

Leetcode 207. Course Schedule

// detecting cycles with topoSort

The screenshot shows the LeetCode interface for problem 207. The submission is successful with a runtime of 5 ms and memory usage of 40.2 MB. The code uses a topological sort approach to detect cycles in the course dependency graph.

Success Details:
 Runtime: 5 ms, faster than 63.15% of Java online submissions for Course Schedule.
 Memory Usage: 40.2 MB, less than 26.31% of Java online submissions for Course Schedule.

Next challenges:
 Course Schedule II, Graph Valid Tree, Course Schedule III

Submissions Table:

Time Submitted	Status	Runtime	Memory	Language
11/14/2020 14:24	Accepted	5 ms	40.2 MB	java

Code Snippet (Java):

```

26 for(int i = 0; i < inDegree.length; i++)
27     if (inDegree[i] == 0)
28         queue.offer(i);
29
30 int cnt = 0;
31 while(!queue.isEmpty()) {
32     int i = queue.poll();
33     cnt++;
34     for(int u: graph.get(i)){
35         inDegree[u]--;
36         if (inDegree[u] == 0)
37             queue.offer(u);
38     }
39 }
40 return cnt == graph.size();
41 }
42 }
43
  
```

Testcase Results:
 Accepted Runtime: 0 ms
 Your input: 2, [[1,0]]
 Output: true
 Expected: true

// detecting cycles with dfs

The screenshot shows the LeetCode interface for problem 207. The submission is successful with a runtime of 2 ms and memory usage of 39.4 MB. The code uses a DFS approach to detect cycles in the course dependency graph.

Success Details:
 Runtime: 2 ms, faster than 99.31% of Java online submissions for Course Schedule.
 Memory Usage: 39.4 MB, less than 79.40% of Java online submissions for Course Schedule.

Next challenges:
 Course Schedule II, Graph Valid Tree, Course Schedule III

Submissions Table:

Time Submitted	Status	Runtime	Memory	Language
11/14/2020 15:04	Accepted	2 ms	39.4 MB	java
11/14/2020 14:52	Wrong Answer	N/A	N/A	java
11/14/2020 14:37	Wrong Answer	N/A	N/A	java
11/14/2020 14:24	Accepted	5 ms	40.2 MB	java

Code Snippet (Java):

```

20 return graph;
21 }
22
23 boolean hasCycle(List<List<Integer>> graph, int node, int[] inCycle){
24     if(inCycle[node] == 1)
25         return true;
26     if(inCycle[node] == -1)
27         return false;
28
29     // if node is unvisited
30     inCycle[node] = 1;
31     for(int u: graph.get(node)) {
32         if(hasCycle(graph, u, inCycle))
33             return true;
34     }
35     inCycle[node] = -1;
36     return false;
37 }
38
39
  
```

Testcase Results:
 Accepted Runtime: 0 ms
 Your input: 2, [[1,0]]
 Output: true, false
 Expected: true, false