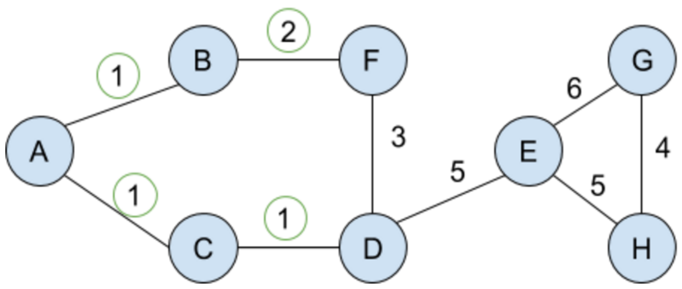
CS 61B Spring 2017 Guerrilla Section 6 Worksheet

15 April 2017

Directions: In groups of 4-5, work on the following exercises. Do not proceed to the next exercise until everyone in your group has the answer and *understands why the answer is what it is.* Of course, a topic appearing on this worksheet does not imply that the topic will appear on the midterm, nor does a topic not appearing on this worksheet imply that the topic will not appear on the midterm.

1 MSTs

(1)



Consider the undirected graph above. We are trying to find the minimum spanning tree (MST) of the graph. The edges with their weight labels circled have already been added to our MST.

(a) What is the next edge to be added to our MST if we are using Kruskal's Algorithm?

(b) What is the next edge to be added to our MST if we are using Prim's Algorithm?
Note: part (a) and part (b) are not related. Don't consider the edge that you added in part (a).

(c) What is the weight of the complete MST?

(2) Consider a graph with negative edges.

(a) How would we modify Kruskal's Algorithm to find a MST on this graph?

(b) We now want to find a minimum spanning graph (no longer needs to be a tree) for this graph. How would we modify Kruskal's Algorithm to find the minimum spanning graph for this graph?

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

2 To Cycle Or Not To Cycle

(1) Given an undirected graph $G = (V, E)$ and an edge $e = (s, t)$ in G, create an $O(V + E)$ time algorithm to determine whether G has a cycle containing e. No code needed. Just describe.

(2) (Extra for Experts. Skip this and come back if you have time)
Given a connected, undirected, weighted, graph, describe an algorithm to construct a set with as few edges as possible such that if those edges were removed, there would be no cycles in the remaining graph. Additionally, choose edges such that the sum of the weights of the edges you remove is minimized. This algorithm must be as fast as possible.

3 Start To Finish

You're given an undirected, positively weighted graph $G = (V, E)$, a list of start vertices S, and a list of end vertices T. Describe an efficient algorithm that returns the shortest path, such that the path starts at one vertex from S and ends at one vertex from T. Hint: Consider adding dummy nodes to the graph to reduce this problem into something simpler.

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

4 One Path to Traverse them All, and Topological Sort Them

Given a directed acyclic graph G, write an algorithm that determines if G contains a path that goes through every vertex exactly once. Briefly justify why the algorithm is correct, and state the runtime.

Assume that the graph is implemented with the following API, where nodes are represented by integers.

```
public class Graph {
        // Returns true if this graph has an edge from u to v.
        public boolean hasEdge(int u, int v);

        // Returns a list of integers, in a topologically sorted order for this graph;
        // implemented in the way described in lecture.
        public List<Integer> topologicalOrder();
}

// Please implement your algorithm in this method:
public boolean onePath(Graph g) {





}
```

Justification:

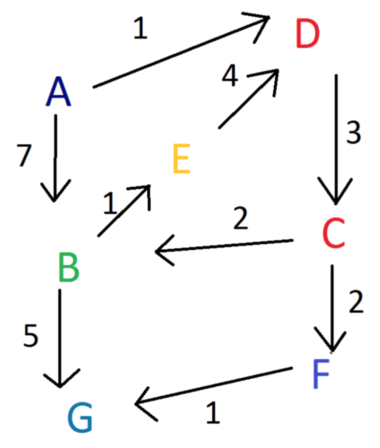
Runtime (in Θ notation):

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

5 The Amazing Race!

Directions: With the United States as your starting point, your goal is to travel to Greece as fast as you can. However, there is a twist such that you can only travel the paths shown on the graph toward Greece. (Note: This graph was not designed by a geographer.)



