Final Review Document

CS 61B Spring 2018

Antares Chen + Kevin Lin

Introduction

Wow this semester has gone by really fast. But before you guys can finish up this class and beat the game, there is one last boss - the final! The final is going to be cumulative across all the topics we've covered this semester. Anything that was mentioned in class is fair game.

Now, this packet will provide supplementary problems on topics covered after midterm 2. Like all previous packets, these questions are ranked by difficulty, but in no way is this packet comprehensive! Most notably, it lacks coding challenges since it's targeted to test your conceptual understanding of the topics. Further, it lacks a few hard questions since the questions I've thought of would probably detract from your studying.

For this final, I want you all to be smart, use your best judgement, and do the work! You have a little time left to really nail down this studying thing and with that I believe you guys have a great chance to really knock this test out of the ballpark. You may have faltered at times during this semester and things might have been hard. But like the great fisherman standing in the frozen sea once said, "**Never give up!**"

Think of all the people cheering you on.

Go out, read the textbook, find practice midterms and really work through them with your friends. Talk to the TA's, go to office hours, and use all the resources around you. Needless to say it is quite useless to continue banging your head against the books if the material doesn't stick.

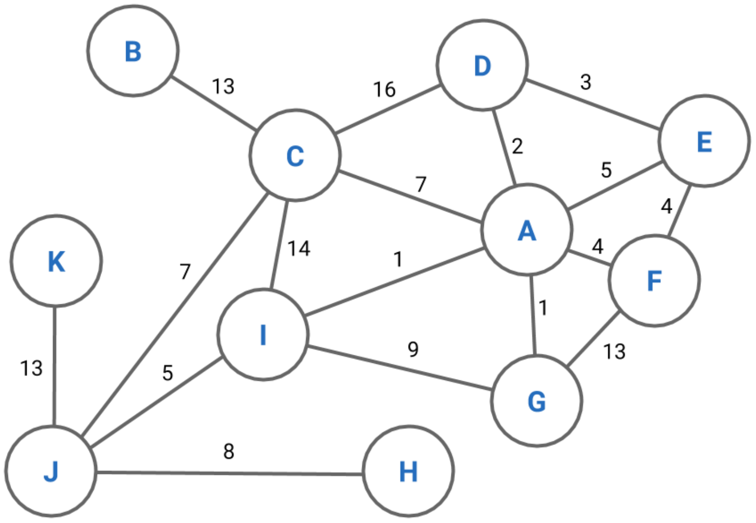
Finally, if you find things becoming too stressful, simply step back, take a deep breath and go out for a long walk. Come back, remember that you're awesome and get back to work. Believe in yourself! And if you don't then believe in us who believe in you.

“Yesterday you said tomorrow; so just do it.” - Abraham Lincoln

Graphs

Easy Mode

Traversal Time Write out the listed traversals starting at node A. Break ties alphabetically.



1) Breadth First Search

2) Depth First Search

3) Dijkstra’s

Runtime Funtime (More Fun Times) For each algorithm, list the worst case runtime.

BFS	DFS	Dijkstra’s	A*	Topological Sort

And for each representation fill in the runtimes.

	Storage size	Add vertex	Add edge	Remove vertex	Remove edge	Query if edge exists
Adjacency List						
Adjacency Matrix						

Last Question When would you use an adjacency matrix over an adjacency list?

Medium Mode

Graph Questions Answer if each problem is always true, sometimes true, or never true.

1) Given a graph with V vertices and V-1 edges, adding an edge would introduce a cycle

2) A DFS traversal starting at vertex u ending at v always finds the shortest path in an unweighted graph.

3) Dijkstra's finds the shortest path for a graph with negative edge weights.

4) A DFS on a directed acyclic graph produces a topological sort.

Dijkstra's Algorithm The runtime for Dijkstra's algorithm is O((V + E) log (V)); however this is specific only to binary heaps. Let's provide a more general runtime bound. Suppose we have a priority queue backed by some arbitrary heap implementation. Given that this unknown heap contains N elements, suppose the runtime of remove-min is f(N) and the runtime of change-priority is g(N).

1) What is the runtime of Dijkstra's in terms of f(V) and g(V)?

2) Turns out the optimal version of Dijkstra's algorithm uses something called a fibonacci heap. The fibonacci heap has amortized constant time change-priority, and log time remove-min. What is the runtime of Dijkstra's algorithm?

