36111-cwk2-S-exerciseA

Name: Bowen Cai stu ID:10406140

Name: Haorui Chen stu ID:10407315

1. Suppose there is at least an acyclic graph with every one of its vertices have minimum in-degree of 1. We can imagine such a graph with N vertices. Pick any of those nodes as starting point, we trace back along an edge that contributes to indegree of current node. For example, the graph have edges ……, (vk, vk+1), (vk+1, vk+2), (vk+2, vk+3), then from vk+3 we go to vk+2, vk+1, vk……sequentially. When we trace back at most N times, we will eventually arrive at a vertex that we have visited, that is, there is a cycle in graph, which is contradicted with the claim above. Thus, every non-empty acyclic directed graph has at least one vertex with indegree 0.

Suppose there is one non-empty acyclic directed graph G(V, E) which has no vertex with in-degree 0. Pick a starting vertex v belongs to V such that v has non-zero out-degree. Then traverse through G. Since G is a finite graph, and v has in-degree greater than 0, there exists a finite sequence v0, v1, … vk=v0 (vk>=2, v0…vk belongs to V) in G such that for all i(0<=i<k), (vi, vi+1) is an edge belongs to E, which means there exists a cycle in G. This is contradicted to the assumption that G is acyclic. Therefore, every non-empty acyclic directed graph has at least one vertex with in-degree 0.

1. First, we need to prove that If a directed graph G has a topological ordering then G is acyclic. Assume G is not acyclic, then there is a cycle in G. we can let those edges in the cycle be (v0, v1), (v1, v2), (v2, v3) ……, (vk-1, vk), (vk, v0). Since G have a topological ordering, the edges in the cycle will have v0 < v1 < v2 < v3 <…< vk-1 < vk < v0, which is impossible. Therefore, if a directed graph G has a topological ordering then G is acyclic.

First, let’s prove that if a directed graph G(V, E) has a topological ordering then it is acyclic. By contradiction, we assume that G(V, E) is a cyclic directed graph which has a topological ordering. Then there exists a cycle C v0, v1, … vk=v0 (vk>=2, v0…vk belongs to V) in G such that for all i(0<=i<k), (vi, vi+1) is an edge belongs to E. Since G has a topological ordering, we can conclude from C that v0 < v1 < v2 … < vk-1 < v0, which is impossible. Therefore, we prove that if a directed graph G(V, E) has a topological ordering then it is acyclic.

Then we need to prove that If directed graph G is acyclic then G has a topological ordering. Assume that acyclic graph G={V ,E} does not have a topological ordering, it means that for every vertex vi ∈V we always have vj ∈V such that (vj, vi) ∈E (if not, then we can start to run topological ordering algorithm on this graph as it always look for a vertex with no in-degree in every iteration), meaning that every vertex in G will have a minimum in-degree of 1. Yet we have proved that such a graph is not acyclic in the previous question, which contradicts with the claim that the graph is acyclic. Therefore, if directed graph G is acyclic then G has a topological ordering.

Next, we need to prove that if directed graph G(V, E) is acyclic then G(V, E) has a topological ordering. Since G is acyclic, we have the conclusion from Question 1 that G has at least one vertex with in-degree 0. We pick one vertex v0 belongs to V with in-degree 0, and remove it from G. The newly derived graph G1(V1, E1) is still acyclic since a cycle cannot be form by removing edge. We again pick one vertex v1 belongs to V1 with in-degree 0, and remove it from G1. The newly derived graph G2(V2, E2) is still acyclic. Repeat the process until there is no vertex left to be removed. According to the order of the vertices being removed from G, we derive an ordering of vertices v0, … vn-1. For an edge (vi, vj) (0<=i<j<n) of G, it must be removed before vj is removed, while vi must be removed after vj is removed, we can obtain an order i<j which holds for every edge(vi, vj), which is a topological ordering. Therefore, we prove that if directed graph G(V, E) is acyclic then G(V, E) has a topological ordering.

According to the proofs above, we can prove that a directed graph is acyclic if and only if it has a topological ordering.

1. First, we need to show that G is acyclic if it does not contain a strict cycle. If a graph contains a cycle, then it must contain at least one strict cycle. We do not insist that the vertices in a cycle are all distinct, meaning that some vertices in the sequence v0...,vk = v0 can be the same. If we have a sequence v0,... , vm, ..., vn=vm, ..., vk = v0, then we have 2 strict cycles: v0, …, vm, …, vk = v0 and vm, ..., vn=vm. if there is no strict cycle in a graph, then there is also no cycle, and the graph is acyclic.

First, let’s prove that if G is acyclic, then it does not contain a strict cycle. By contradiction, assume that G(V, E) is acyclic but also contains a strict cycle SC such that SC is a sequence of v0, … vk-1(k>=2) such that for all i (0<=i<k-1), (vi, vi+1) belongs to E and (vk-1, v0) belongs to E. We can then form a sequence of vertices v0, …, vk=v0(k>=2) such that, for all i(0<=i<k), (vi, vi+1) belongs to E, which is a cycle. However, it is impossible for an acyclic graph G to contain a cycle. Therefore, we prove that if G is acyclic, then it does not contain a strict cycle.

