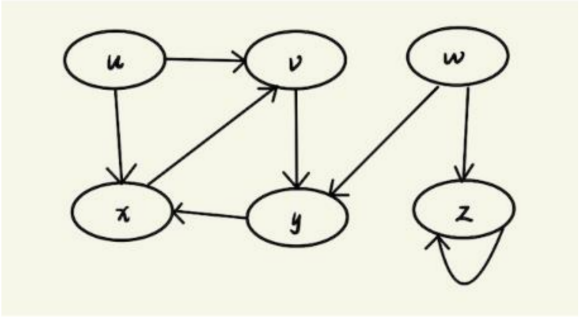
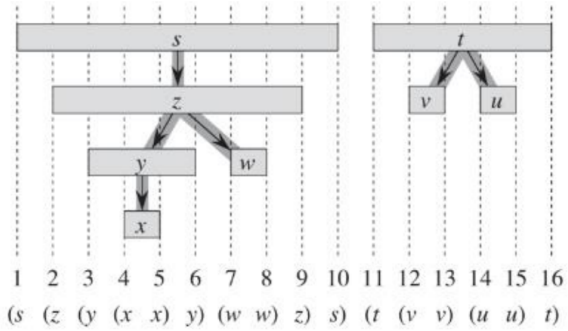
EL-GY 9343 Homework 8

Yihao Wang
yw7486@nyu.edu

1. What is the running time of DFS if the graph ($G = (V, E)$, where there are $|V|$ nodes and $|E|$ edges) is given as an adjacency list and adjacency matrix? Justify your running time.
   - If using adjacency list, it takes $\Theta(degree(u))$ to list all adjacent vertices for each vertex u when visiting u. Moreover, the sum of length of all adjacent lists is $\Theta(|E|)$. Therefore, the total running time is $\underline{\Theta(|V| + |E|)}$.
   - If using adjacency matrix, it takes $\Theta(|V|)$ to list all adjacent vertices for each vertex u when visiting u. Therefore, in total it takes $\underline{O(|V|^2)}$.

2. Draw the parenthesis structure of the DFS of bottom left figure (start from u, assume that DFS considers vertices in alphabetical order) and see the example parenthesis structure as is shown in the bottom right.





$(u\ (v\ (y\ (x\ x)\ y)\ v)\ u)\ (w\ (z\ z)\ w)$

3. **Bipartiteness**. Given a undirected graph $G = (V, E)$, it is *bipartite* if there exist U and W such that $U \cup W = V$, $U \cap W = \varnothing$, and every edge has one endpoint each in U and W.
   (a) Prove: G is bipartite only if G has no odd cycle. (Hint: proof by contradiction)
   Let's say we have a bipartite graph $G = (V, E)$ such that $U \cup W = V$, $U \cap W = \varnothing$, and there is an odd cycle which contain $2K + 1$ vertices, where $K \geq 1$. Since every edge has one endpoint each in U and W, each u_i must be connected with 2 w_j, and vice versa. Let $u_1, u_2, \ldots, u_I$ be vertices in the odd cycle that are in U and $w_1, w_2, \ldots, w_J$ be vertices in W, where $I + J = 2K + 1$. W.l.o.g, let $I \leq K, J \geq K + 1$, and let w_i be connected to u_i and u_i be connected to w_{i+1} for $i = 1, 2, \ldots, I$. Therefore, $\{w_j\}$ for all $j \geq I + 1$ are connected with each other, meaning there are endpoints of some edges fall in the same part. Here we have the contradiction, indicating the initial assumption is wrong.

Algorithm 1 Testing bipartiteness of graphs

1: Do BFS starting at any node u
2: Let $L_0, L_1, L_2, \ldots, L_k$ be layers in the resulting breadth-first tree ($L_0 = \{u\}, L_i, i = 1, 2, \ldots, k$ contains the vertices at distance i from u)
3: **if** There is no edge inside any layer L_i **then**
4: Declare G to be bipartite, and $U = L_0 \cup L_2 \cup L_4 \cup \ldots, W = L_1 \cup L_3 \cup L_5 \cup \ldots$ are the bipartition
5: **else**
6: Declare G to be non-bipartite
7: **end if**

(b) In fact, G is bipartite if and only if G has no odd cycle. Suppose this is given, consider the Algorithm 1 and briefly describe why it is correct.
To have a odd cycle in the graph, there must be an edge between two layers L_i and L_j, where $|i - j|$ is an even number (including 0). However, BFS ensures that there is no edge between layers that are not adjacent. Therefore, we only need to check if there are edges inside any layer to determine if the graph is bipartite.