Hard Mode

Algorithm Design Provide an algorithm matching the given time bound for each problem.

1) Suppose you are terribly lost in downtown SF (basically a maze), but happen to have an infinite amount of pennies (you made a killing investing in Kelp, a hot new Yelp-for-seafood startup that recently IPO'd). Discuss how you would get out of SF by solving the following problem.

   Let G be an undirected connected graph. Give an O(V + E) time algorithm to compute a path in G that traverses each edge in E exactly once in each direction.

2) A directed graph G is singly connected if for all vertices u, v, u connected to v implies that there is at most one simple path from u to v. Provide an efficient algorithm to determine whether or not a directed graph is singly connected

Dynamic Programming

Easy Mode

Count... Stairways? Rewrite this exponential-time procedure as a dynamic program.

```
public static int slowlyCountStairWays(int n) {
    if (n == 1) {
        return 1;
    } elif (n == 2) {
        return 2;
    } else {
        return slowlyCountStairWays(n - 1) + slowlyCountStairWays(n - 2);
    }
}

public static int quicklyCountStairWays(int n) {
    if (n == 1) {
        return 1;
    }















}
```

Medium Mode

Tree Recursion <3, Never Change Once the machines take over, the denomination of every coin will be a power of two: 1-cent, 2-cent, 4-cent, 8-cent, 16-cent, etc. There will be no limit to how much a coin can be worth.

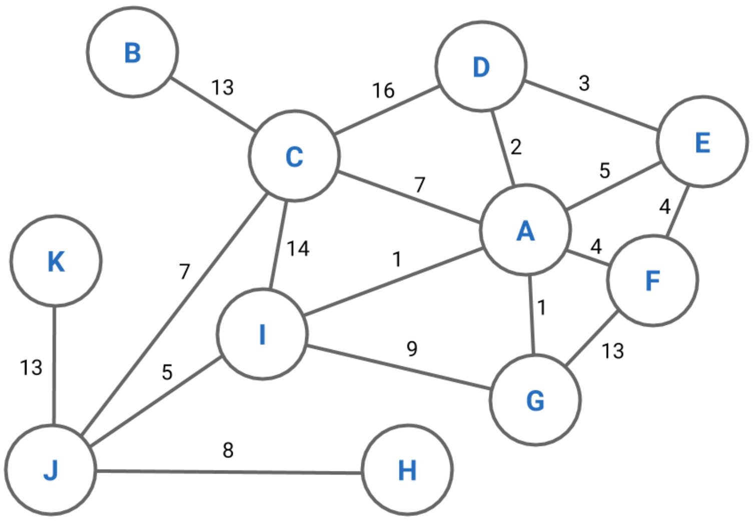
Write a bottom-up dynamic program, `countChange`, that takes a positive integer `n` and returns the number of ways to make change for `n` using these coins of the future. Compute bottom-up; do not use memoization.

```
public static int countChange(int n) {











}
```

Disjoint Sets and MST

Easy Mode

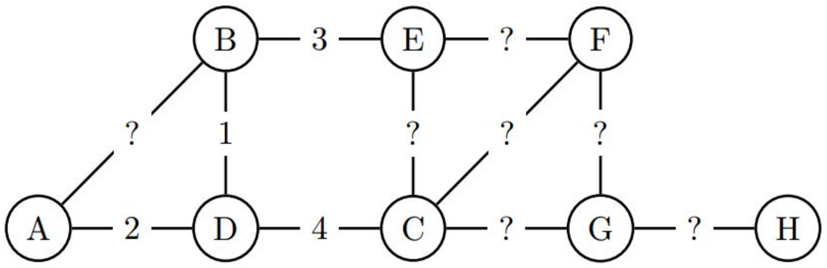
Run Forest Run For the following graph, draw the MST.



Now run Kruskal's algorithm using a disjoint set structure on the graph above. After 5 iterations, what does the disjoint set data structure look like?

