자료구조 및 알고리즘 2 Revisited

김동진
(NHN NEXT)

Node 개수 Count (1)

- 문제
  - Binary tree가 주어져 있다.
  - 모든 노드의 개수를 구하는 프로그램을 작성하시오.

- Solution
  - 다음 장…

Node 개수 Count (2)

- Solution

```
 9 int countNode(node_t *node)
10 {
11     if(node == NULL) return 0;
12
13     return (countNode(node->leftChild) + countNode(node->rightChild) + 1);
14 }
```

Breadth-First Search (1)

- 문제
  - Graph가 주어져 있다.
  - Breadth-First search 로 노드를 출력하는 코드를 작성하시오.

- Solution
  - 다음 장…

Breadth-First Search (2)

```
28 void bfs(graph_t *graph)
29 {
30         queue <node_t *>        myQueue;
31         node_t                  *curNode;
32
33         curNode = graph->verArr[0];
34
35         myQueue.push(curNode);
36         curNode->visited = true;
37         while(!myQueue.empty()) {
38                 cout << myQueue.front()->id << endl;
39
40                 curNode = myQueue.front();
41                 for(int id = 0; id < curNode->adjVerArr.size(); id++) {
42                         if(curNode->adjVerArr[id]->visited == false) {
43                                 myQueue.push(curNode->adjVerArr[id]);
44                                 curNode->adjVerArr[id]->visited = true;
45                         }
46                 }
47
48                 myQueue.pop();
49         }
50 }
```

Depth-First Search

- 문제
  - Graph가 주어져 있다.
  - Depth-First search로 노드를 출력하는 코드를 작성하시오.

- Solution
  - 다음 장…

```
24 void dfs_visit(node_t *node, int *timeStamp, list<node_t *> *completedOrder)
25 {
26         node->visited = true;
27         node->timeStamp = ++(*timeStamp);
28
29         for(int id = 0; id < node->adjVerArr.size(); id++) {
30                 if(node->adjVerArr[id]->visited == false)
31                         dfs_visit(node->adjVerArr[id], timeStamp, completedOrder);
32         }
33
34         node->timeStamp = ++(*timeStamp);
35         completedOrder->push_front(node);
36 }
37
38 int     dfs(graph_t *graph, list<node_t *> *completedOrder)
39 {
40         int     timeStamp = 0;
41
42         if(graph == nullptr || completedOrder == nullptr) return -1;
43
44         for(int id = 0; id < graph->verArr.size(); id++)
45                 graph->verArr[id]->visited = false;
46
47         for(int id = 0; id < graph->verArr.size(); id++) {
48                 if(graph->verArr[id]->visited == false)
49                         dfs_visit(graph->verArr[id], &timeStamp, completedOrder);
50         }
51
52         return 0;
53 }
```

Topological Sort

- 문제
  - Directed Graph가 주어져 있다.
  - Directed edge a → b가 있으면 topological sort 결과에서 a는 b보다 앞에 위치해야 한다.
  - Topological sort하는 코드를 작성하시오.
    - Cycle이 존재하면 false를 return한다.

- Solution
  - 다음 장…

```
25 bool dfs_visit(node_t *rootOfCurTree, node_t *node, list<node_t *> *sortedList)
26 {
27     if(node->visited) {
28         if(node->root == rootOfCurTree)
29             return false;
30         else    return true;
31     }
32
33     node->visited = true;
34     node->root = rootOfCurTree;
35
36     for(int id = 0; id < node->adjVerArr.size(); id++)
37         if(dfs_visit(rootOfCurTree, node->adjVerArr[id], sortedList) == false) return false;
38
39     sortedList->push_front(node);
40
41     return true;
42 }
43
44 bool    dfs(graph_t *graph, list<node_t *> *sortedList)
45 {
46     if(graph == nullptr || sortedList == nullptr) return false;
47
48     for(int id = 0; id < graph->verArr.size(); id++)
49         graph->verArr[id]->visited = false;
50
51     for(int id = 0; id < graph->verArr.size(); id++) {
52         if(graph->verArr[id]->visited == false) {
53             graph->verArr[id]->root = graph->verArr[id];
54             if(dfs_visit(graph->verArr[id], graph->verArr[id], sortedList) == false) return false;
55         }
56     }
57
58     return true;
59 }
```

버그 있음.
Node가 gray인지 black인지 확인하는 방법을 사용해야 함.