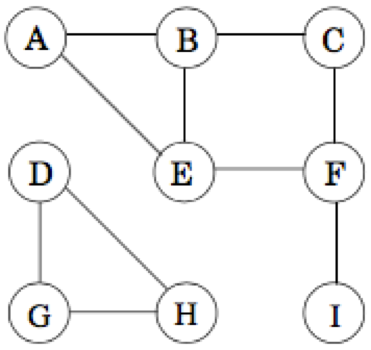
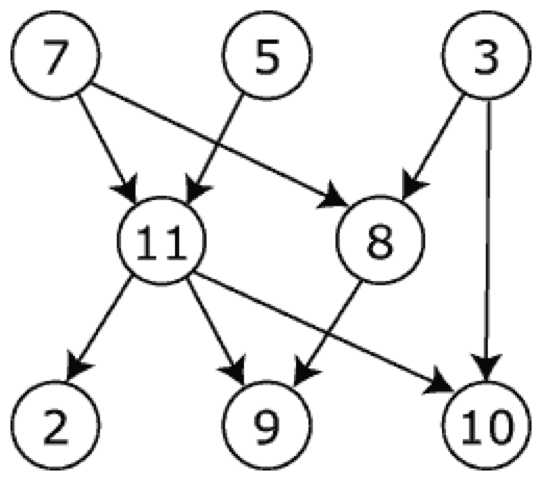
X1Design and Analysis of Algorithm

### **Part I - Fundamentals of Graphs Algorithm**

### **Problem 1 5+5 points**

Perform a depth-first search on the following graphs (undirected and directed); whenever there’s a choice of vertices, pick the one that is alphabetically or numerically first. Classify each edge as a tree edge or back edge, and give the pre and post number of each vertex. Write the Pseudo Code which writes off which edge is which in case of both directed graphs (back, forward, tree, and cross edge) and undirected graphs(either tree edge or non-tree edge).

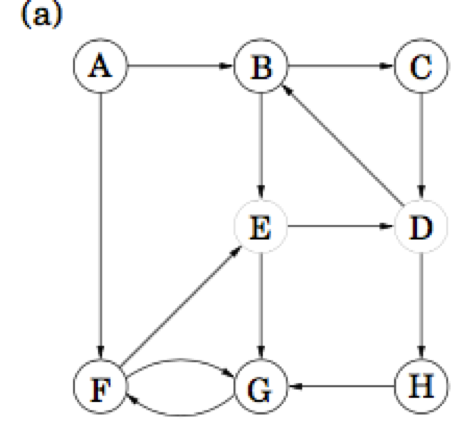
 

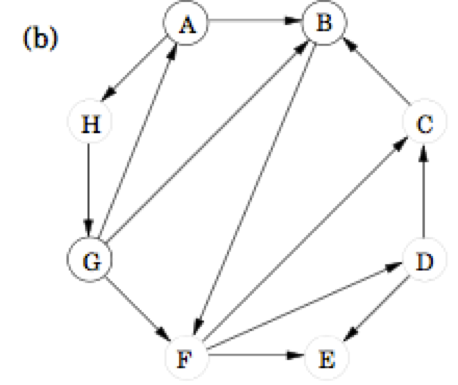
### **Problem 2 (Pseudo Code required, use your implementation of Graphs which we did in the last HW) 5x2=10 points**

Write an algorithm efficiently to find SOURCE and SINK nodes from any given graph, with this given information that there exists a source as well as sink vertex. Using DFS only once.

* If the Graph is an Adjacency Matrix.
* If the Graph is an Adjacency List.

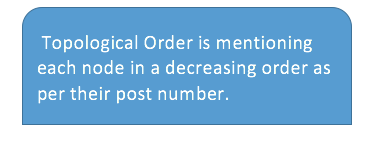
### **Problem 3 5+5+10 points**

* Perform depth-first search on each of the following graphs; whenever there’s a choice of vertices, pick the one that is alphabetically first. Classify each edge as a tree edge, forward edge, back edge, or cross edge, and give the pre and post number of each.   
    