Medium Mode

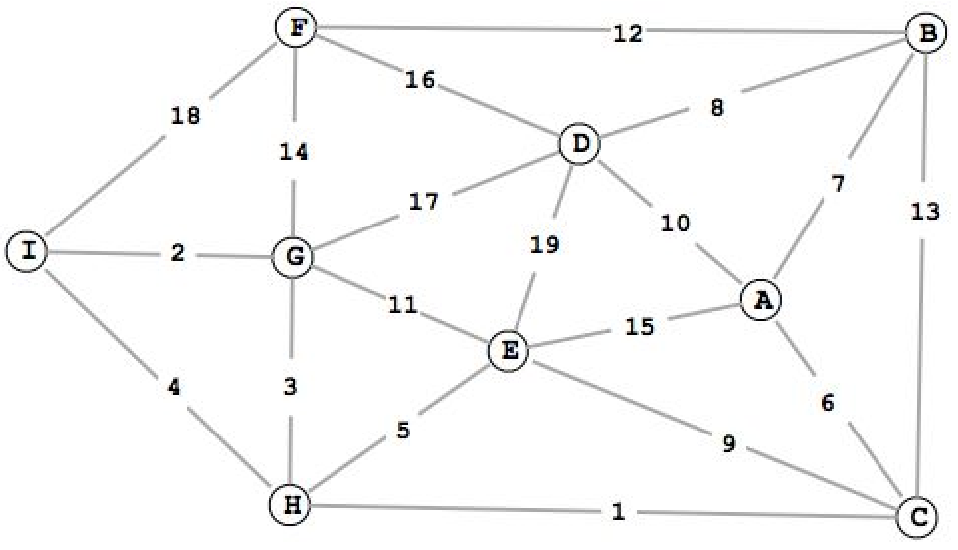
Hidden Weight In this graph, some of the edge weights are known, while the rest are unknown.



$$cost(A, D) = 2, cost(B, D) = 1, cost(C, D) = 4, cost(B, E) = 3$$

List all edges that must belong to a minimum spanning tree, regardless of what the unknown edge weights turn out to be. Justify your answers.

Princeton Schminceton Answer the following questions for the graph below. Note that the edge weights are distinct with values ranging from 1 to 1.



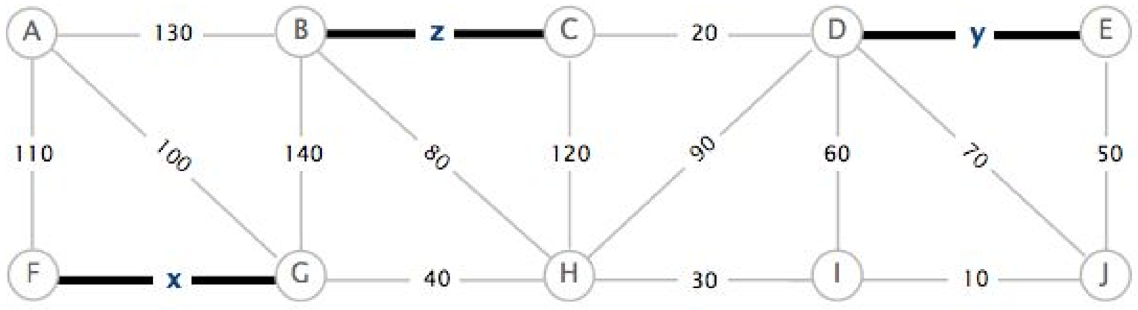
1) Complete the sequence of edges in the MST in the order that Kruskal's algorithm includes them.

   1 ____ ____ ____ ____ ____ ____ ____

2) Suppose that the edge D-I of weight w is added to the graph. For which values of w is the edge D-I in a MST?

3) Given a minimum spanning tree T of a weighted graph G, describe an O(V) algorithm for determining whether or not T remains a MST after an edge (x, y) of weight w is added.

Princeton Part 2 Suppose that a MST of the following edge-weighted graph contains the edges with weights x, y, and z.



1) List the weights of the other edges in the MST in ascending order of weight

    10 ____ ____ ____ ____ ____

2) Circle which one or more of the following can be the value of x?

    5 15 25 35 45 55 65 75 85 95 105 115 125 135 145

3) Circle which one or more of the following can be the value of y?

    5 15 25 35 45 55 65 75 85 95 105 115 125 135 145

4) Circle which one or more of the following can be the value of z?

    5 15 25 35 45 55 65 75 85 95 105 115 125 135 145

Constant Kruskal's Suppose that for a given graph G, the weights of its edges are in the range 1 to V where V is the number of vertices. How fast can Kruskal's algorithm run?

