

Instantly share code, notes, and snippets.



douglas-vaz / [graph\\_search.cpp](#)

Created 6 years ago

Breadth First Search and Depth First Search in C++

[graph\\_search.cpp](#)

```
1  #include <iostream>
2  #include <string>
3  #include <vector>
4  #include <queue>
5  #include <stack>
6  #include <algorithm>
7
8  using namespace std;
9
10 class Node{
11     char value;
12     vector<Node> children;
13 public:
14     Node(char c){
15         value = c;
16     }
17
18     void addChild(Node n){
19         children.push_back(n);
20         return;
21     }
22
23     void addChild(char n){
24         Node foo(n);
25         children.push_back(foo);
26     }
27
28     char getValue(){
29         return value;
30     }
31
32     vector<Node> getChildren(){
33         return children;
34     }
35
36     bool isLeaf(){
37         return children.size()==0;
38     }
39
40     bool operator==(Node b){
41         return b.value==value;
42     }
43 };
44
45
46 void construct(Node *r)
47 {
48     string foo;
49     cout<<"Enter children for "<< r->getValue() <<" (-1 for leaf)"<<endl;
50     cin>>foo;
51
52     if(foo == "-1")
53         return;
54     else{
```

```
55         for(int i = 0; i < foo.length(); i++)
56         {
57             Node t(foo[i]);
58             construct(&t);
59
60             r->addChild(t);
61         }
62     }
63 }
64
65 string breadthFirstSearch(Node root, Node goal)
66 {
67     std::queue<Node> Q;
68     std::vector<Node> children;
69     string path = "";
70
71     Q.push(root);
72
73     while(!Q.empty())
74     {
75         Node t = Q.front();
76         path += t.getValue();
77
78         Q.pop();
79
80         if(t == goal){
81             return path;
82         }
83         children = t.getChildren();
84         for (int i = 0; i < children.size(); ++i)
85         {
86             Q.push(children[i]);
87         }
88     }
89     return path;
90 }
91
92 string depthFirstSearch(Node root, Node goal)
93 {
94     std::stack<Node> Q;
95     std::vector<Node> children;
96     string path = "";
97
98     Q.push(root);
99
100    while(!Q.empty())
101    {
102        Node t = Q.top();
103        path += t.getValue();
104
105        Q.pop();
106
107        if(t == goal){
108            return path;
109        }
110        children = t.getChildren();
111        std::reverse(children.begin(), children.end());
112
113        for (int i = 0; i < children.size(); ++i){
114            Q.push(children[i]);
115        }
116    }
117    return path;
118 }
119
120 int main(int argc, char** args)
121 {
```

```
122     char r;
123     cout<<"Enter root node"<<endl;
124     cin>>r;
125
126     Node root(r);
127     construct(&root);
128
129     cout<<"Enter Node to search for: ";
130     cin>>r;
131
132     cout<<endl;
133
134     cout<<"BFS Traversal: "<<breadthFirstSearch(root, Node(' '))<<endl;
135     cout<<"BFS Search Path: "<<breadthFirstSearch(root, Node(r))<<endl<<endl;
136
137     cout<<"DFS Traversal: "<<depthFirstSearch(root, Node(' '))<<endl;
138     cout<<"DFS Search Path: "<<depthFirstSearch(root, Node(r))<<endl;
139
140     return 0;
141 }
```



bdawco commented on Apr 22, 2016

Hi I need help!!  
can any body explain me the DSF program?



asam139 commented on Feb 5

Perfect!!! Thanks so much +1: