CSCE 221 Cover Page

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more Aggie Honor System Office https://aggiehonor.tamu.edu/

Name	Tianlan Li
UIN	532003637
Email address	rainsuds@tamu.edu

Cite your sources using the table below. Interactions with TAs and resources presented in lecture do not have to be cited.

People	1. None
Webpages	1. None
Printed Materials	1. None
Other Sources	1. None

Homework 3

See the Canvas calendar for the deadline

Typeset your solutions to the homework problems preferably in LaTeX or LyX. See the class webpage for information about their installation and tutorials. There are 6 problems on 6 separate pages.

1. (10 points) An airport is developing a computer simulation of air-traffic control that handles events such as landings and takeoffs. Each event has a *time-stamp* that denotes the time when the event occurs. The simulation program needs to efficiently perform the following two fundamental operations:

   - Insert an event with a given time-stamp (that is, add a future event)
   - Extract the event with a smallest time-stamp (that is, determine the next event to process)

   (a) What data structure should be used to implement the above operations efficiently? Explain your reasoning.

Solution: Minimum Heap. For inserting a new plane based of their time-stamp maintains the heap property in which the parent node is always less than its children. Therefore, the insertion of new child are efficient based on this approach. For extraction, the minimum prioity always has the minimum element at its root, therefore extracting the minimu element is very efficient.

   (b) Provide the big-O asymptotic complexity of inserting a time-stamp into the data structure identified in part 1a

Solution: $O(logn)$

   (c) Provide the big-O asymptotic complexity of extracting a time-stamp from the data structure identified in part 1a

Solution: $O(logn)$

2. (15 points) A *complete* graph is an undirected graph in which every pair of vertices are connected with an edge. Consider the following complete graph with $n = 6$ vertices.

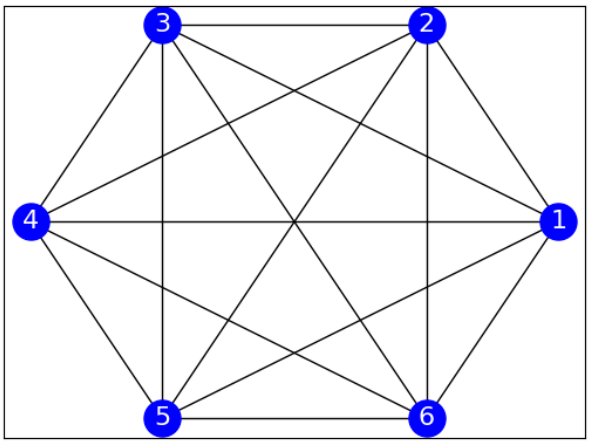


Figure 1: A complete graph with $n = 6$ vertices

(a) In what order does `DFS` explore vertices in the above graph? Assume `DFS` starts at vertex 4. The adjacency lists are in ascending order by the numeric label.

Solution: 4, 1, 2, 3, 5, 6

(b) What is the running time of `DFS` on a complete graph with n vertices? Provide an asymptotic big-oh bound in terms of the number of vertices. Explain your reasoning.

Solution: $O(V + E)$ where V is the number of vertices and E is the number of edges. The total number of edges can be written as $\frac{n(n-1)}{2}$ since the vertices in a complete graph is connected to all other vetices. Therefore $O(V + E)$ can be written as $O(n + \frac{n(n-1)}{2}) = O(n^2)$

(c) How many back edges, forward edges, cross edges, and exploratory edges are generated by running `DFS` on a complete graph with n vertices?

Solution: Back Edges: Back edges will be generated by every edge that is not a tree edge. Since tree edges are given by $n-1$, and the total number of edges is given by $\frac{n(n-1)}{2}$. Therefore, back edges are given by $\frac{n(n-1)}{2} - (n-1)$, which is $\frac{n^2-3n-2}{2}$.

Forward Edges: Forward edges are edges that connect a vertex to a descendant in the DFS tree. Since every possible connection exists, every edge that leads to a yet-to-be-visited vertex can be considered a forward edge. However, in the DFS process, each vertex is visited exactly once, and once a vertex is visited, all its edges are explored. Therefore, there are no forward edges in this context.

Cross Edges: Cross edges are edges that are not part of the DFS tree and they connect vertices such that neither is an ancestor of the other in the DFS tree. Since every pair of vertices is connected and DFS visits each vertex exactly once, there are no cross edges.

Exploratory (Tree) Edges: The DFS will generate a total of $n - 1$ exploratory edges in a complete graph. This is because in a DFS spanning tree of a connected graph, there are always $V - 1$ edges, where V is the number of vertices.