  

Given a directed graph write an algorithm which detects if it has a cycle or not? The cycle is if you start from a vertex and move ahead following the edges (directed) and after few hops come to the same vertex again; e.g. in above picture b) A-> H-G->A is a cycle.

### **Problem 4 Pseudo Code required along with screenshots 5x4 = 20 points**

MAKE TOPOLOGICAL ORDERING CODE(given a DAG you should be able to run the program, and the program should return topological ordering). Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

1. What are the sinks vertices of the graph? No code required.
2. What topological ordering is found by your code? If you look into the ordering to be like - rearrange the ordering of the adjacency matrix (along with node in the graph, also store the symbol of the node so that you can print it along while printing the topological ordering).
   * ABCDEFGH
   * OR FEDGHBA
   * OR DCEFBAGH
3. How many topological orderings does this graph have?
4. What if there is a cycle? Modify your code in such a way that it should be able to detect whether the solution is not possible if there exists a cycle in the given dependency graph.

### **Problem 5 10 points**

Write an algorithm (Pseudo Code) to assign each node in the given Graph G, its component number. Run your algorithm on this given graph. Where component number of vertex A, B, C, D and E is 1, component number of vertex F, G and H is 2 and component number of I and J is 3.

### **Problem 6 (Short Questions) 10 points**

* NOTE: Degree sequence is when you write down the degree of all the vertices in decreasing order.

1. The sum of all the numbers in the Degree Sequence will always be even number. Why?
2. The Odd numbers in the sequence will always appear even number of times. Why?
3. At-least one number will be repeated in the sequence. Why?

* Salma invites three of her office colleagues for dinner. Salma’s husband Aslam (known as A) received the guest when they have arrived while Salma was in the kitchen preparing food. Aslam shook hands with some of the guests. Obviously Salma was curious about whom Aslam shook hands and with whom he did not. She cannot ask this question directly so she simply asks every colleague about the number of hands she had shaken; she gets the following answer: B says 3, C says 2 and D says 1. Now Salma asks the same question to her husband; he answers zero (he does not shake hands with females). Is the husband telling the truth assuming the guests told the truth? How about if he answers 3 (he showed no discrimination). Find out the number of times the husband actually shook hands and more importantly with whom he shook hands and whom he has ignored.

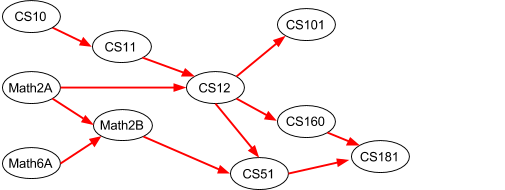
### **Problem 7 Pseudo Code required 5+5+5 points**

1. Write the Pseudo Code for **BFSVisit**, given a source S find out all the shortest distances from S to all the vertices.   
2. Write the Pseudo Code for **BFSAll** which given a list Ss(array of) sources find out the closest distance from any node to one of the closest source in Ss.   
3. Given a list of vertices after running **BFSAll**, print closest paths from the source (one of the source in Ss) to the corresponding vertex.

### **Problem 8 Grid Graph and relatives Pseudo Code required 5+5+5 points**

* Given a grid graph, represented in the form of two dimensional matrix, where you don’t need any extra pointer memory, rather you use indexing of left, right, top and bottom, read from the file and do the following tasks
  1. Solve the Prison break problem using DFS and report the path.
  2. Solve the Prison Break problem using DFS and report the shortest path.
  3. Extend the above parts problems such that now imagine you have multiple prisoners and you have tell about each prisoner whether one can break the prison or not. HINT: First memorize all the locations and then try calling for each location once.

**PART 2 - THE CHALLENGES**

**Challenge 1**

Below is a set of required courses for a degree. The directed acyclic graph shows the prerequisite structure for the courses.