Letter in graph	Name of location	Heuristic to End
A	America (USA)	6
B	Brazil	4
C	China	7
D	Dominican Republic	7
E	Egypt	2
F	France	6
G	Greece	0

(a) Run Dijkstra's algorithm to find the shortest paths to all destinations starting from A, breaking ties alphabetically if necessary.

(b) Use A* search to find the optimal path to Greece, breaking ties alphabetically if necessary.

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

6 Which Sort to Use?

For each of the following scenarios, choose the best sort to use and explain your reasoning.

(a) The list you have to sort was created by taking a sorted list and swapping N pairs of adjacent elements.

(b) You have to sort a list on a machine where swapping two elements is much more costly than comparing two elements (and you want to do the sort in place).

(c) Your list is so large that not all of the data will fit into RAM at once. As is, at any given time most of the list must be stored in external memory (on disk), where accessing it is extremely slow.

(d) Given a list of emails ordered by send time, sort the list such the emails are ordered by the sender's name first while secondarily maintaining the time-ordering.

(e) You have a randomly shuffled list of numbers unbounded in size and want to sort the elements.

(f) Suppose you're designing a secure system that needs to defend against *adversarial inputs*. An attacker can give you any list they choose and understand exactly how your sorting algorithms are implemented.

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

7 Empirical Analysis

Andrew, Kevin, and Fahad performed timing tests on several sorting algorithms: selection sort, insertion sort, merge sort, and tree sort (repeated insertions into a binary search tree with no attempt to balance, followed by a traversal of the tree). They timed each sorting algorithm on several datasets of 2000 values. Unfortunately, Andrew forgot to label each experiment with its sort! Help them out by figuring out which times match with which sorting method.

Time to sort 2000 random values	Time to sort 2000 values already in increasing order	Time to sort 2000 values already in decreasing order	Sorting method (Selection, Insertion, Merge, or Tree)
1098	29	1685	
183	1624	1570	
191	207	195	
1698	1776	1734	

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

8 What's that Sort!?

Which sorting algorithms do the following illustrate? Your options are merge sort, insertion sort, selection sort, heap sort, quick sort. Algorithms illustrated may not conform exactly to those presented in discussion and in lecture. Please note that each of these are snapshots as the algorithm runs, not all iterations of its running.

(a)
```
5103 9914 0608 3715 6035 2261 9797 7188 1163 4411
0608 1163 5103 3715 6035 2261 9797 7188 9914 4411
0608 1163 2261 3715 6035 5103 9797 7188 9914 4411
```

(b)
```
5103 9797 0608 3715 6035 2261 9914 7188 1163 4411
0608 3715 2261 1163 4411 5103 9797 6035 9914 7188
0608 3715 2261 1163 4411 5103 6035 7188  9797 9914
```

(c)
```
dze ccf hwy pjk bkw xce aux qtr
ccf dze hwy pjk bkw xce aux qtr
ccf dze hwy pjk aux bkw qtr xce
aux bkw ccf dze hwy pjk qtr xce
```

(d)
```
dze ccf bkw hwy pjk xce aux qtr xpa atm
dze ccf bkw hwy pjk xce aux qtr atm xpa
dze ccf bkw hwy pjk xce aux atm qtr xpa
dze ccf bkw hwy pjk xce atm aux qtr xpa
dze ccf bkw hwy pjk atm aux qtr xce xpa
dze ccf bkw hwy atm aux pjk qtr xce xpa
dze ccf bkw atm aux hwy pjk qtr xce xpa
dze ccf atm aux bkw hwy pjk qtr xce xpa
dze atm aux bkw ccf hwy pjk qtr xce xpa
atm aux bkw ccf dze hwy pjk qtr xce xpa
```

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!