3. (10 points) Answer each of the following questions with a tight big-O asymptotic bound. Justify with algorithmic reasoning.

   (a) A priority queue, `UnsortedMPQ`, is implemented based on an unsorted array. What is the running time of the operation which retrieves the minimum value?

   $$\texttt{unsorted\_min(n)} \in O(\ldots)$$

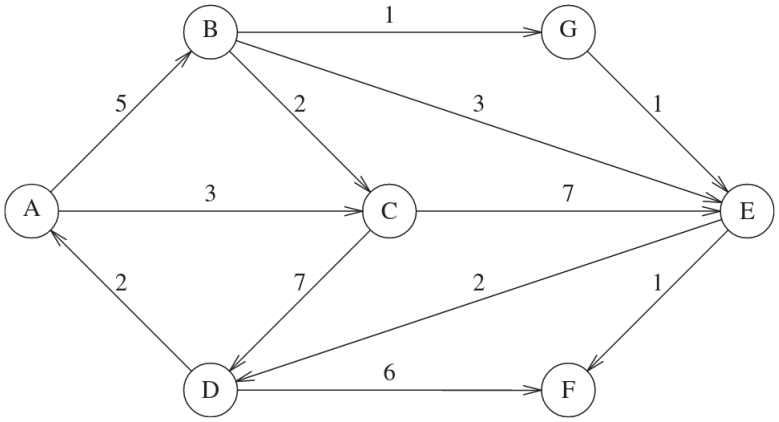
Solution: $O(n)$. Since each element would have to be visited exactly once to determine the minimum value of the array.

   (b) Dijkstra's algorithm is implemented based on this unsorted minimum priority queue. What is the running time of this implementation of Dijkstra's algorithm?

   $$\texttt{dijkstra(V,E)} \in O(\ldots)$$

Solution: $O(V^2)$. Since the running time for retriving min element is $O(n)$, and unsorted MPQ is used to implement Dijkstra's algorithm in this case. Each vertex is visited exactly once after extracting the minimum value from the queue, and each vertex that is called visits the number of edges of unvisited vertices. Therefore, the running time of this algorithm is given by $O(V^2 + E)$. E in this case, cannot be larger than the number of V, $E = V - 1$. So the running time for this algorithm implemented using MPQ is $O(V^2 + V - 1) = O(V^2)$.

4. (15 points) Find the shortest path from D to all other vertices for the graph below.



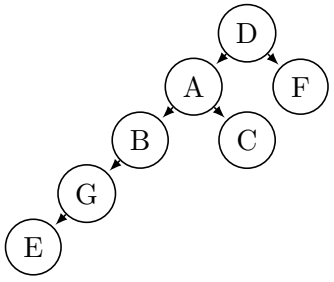
(a) Illustrate the minimum priority queue at each iteration Dijkstra's algorithm.

Solution:

MPQ Dijkstra								
num	Pop	A	B	C	D	E	F	G
0	-	∞	∞	∞	0	∞	∞	∞
1	D	2^D	∞	∞		∞	6^D	∞
2	A		7^A	5^A		∞	6^D	∞
3	C		7^A			12^C	6^D	∞
4	F		7^A			12^C		∞
5	B					10^B		8^B
6	G					9^G		
7	E							

(b) Draw the Shortest Path Tree.

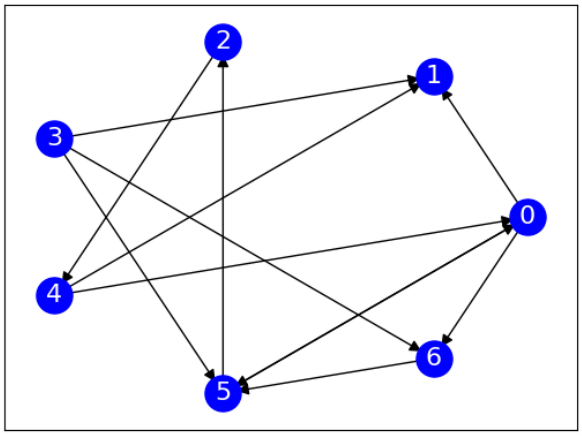
Solution:



(c) What is the running time of the Dijkstra's algorithm under the assumption that the graph is implemented based on an adjacency list and the minimum priority queue is implemented based on a binary heap?

Solution: The running time of Dijkstra's algorithm under these assumptions are $O((V+E)logV)$ where V is the number of vertices and E is the number of edges. Initialization of each vertex in a graph takes $O(V)$ total. Using Minimum Priority Queue to extact each elements takes total of $O(VlogV)$. The relax function is called for every adjacent edges of a vertex, which is $O(ElogV)$. The total running time is $O((V+E)logV)$.

