CS251 Homework 4: Graphs and Strings

Out: March 30, 2018 @ 9:00 PM
Due: April 6, 2018 @ 11:59 PM

Submission Instructions: Please submit a typeset PDF on blackboard. For multiple choice questions, you must provide an explanation along with your answer. Answers without explanations will receive 0 points, even if correct.

1. For each of the following scenarios, what graph representation should you choose, and why? Assume no self-loops or parallel edges.

a. An undirected graph G with n vertices and m edges, where n >> m, which must frequently add and remove vertices. **A Tree would be best because you can easily add and remove vertices.**

b. A very large, dense, weighted undirected graph that doesn't change, for which you must compute all-pairs shortest paths. **A BFS would be the best because it is able to find a short path easily.**

c. A weighted undirected graph for which you wish to generate a minimum spanning tree using Prim-Jarnik's algorithm. **An adjacency matrix would be the best because it is able to store the information.**

2. Match the algorithms to the goals below. Note that this can be a many-to-many mapping. (put the goal number next to the algorithm)

Algorithms:

DFS	**1, 2, 4, 7**
BFS	**1, 2, 7**
Floyd-Warshall	**1, 2, 5**
Prim-Jarnik	**1, 3**
Bellman-Ford	**1, 6,**
Dijkstra	**1, 6**
Kruskal	**1, 3**

Goals:

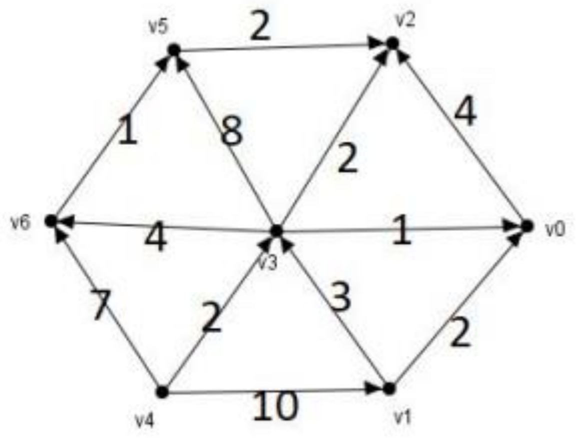
1. Graph traversal
2. Transitive closure
3. Minimum spanning tree
4. Find cycles
5. All-pairs shortest paths
6. Single-source shortest paths
7. Get connected components

3. Assume a directed graph G with n vertices and m edges, in which the weight of all edges are equal. Which of the following time complexities is needed to compute the shortest path from a vertex V to every other vertex in G?

a. O(nm)
b. O(n + m)
c. O(n^2)
d. O(n log m)
e. None of the above

The answer is B. This is because if all edges are the same weight we can use a BFS to find the shortest path. With that being the case the time complexity of a BFS would be O(n+m).

The following weighted graph is used in the questions below.



4. What is the weight of the shortest path from v_4 to v_5? Show your work.

a. 2
b. 4
c. 7
d. 8

The answer is C. This is because to get to v5 you must either traverse though v3 or v6 from v4. Looking at the different paths, there are 3 options. V4 to v3 to v5 is option 1, v4 to v6 to v5 is option 2, and v4 to v3 to v6 to v5 is option 3. Option 1 provides a total of 10, option 2 provides a total of 8, and option 3 provides a total of 7 which is the shortest path.

5. If Dijkstra is started from v_4, which vertex would be the last one to be visited? Show your work.

a. v_0
b. v_1
c. v_2
d. v_5

The answer is B. Starting at v4 you would check v1, v3, and then v6. Then pick v3 as it is the shortest path, labeling v1 as 10, v3 as 2, and v6 as 7. Next we check v0, v2, v5, then v6. We would label v0 as 3 v2 as 4, v5 as 10, and v6 as 6. Then traverse to v0. Check v2, it's a 7 length path which is not shorter so it's not updated. We next go to v2 as it is the shortest path from the source and it has no paths from it so we move to the next node which is v6. From there we check v5 which would update it to 7. We then traverse to v5 as it is the next shortest path from source. We check its outgoing path to v2 get a value of 9 which is greater than the original value so we scrap it. We then go to v1 as the next node. We look at its outgoing path to v0 and then v3, neither are shorter paths than previous. By going through all these steps the last vertex to be visited would be v1. This table below illustrates the shortest paths taken from the source as well.