Your job is to devise an academic plan for a student. An academic plan is a set of courses for the student to take in each quarter. In each case, you need to devise an academic plan which respects the prerequisites and takes the fewest number of quarters. You can assume that every course is offered in every quarter. **Design Algorithms for the above 4 plans? NOTE: Your algorithm must of independent of the above example.**

* Devise an academic plan for the student if he can only take one course per quarter.
* Devise an academic plan for the student if he can take up to two courses per quarter.
* Devise an academic plan for the student if he can take up to three courses per quarter.
* What's the fewest number of quarters the degree will take if the student can take an unlimited number of courses per quarter?

# **Challenge 2**

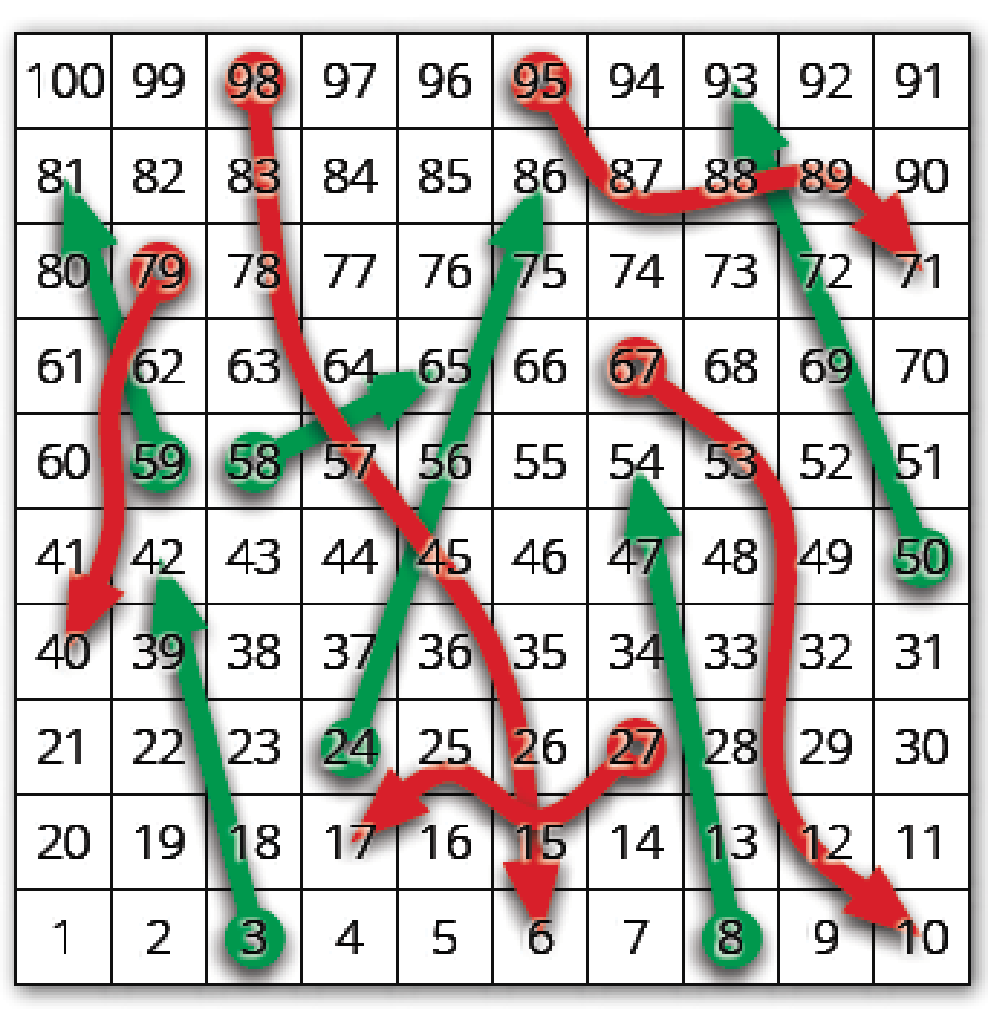
CLRS: *22.1-5, 22.1-6*

# **Challenge 3**

1. We saw the algorithm for BFS(G, s) which sets the distance as well as parents attribute of each vertex on the closest path from s to every vertex reachable from s. Write an algorithm PRINT-SourceToDestinationPATH(G, s, d) which should print the path from s to d. Hint: Use Recursion.
2. There are two types of professional wrestlers: “Aanglo Pehlwaan” (“Pakistani Players”) and “Baanglo Pehlwaans” (“Indian Players”). Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have n professional wrestlers and we have a list of r pairs of wrestlers for which there are rivalries. Give an O(n+ r)-time algorithm that determines whether it is possible to designate some of the wrestlers as “Aanglo Pehlans” and the remainder as “Baanglo Pehlwaans” such that each rivalry is between an Aanglo and a Banglo. If it is possible to perform such a designation, your algorithm should produce it.
3. A directed graph G(V,E) is semiconnected if, for all pairs of vertices , we have  or  Give an efficient algorithm to determine whether or not $G$ is semi-connected. Prove that your algorithm is correct, and analyze its running time.
4. A graph (*V*, *E*) is *bipartite* if the vertices *V* can be partitioned into two subsets *L* and *R*, such that every edge has one vertex in *L* and the other in *R*.
   1. Prove that every tree is a bipartite graph.
   2. Describe and analyze an efficient algorithm that determines whether a given undirected graph is bipartite or not?