What if the edge weights are integers in the range 1 to W where W is an asymptotic variable?

Hard Mode

Random MST Properties State if the following are true or false. If they are true, provide an informal argument else provide a counterexample.

1) Suppose G is a graph and T is a MST of G. If we decrease the weight of any edge in T, then T is still a MST.

2) A graph has a unique minimum spanning tree if, for every cut of the graph (every way you can split a graph into two separate parts), there is a unique light edge crossing the cut.

Update MST Suppose instead of decreasing the weight of an edge in the MST, we decrease the weight of any random edge not in the MST. Give an efficient algorithm for finding the MST in the modified graph.

Out of Sorts

Easy Mode

Classify the Sorts Answer the following questions assuming you have some input lists where N is the list's size.

1) List all the sorts that run worst case O(N^2).

2) List all the sorts that run worst case O(N log N).

3) List all the stable sorts.

Fill the Table Fill in the cells of the following table.

Algorithm	Best case runtime	Worst case runtime	Best case example	Worst case example
Heap Sort				
Quicksort				
Merge Sort				
Selection Sort				
Insertion Sort				
Radix Sort				

Run the Sorts Listed below are partially sorted lists, that would occur in the middle of running a sort on the input list. For each partially sorted list, name the algorithm that may have sorted it. Choices include heapsort, quicksort (assuming the first element as the pivot), mergesort, insertion sort, LSD radix sort, MSD radix sort and selection sort.

1) 12, 7, 8, 4, 10, 2, 5, 34, 14
   7, 8, 4, 10, 2, 5, 12, 34, 14
   4, 2, 5, 7, 8, 10, 12, 14, 34

2) 23, 45, 12, 4, 65, 34, 20, 43
   4, 12, 23, 45, 65, 34, 20, 43

3) 12, 32, 14, 11, 17, 38, 23, 34
   12, 14, 11, 17, 23, 32, 38, 34

4) 45, 23, 1, 65, 34, 3, 76, 25
   23, 45, 1, 65, 3, 34, 25, 76
   1, 23, 45, 65, 3, 25, 34, 76

5) 23, 44, 12, 11, 54, 33, 1, 41
   54, 44, 33, 41, 23, 12, 1, 11
   44, 41, 33, 11, 23, 12, 1, 54

Average Case Runtimes What are the average case runtimes for quicksort and insertion sort?

Medium Mode

Conceptual Questions

1) Give an example of when insertion sort is more efficient than mergesort.

2) When would you use mergesort over quicksort?

3) When would you use radix sort over a comparison sort?

Design Questions

1) You have an array of integers where the length = N and the maximum number of digits is given by K. Suppose that K >> N (that is K is much larger than N). What sort would you use?

2) Assume for the previous question that you used LSD radix sort. What is the runtime?

3) Suppose you are given an integer array with length = N such that there are N - sqrt(N) copies of the minimum element located in the front of the array. What is the worst case runtime of insertion sort on this list?

4) For the same construction, what is the worst case runtime of selection sort?

Hard Mode

CS 61B Lane

The rent is too dang high in Berkeley and so you decide to buy a house located at the idyllic CS 61B Lane. CS 61B Lane is, of course, a friendly neighborhood and so everyone who lives on CS 61B Lane is automatically friends with her two closest neighbors.

For example, on CS 61B Lane we may have houses for: Aidan, Kevin, Kaylee, Ching, Alex, and Matt[1]. Here Kaylee is friends with Aidan, Kevin, Ching and Alex while Alex is friends with Kaylee, Ching, and Matt.

Let us represent CS 61B Lane as an array of House objects. Each House object has two instance variables: the owner's name and the address. Now given an array of N houses in *no particular order*, give an algorithm that returns a data structure such that when you query the data structure with a person's name, it will return to you a list of friends.

Suppose you are given an array of objects *House* in no particular order. Each House object has two fields: the owner's name and the address. Now suppose we are given the array for CS 61B Lane. However, CS 61B Lane is a friendly neighborhood, so every owner is friends with the two closest houses.

Provide a the best possible. runtime bound for construction of and querying from the data structure.

[1] Antares actually lives in a box off the side of Euclid...

Across The Vast
Expanse of Space
Memes Echo
Into Eternity