Longest Cycle in a Graph

Wang Xiyu

23 Dec 2024

Problem Statement

Given a directed weighted graph, find the length of the longest cyclic path.

Problem Constraints

Easy

0..1 outgoing edge per node.

Generalized

0..* outgoing edges per node.

Variation

The graph is acyclic, find the longest path (critical path). All edges have non-negative weights.

Concept Proof

Easy

Since $\forall v \in V, |E_v : \{e|e \in E \wedge e(v, v')\}| <= 1$, $!\exists e \in E, e$ belongs to more than one possible cycle, and $!\exists v \in V, v$ belongs to more than one possible cycle. $\implies$ if we have visited a $v \in G$,

Theorem 0.1. *If there exist more than one cycle in a graph of said constraints, they are in disconnected components.*

Proof. From the definition of cycle, and problem constriants,

$$\forall v \in C, (v, v') \in E \implies v' \in C$$

Which implies a cycle would not have an outgoing edge. Group all $v \in C$ to v'', v'' has no outgoing edge $\implies v''$ is a sink. Suppose there are more than one cycle in a connected component in G,

$$\exists v \in V, \forall v_{C1} \in C_1, \forall v_{C2} \in C_2, v \vdash^* v_{C1} \wedge v \vdash^* v_{C2}$$

$\implies \exists v_{branch}(v \vdash^* v_{branch} \wedge v_{branch} \vdash v_{C1} \wedge v_{branch} \vdash v_{C2})$ has more than one outgoing edge, a contradiction. $\square$

Therefore, it is possible to enumerate all cycles in G in one iteration of the edge array, and the longest among which is the solution.

Generalized

The generalized version of this problem is NP-Complete, as it can be reduced from Hamitonian Circuit. Scenario-specific techniques need to be employed to derive an efficient algorithm.

Theorem 0.2. *Longest cycle in a directed weighted graph is NP-Complete.*

Proof. Longest cycle is NP.

Consider the decision version of the problem: if $\exists$ a cycle in graph G that has total lenght $\geq k$. It is trivial to show that both checking if a set of vertices forms a cycle, and summing up the lenght of involved edges takes polynomial time.
Longest cycle is NP-hard.

Equivalent to Longest path in directed graph, by starting and ending in the same vertex. □

Variation: Longest path in DAG

use topological sort and DP

Solution

Easy

```
class Solution {
  public:
    int maxLength = -1;
    void getcycle(vector<int> &edges,int start,
            vector<bool>& visit,vector<int>& store){
        if(start == -1)return ; // if prev node leads to nowhere, current node is a sink
        if(visit[start]){ // if the node is already visited
            int count = -1;
            for(int i =0;i<store.size();i++){
                if(store[i]==start){ // if current node has been recorded
                    count = i; // set count to be the index of start in store
                    break;
                }
            }
            if(count==-1)return; // if current node is not found in a path
                                 // -> no cycle in current component
            int size = (store.size()-count);
            maxLength = max(maxLength,size); // update current max cycle length
            return ;
        }
        visit[start] = true; // mark this node as visited
        store.push_back(start); // mark this node as appeared in current path
        getcycle(edges,edges[start],visit,store); // edges[start] -> start.next
        return ; // finish exploring a component
    }
    int longestCycle(vector<int>& edges) {
        vector<bool> visit(edges.size(),0);
        for(int i =0;i<edges.size();i++){
            if(visit[i])continue; // node i is in an already explored component
            vector<int> store;
            getcycle(edges,i,visit,store); // find the cycle length
                                          // (if existent) in the component containing i
        }
```

```
        return maxLength;
    }
};
```

Variation: Longest path in DAG

```
class Solution {
    public:
        vector<int> dist;
        vector<bool> visited;
        void dfs(vector)
};
```