5. (15 points) Find the shortest path from vertex 3 to all other vertices for the graph below.



(a) Which graph algorithm can solve the problem most *efficiently*?

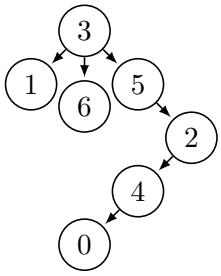
Solution: BFS.

(b) How could you use the same algorithm if the graph had edge weights? (*Hint*: You may want to create intermediate nodes.)

Solution: Using intermediate nodes, for each edge in the original graph with a weight greater than 1, introduce intermediate nodes to break the edge into several edges of weight 1. For example, if there's an edge of weight 3 between Node A and Node B, you would remove this edge and add two intermediate nodes, X and Y. You would then create edges A-X, X-Y, and Y-B, each with a weight of 1. After Transforming all the edges into weights of 1, apply BFS on the new graph, then transform the path with intermediate nodes removed.

(c) Draw the Shortest Path Tree.

Solution:



6. (15 points) There are five small islands in a lake, and the state wants to build four bridges to connect them so that each island can be reached from any other one via one or more bridges. The cost of bridge construction is proportional to its length. The distance between pairs of islands are given in the following table.

	1	2	3	4	5
1	-	10	15	10	20
2	-	-	15	20	20
3	-	-	-	15	30
4	-	-	-	-	10
5	-	-	-	-	-

Table 1: The distance between any two islands

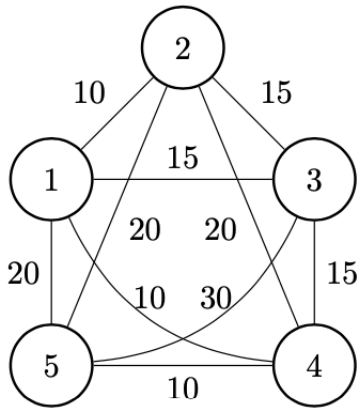


Figure 2: The distance between any two islands

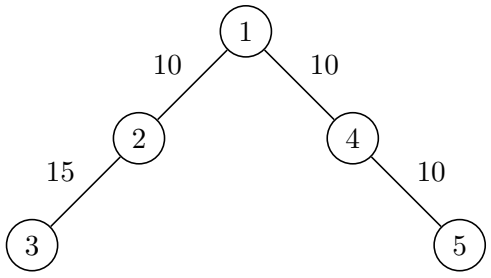
(a) Illustrate the steps of Prim's algorithm using the graph above. Draw the Minimum Spanning Tree. What is the length of the bridges?

Solution: Using Prim's algoritm:

1. Start with any vertex: Let's start with island 1.
2. Choose the smallest edge connecting to a new vertex: The smallest edge from island 1 is to island 2 (10 units).
3. Repeat the process: Now, we consider the edges from both island 1 and island 4.
4. Keep adding the smallest edge that connects a new vertex: Continue this process until all vertices are included.

The total length of bridges is $10 + 10 + 10 + 15 = 45$.

Prim						
	Pop	1	2	3	4	5
1	(1,2)	-	10	15	10	20
2	(2,3)	-	-	15	20	20
3	(1,4)	-	-	-	15	30
4	(4,5)	-	-	-	-	10
5		-	-	-	-	-



(b) Illustrate the steps of Kruskal's algorithm using the graph above. Draw the Minimum Spanning Tree. What is the length of the bridges?

Solution: Using Kruskal's Algorithm:

1. Sort all the edges by length:
{(1,2): 10, (1,4): 10, (4,5): 10, (1,3): 15, (2,3): 15, (3,4): 15, (1,5); 20, (2,5): 20, (3,5): 30 }

2. Add the shortest edge, (1,4): 10 to the MST. 3. Add the next shortest edge without forming a cycle to the MST, $(4,5) : 10$ and $(1,2) : 10$. Since (1,3) creates a cycle with $(1,2)$ and $(1,3)$.

4. Add the final edge to connect all the islands, $(2,3) : 15$. The total length of bridges is also $10 + 10 + 10 + 15 = 45$.

Kruskal									
	(1,2)	(1,4)	(4,5)	(1,3)	(2,3)	(3,4)	(1,5)	(2,5)	(3,5)
Weight	10	10	10	15	15	15	20	20	30
Insertion	Y	Y	Y	N	Y	N	N	N	N
Order	1	2	3		4