(c) Analyze the time complexity (worst-case, big-O) of the algorithm, in terms of $|V|$ and $|E|$. (Hint: can we check that there is no edge inside any layer in $O(|E|)$? Why?)
It costs $O(|V| + |E|)$ to do BFS. We can achieve $O(|E|)$ when checking there is no edge inside any layer by traversing all edges and see if two endpoints are in different layers. Thus, in total the time complexity is $O(|V| + |E|)$.

4. You're helping a group of ethnographers analyze some oral history data they've collected by interviewing members of a village to learn about the lives of people who've lived there over the past two hundred years. From these interviews, they've learned about a set of n people (all of them now deceased), whom we'll denote $P_1, P_2, \ldots, P_n$. They've also collected facts about when these people lived relative to one another. Each fact has one of the following two forms:

(a) For some i and j, person P_i died before person P_j was born; or
(b) for some i and j, the life spans of P_i and P_j overlapped at least partially.

Naturally, they're not sure that all these facts are correct; memories are not so good, and a lot of this was passed down by word of mouth. So what they'd like you to determine is whether the data they've collected is at least internally consistent, in the sense that there could have existed a set of people for which all the facts they've learned simultaneously hold.
The original problem asks you to give an efficient algorithm to either produce proposed dates of birth and death for each of the n people so that all the facts hold true, or report (correctly) that no such dates can exist — that is, the facts collected by the ethnographers are not internally consistent. Since topological sort will be taught next week, now let's do make it simple. **You only need to give an efficient algorithm to determine whether the data is internally consistent.**

Hints: Let's say there are totally m recorded facts, and your algorithm should run in $O(m+n)$. It should be useful to turn the data into a graph. If each node represents an event (birth or death of someone), what is the total number of nodes? How to define the edges, directed or undirected? What makes the data inconsistent? What is the key structure to check in the graph?

For all n people, we use two node per person to represent their birth and death event, so there are in total $2n$ nodes. And we defined an edge as a time sequence according to a fact:

- If the fact is in form (a), we can create an edge that points from the death event of P_i to the birth event of P_j.
- If the fact is in form (b), we can create an edge that points from the birth event of P_i to the death event of P_j, and another event that points from the birth event of P_j to the death event of P_i.

After creating the whole directed graph, we have $|V| = 2n$ and $|E| = \Theta(m)$. The critical structure to detect is the cycle, because there shouldn't be any cycle in the time line. To detect the cycle, we can use a modified version of DFS as in Algo.2, which use a modified version of DFSVisit as in Algo.3.

The total time complexity of this version of DFS is still $O(|E| + |V|)$, which yields a running time of $O(m+n)$.

Algorithm 2 Determining Inconsistency in Birth and Death Events

Input: n people and m facts

Output: whether or not all facts are consistent

Build a directed graph G from n people and m facts according to aforementioned description
for v in G **do**
    $A[v] \leftarrow$ False; $T[v] \leftarrow$ False
end for
for v in G **do**
    if DFSVisit(v, A, T) **then**
        return True
    end if
end for
return False

Algorithm 3 DFSVisit

Input: current vertex v, visitedArray A, tracedArray T

Output: whether or not there is a cycle in the sub-graph

$A[v] \leftarrow$ True
$T[v] \leftarrow$ True ▷ Mark v as being traced when visiting the sub-graph of v
for u in Adj(v) **do**
    if $A[u]$ **then**
        return $T[u]$ ▷ There is a cycle if u is visited and being traced at the same time
    else
        return DFSVisit(u)
    end if
end for
$T[v] \leftarrow$ False
return False