Problem Set 11

Graphs: Representation, Traversal

---

1. Suppose a weighted undirected graph has *n* vertices and *e* edges. The weights are all integers. Assume that the space needed to store an integer is the same as the space needed to store an object reference, both equal to one unit. *What is the minimum value of e* for which the adjacency matrix representation would require less space than the adjacency linked lists representation? Ignore the space needed to store vertex labels.

---

2. The complement of an **undirected** graph, **G**, is a graph **GC** such that:
   - **GC** has the same set of vertices as **G**
   - For every edge *(i,j)* in **G**, there is no edge *(i,j)* in **GC**
   - For every pair of vertices *p* and *q* in **G** for which there is no edge *(p,q)*, there is an edge *(p,q)* in **GC**.

   Implement a method that would return the complement of the **undirected** graph on which this method is applied.

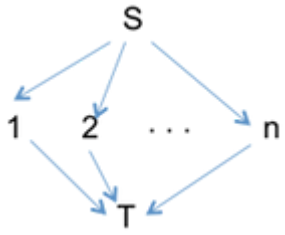
```
class Edge {
   int vnum;
   Edge next;
}

public class Graph {
   Edge[] adjlists;  // adjacency linked lists
   ...
   public Graph complement() {
      // FILL IN THIS METHOD
      ...
   }
}
```

   What would be the worst case running time (big O) of an implementation for a graph with *n* vertices and *e* edges?

---

3. Consider this graph:



   This graph has *n*+2 vertices and 2*n* edges. For every vertex labeled *i*, *1* <= *i* <= *n*, there is an edge from *S* to *i*, and an edge from *i* to *T*.

   1. How many different depth-first search sequences are possible if the start vertex is *S*?
   2. How many different breadth-first search sequences are possible if the start vertex is *S*?

---

4. * You can use DFS to check if there is a path from one vertex to another in a directed graph.

Implement the method **hasPath** in the following. Use additional class fields/helper methods as needed:

```
public class Neighbor {
   public int vertex;
   public Neighbor next;
   ...
}

public class Graph {
   Neighbor[] adjLists;  // adjacency linked lists for all vertices

   // returns true if there is a path from v to w, false otherwise
   public boolean hasPath(int v, int w) {
     // FILL IN THIS METHOD
     ...
   }
}
```