# **Challenge 4**

*Snakes and Ladders* is a classic board game, originating in India no later than the 16th century. The board consists of an *n* × *n* grid of squares, numbered consecutively from 1 to *n*2, starting in the bottom left corner and proceeding row by row from bottom to top, with rows alternating to the left and right. Certain pairs of squares in this grid, always in different rows, are connected by either “snakes” (leading down) or “ladders” (leading up).

Each square can be an endpoint of at most one snake or ladder. 

A typical Snakes and Ladders board.

Upward straight arrows are ladders; downward wavy arrows are snakes.

You start with a token in cell 1, in the bottom left corner. In each move, you advance your token up to *k* positions, for some fixed constant *k*. If the token ends the move at the *top* end of a snake, it slides down to the bottom of that snake. Similarly, if the token ends the move at the *bottom* end of a ladder, it climbs up to the top of that ladder.  
Describe and analyze an algorithm to compute the smallest number of moves required for the token to reach the last square of the grid.

1. How to transform this problem into Graphs notation?
2. What problem should we be solving which should give the result of the above problem?
3. Write an algorithm to solve that Graph problem which will result into the optimal answer of the above question?

**Challenge 5** - When there is more than one shortest path from one node *s* to another node *t*, it is often convenient to choose a shortest path with the fewest edges; call this the *best path* from *s* to *t*. Suppose we are given a directed graph *G* with positive edge weights and a source vertex *s* in *G*. Describe and analyze an algorithm to compute *best* paths in *G* from *s* to every other vertex.

**Challenge 6** You are given a set of cities, along with the pattern of highways between them, in the form of an undirected graph G= (V,E). Each stretch of highway e 2 E connects two of the cities, and you know its length in miles, le . You want to get from city s to city t . There's one problem: your car can only hold enough gas to cover L miles. There are gas stations in each city, but not between cities. Therefore, you can only take a route if every one of its edges has length le L . Write an algorithm to drive your car from city S to T in a way that your car won’t run out of fuel.

**Challenge 7**

You just discovered your best friend from elementary school on facebook. You both want to meet as soon as possible, but you live in two different cities that are far apart. To minimize travel time, you agree to meet at an intermediate city, and then you simultaneously hop in your cars and start driving toward each other. But where *exactly* should you meet?

