The 5th number in this sequence is 52

Example 2:
Input:
n = 10000
Output:
2255522252225

For example:

Test	Input	Result

`int n;`
`cin >> n;`
`cout << nthNiceNumber(n) << endl;`
	5	52

`int n;`
`cin >> n;`
`cout << nthNiceNumber(n) << endl;`
	10000	2255522252225

Answer: (penalty regime: 0 %)

Reset answer

```
// iostream, vector and queue are included
// You can write helper methods

long long nthNiceNumber(int n) {
    queue<long long> q;
    q.push(2);
    q.push(5);
    while(true)
    {
        long long curr = q.front();
        q.pop();
        n--;
        if (n==0)
        return curr;
        q.push(10*curr + 2);
        q.push(10*curr + 5);
    }
    return 0.0;
}
```

Precheck Kiểm tra

	Test	Input	Expected	Got	

✓	`int n;`
	`cin >> n;`
	`cout << nthNiceNumber(n) << endl;`
		5	52	52	✓

✓	`int n;`
	`cin >> n;`
	`cout << nthNiceNumber(n) << endl;`
		10000	2255522252225	2255522252225	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.