Then we need to prove that a graph does not contain a strict cycle if it is acyclic. Assume that there is an acyclic graph G that contains a strict cycle, it means that G contains a distinct vertex sequence v0...,vk = v0. Yet by definition this sequence also gives us a cycle, then a cycle appears in acyclic G, which is impossible. Therefore, a graph does not contain a strict cycle if it is acyclic.

1. We try to show that (a) implies (b), (b) implies (c) and (c) implies (a).
2. (b) implies (a). if a graph doesn’t contain cycle, then it can only contain trees/linked lists/isolated points. We regard linked list as a tree, then for each tree component, we can put all vertices in odd layer to set U0, all vertices in even layer to set U1. For isolated points we can put them into any of these 2 sets. In this way we can guarantee that every vertex in set U0 is not connected with other vertices in U0, so for vertices in U1. By doing this, we can see the graph that contain tree/multiple trees is bi-partite. If a graph only contains cycle with even length, then for each cycle, we can label its vertices as v0, v1, … v2n and assign all vertices with odd subscript to set U0, vertices with even subscript to set U1, for those vertices that are not in a cycle, we can see them as trees and assign their vertices like what we did in the first part. Thus we can ensure every vertex in set U0 is not connected with other vertices in U0, so for vertices in U1, and the graph is bi-partite. Therefore, if a graph doesn’t contain cycle with odd length, then it is bi-partite.
3. (c) implies (b). we try to prove its contrapositive. The purpose of algorithm bCheck(G = (V,E)) is to check whether the graph is 2-colorable. If the graph is 2-colorable, bCheck returns true, otherwise it returns false. Consider a graph G that contains cycle of odd length. We mark the vertices in the cycle as v0, v1, … v2n=1. For 2 colors A and B, we color the first 2n vertices as A, B, A, B… as this is the only way if we want to make the cycle 2-colorable, but for v2n=1, neither of A and B can be assigned to make a 2-colorable graph, therefore, a graph contains cycle of odd length is not 2-colorable and bCheck will return false.
4. (a) implies (c). for a bi-partite graph G= (V= U0∪U1,E), we can assign all vertices in U0 with color A, all vertices in U1 with color B. If we check such a graph, we can know that each edge e∈E has one vertex colored A and the other colored B since for every edge e, exactly one vertex of e is in U0 and the other in U1. Therefore, bi-partite graph is 2-colorable.

We have proved that (b) → (a), (c) → (b), (a) → (c). By transitivity of implication, we have (a) → (b), (b) → (c), (c) → (a), thus having (a) ⇔ (b), (b) ⇔ (c), (c) ⇔ (a), and the 3 statements are equivalent.

First, let’s prove that if G(U0, U1, E) is bipartite then G contains no cycle of odd length. By contradiction, assume that C is a cycle of odd length in G, then C is a sequence of vertices v0, v1, … vk-1, vk=v0 (k>=2). Since for every edge e of E, exactly one vertex of e is in U0 while another vertex in U1, we suppose v0, v2, … vk are in U0, then v1, v3, … vk-1 are in U1, vice versa. Let e1 denotes the edge connecting v0 with v1, e2 denotes the edge connecting v1 with v2..., then ek will connect vk-1 and vk. Because C is of odd length, k will be an odd number, which means vk will also be in U1. However, U0 and U1 are disjoint sets, it is impossible for vk to be in both U0 and U1. Therefore, we prove that if G(U0, U1, E) is bipartite the nG contains no cycle of odd length.

Now let’s prove that if G contains no cycle of odd length then bCheck(G) returns true. Analyzing the algorithm, we know that bCheck(G) will return true if any adjacent vertex of G can be coloured differently with only 2 colours. We randomly choose a vertex v, and colour each vertex according to their shortest distance from v. That is, if a vertex is even distance away from v, we will colour it differently from v, else, we will give it the same colour with v. Coloured in this way, for every even number n and odd number m (n, m < longest path from v)(vice versa), there are no nodes in G that are n shortest distance apart from each other, while also having a path of length m between them. We will further prove that by contradiction. Assume there are vertices pairs satisfying the two conditions at the same time, and we pick the vertex pair vx, vy which has the minimum shortest distance nmin among these vertex pairs. Then for vx and vy, the path between them of length n and m will have no vertices in common except themselves. However, the two paths form a cycle of length (n+m) which is of odd length and is contradictory with the assumption that G contains no cycle of odd length. Therefore, we prove that if G contains no cycle of odd length then bCheck(G) returns true.

Finally, let’s prove if bCheck(G) returns true, then G is bipartite. We denote f as the two colouring of G derived from running bCheck(G). Since f is a proper colouring, there are no edges within the same colouring vertices set of G. Therefore, G is bipartite with bipartition of the two different colouring vertices sets.

According to the proofs above, we can prove that the three statements in the questions are equal.