Final Review
Big-Oh Analysis

• Must be able to analyze algorithms in final (second half of class material) using Big-O Notation
Sorting

- Simple Sorts: Insertion Sort, Selection Sort
- Heap Sort
- Merge Sort
- Quick Sort
- Lower Bound for Comparison-based Sorting
- Bucket Sort & Radix Sort
Graphs

- Graph Basics & Representations: Adjacency List & Adjacency Matrix
- Graph Traversals
- Topological Sort
- Dijkstra's Algorithm for Finding Shortest Paths
- Prim's Algorithm for Finding Minimum Spanning Trees
- Kruskal's Algorithm for Finding Minimum Spanning Trees
Algorithms

• Greedy vs Divide and Conquer vs Dynamic Programming

• Examples of each
P = NP?

- Hard and easy problems to solve
- Reductions
- Class P
- Class NP
- NP-Complete
Parallelism

- ForkJoin Parallelism, and Associated Terms (Work, Span, etc.)
- ForkJoin Applications, ex: Parallel Summing of an Array
- Amdahl's Law
Example problems
Use the following graph for this problem. Where needed and not determined by the algorithm, assume that any algorithm begins at node A.

```
Use the following graph for this problem. Where needed and not determined by the algorithm, assume that any algorithm begins at node A.

```

a)  (4 pts) Draw both the adjacency matrix and adjacency list representations of this graph. Be sure to specify which is which.

b)  (2 pts) Give two valid topological orderings of the nodes in the graph.
c) (4 pts) Step through Dijkstra’s Algorithm to calculate the single source shortest path from A to every other vertex. You only need to show your final table, but you should show your steps in the table below for partial credit. Show your steps by crossing through values that are replaced by a new value. Note that the next question asks you to recall what order vertices were declared known.

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Known</th>
<th>Distance</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

d) (1 pts) In what order would Dijkstra’s algorithm mark each node as known?

e) (1 pt) What is the shortest (weighted) path from A to D?

f) (1 pt) What is the length (weighted cost) of the shortest path you listed in part (e)?

g) (3 pts) Imagine that the graph were undirected (i.e., ignore the directions of the edges). Write the edges considered by Kruskal’s algorithm in the order they are considered. Assume the algorithm terminates as soon as the MST has been completed. Write an edge between vertices A and B as (A,B).
Question 7. (4 points) Quicksort has an expected sorting time of $O(n \log n)$ provided that the pivot value is chosen with some care.

(a) What needs to be true about the choice of pivot value to ensure that quicksort has an expected time of $O(n \log n)$?

(b) Give one effective strategy for picking pivot values that will ensure that the condition needed in part (a) will usually be met to ensure that quicksort runs in $O(n \log n)$ time.
1) [10 points total] Sorting: (Assume that array `sun[]` has indices: 0 to `size-1`)

```java
SunnySort(int[] sun) {
    for (int i = 1; i < size; i++) {
        int j;
        int temp = sun[i];
        for (j = i; j > 0 && temp < sun[j-1]; j--) {
            sun[j] = sun[j-1];
        }
        sun[j] = temp;
    }
}
```

a) [2 points] This is actually a sort mentioned in class. What sort is this?

b) [4 points] Describe the best and worst case data set for this sort. (If all cases behave similarly, please state that.) What is the big-O running time of those two data sets?

Best Case data set?

Best Case running time?

Worst Case data set?

Worst Case running time?

c) [2 points] Is it an in-place sort? Why or why not?
(no credit without a reason or a definition of in-place, for partial credit define in-place sorting)

d) [2 points] Is it a stable sort? Why or why not?
(no credit without a reason or definition of stable, for partial credit define stable sorting)