	V4	**V0**	**V1**	**V2**	**V3**	**V5**	**V6**
d	**0**	**3**	**10**	**4**	**2**	**7**	**6**
pi	**-**	**V3**	**V4**	**V3**	**V4**	**V6**	**V3**

6. If the directions of the edges are removed, what would be the weight of the minimum spanning tree? Show your work.
    a. 8
    b. 10
    c. 11
    d. 12

The answer is B. Starting at v3 go to v4 giving a total of 2. Then go back to v3 and go to v0 then v1 to total 5. Next go back to v3 and go to v2, then v5, then v6 giving a total of 10.

7. How many topological orders does the graph have? Show your work.
    a. 1
    b. 3
    c. 4
    d. 6

The answer is D. The first one can be found doing a left to right, top to bottom. The second is smallest numbered available vertex first. The third is the fewest edges first. The fourth would be the largest numbered available vertex first. The fifth would be attempting top to bottom, left to right. And the last one would be an arbitrary sorting. Giving us 6 total topological orders

8. The brute-force pattern matching algorithm on the slides runs until either
    a. A match is found, or
    b. All placements of the pattern have been tried

Write the pseudocode for an algorithm such that it reports all occurrences of a pattern P with the number of shifts needed to achieve that.

```
Algorithm BruteForceMatch(T, P)
Input text T of size n and pattern P of size m
Output starting index of a substring of T equal to P or −1 if no such substring exists
for i ← 0 to n − m
     { test shift i of the pattern }
     j ← 0
     while j < m ∧ T[i + j] = P[j]
               j ← j + 1
          if j = m
               return i {match at i}
          else
               break while loop {mismatch}
return -1 {no match anywhere}
```

9. Show the comparisons the naive string matcher makes for the pattern P = 0001 in the text T = 000010001010001

```
S=0  000010001010001
     0001 mismatch
S=1  000010001010001
      0001 match
S=2  000010001010001
       0001 mismatch
S=3  000010001010001
        0001 mismatch
S=4  000010001010001
         0001 mismatch
S=5  000010001010001
          0001 match
S=6  000010001010001
           0001 mismatch
S=7  000010001010001
            0001 mismatch
S=8  000010001010001
             0001 mismatch
S=9  000010001010001
              0001 mismatch
S=10 000010001010001
               0001 mismatch
S=11 000010001010001
                0001 match
```

10. Suppose that all characters in the pattern P are different. Show the pseudocode of an algorithm that accelerate the brute-force algorithm to run in time O(n) on an n-character text T.

```
c ← first(P)
while c ≠ Λ do
        if valid(P,c) then output(P, c)
                c ← next(P,c)
end while
```

11. Using Boyer-Moore's algorithm and the text "puppy puppet looks happy", find the substring “puppet”. Compute first the last occurrence function. For the alphabet assume it is the letters that compose the text. Discard blank spaces.
The pattern would occur at shift 6 using Boyer-Moore’s algorithm.

12. Following the Boyer-Moore algorithm, what is the number of comparisons required for finding the previous pattern in the given text? Help yourself with a drawing similar to the examples showed in class and paste it with your answer.
There would be two comparisons. It would do the first comparison at the first spot and then after it would move past the mismatched character and in this case would find the match after doing so.

Extra Credit Questions:

13. Suppose we want to perform sequence pattern matching. For example, given a text “hello” and a pattern “eo”, the matcher should return “true” (because the sequence “eo” exists in “hello”). Can an unmodified Boyer-Moore support sequence pattern matching? Justify your answer.

14. Write an algorithm to perform sequence pattern matching. The algorithm should run in O(n) time and O(n) space, n is the length of the input text. The algorithm takes as input the text T and the pattern P. The output should be true if there is a sequence S in T that matches P.