You are given a weighted graph *G* = (*V*, *E*), where the vertices *V* represent cities and the edges *E* represent roads that directly connect cities. Each edge *e* has a weight *w*(*e*) equal to the time required to travel between the two cities. You are also given a vertex *p*, representing your starting location, and a vertex *q*, representing your friend’s starting location.

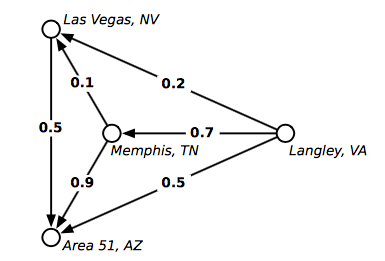
Describe and analyze an algorithm to find the target vertex *t* that allows you and your friend to meet as quickly as possible.

**Challenge 8**

During your Algorithm Analysis final exam, you are given a specific undirected graph *G* = (*V*, *E*) with non- negative edge weights, and you are asked to compute both the minimum spanning tree of *G* and the tree of shortest paths in *G* rooted at some fixed source vertex *s*.

Two and a half hours into the exam, Sarfraz announces that there was a mistake in the exam; every edge weight should be increased by 1. Well, that’s just great. Now what?

1. Do you need to recompute the minimum spanning tree? Either prove that increasing all edge weights by 1 cannot change the minimum spanning tree, or give an example where the minimum spanning tree changes.
2. Do you need to recompute the shortest path tree rooted at *s*? Either prove that increasing all edge weights by 1 cannot change the shortest path tree, or give an example where the shortest path tree changes.

**Challenge 9** 

Mulder and Scully have computed, for every road in the United States, the exact probability that someone driving on that road *won’t* be abducted by aliens. Agent Mulder needs to drive from Langley, Virginia to Area 51, Nevada. What route should he take so that he has the least chance of being abducted?

More formally, you are given a directed graph *G* = (*V*, *E*), where every edge *e* has an independent safety probability *p*(*e*). The *safety* of a path is the product of the safety probabilities of its edges. Design and analyze an algorithm to determine the safest path from a given start vertex *s* to a given target vertex *t*.

For example, with the probabilities shown above, if Mulder tries to drive directly from Langley to Area 51, he has a 50% chance of getting there without being abducted. If he stops in Memphis, he has a 0.7 × 0.9 = 63% chance of arriving safely. If he stops first in Memphis and then in Las Vegas, he has a 1 − 0.7 × 0.1 × 0.5 = 96.5% chance of being abducted! (That’s how they got Elvis, you know.) Although this example is a dag, your algorithm must handle *arbitrary* directed graphs.

**Challenge 10**

On an overnight camping trip in Safari National Park, you are woken from a restless sleep by a scream. As you crawl out of your tent to investigate, a terrified park ranger runs out of the woods, covered in blood and clutching a crumpled piece of paper to his chest. As he reaches your tent, he gasps, “Get out. . . while. . . you. . . ”, thrusts the paper into your hands, and falls to the ground. Checking his pulse, you discover that the ranger is stone dead.

You look down at the paper and recognize a map of the park, drawn as an undirected graph, where vertices represent landmarks in the park, and edges represent trails between those landmarks. (Trails start and end at landmarks and do not cross.) You recognize one of the vertices as your current location; several vertices on the boundary of the map are labeled EXIT.

On closer examination, you notice that someone (perhaps the poor dead park ranger) has written a number of vampires at each trail. The note warns you that stepping off the marked trails will result in a slow and painful death and it also says that you can kill only **One Zombie** on your path to safest exit which means you may or may not find a zombie on the path to safest exit.. You glance down at the corpse at your feet. Yes, his death certainly looked painful. Wait, was that a twitch? Are his teeth getting longer? After driving a tent stake through the undead ranger’s heart, you wisely decide to leave the park immediately.

Describe and analyze an efficient algorithm to find a path from your current location to safest EXIT nodes.