Problem Set 1

8.8/10

Shrina Parikh

1.

```
{} --- > {F} --- > {F,H} --- > {F,H,K} [doesn't work, missing edge]
      |         |
      |         |
      |         | - > {F,J} --- > {F,J,K} [doesn't work, missing edge]
      |
      | -> {G} --- > {G, H} --- > {G,H,K} [doesn't work, missing edge]
      |             |
      |             |
      |             | -> {G, J} --- > {G,J,K} [doesn't work, missing edge]
      |
      | -> {H} --- > {H, J} [Goal state achieved]
```

Skipping cases such as {F, G}, {F,K}, etc.
Please check the solution

-0.2

2.

```
{} --- > {F} --- > {F,H}
    |
    |
    | -> {G} --- > {G,H}
    |
    |
    | -> {H} --- > {H,J} [Goal state achieved]
```

Skipping cases such as {J}, {K}, etc.
Please check the solution

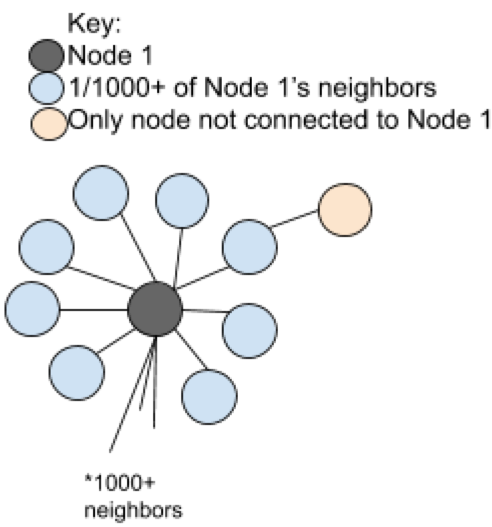
3.

a. One instance of the Best Vertex Cover where doing a BFS will construct 1000 (or more) times as many states as doing a DFS is in a graph in which a node, let's call this node Node 1, has 1000 (or more) neighbors whose only neighbor is Node 1, except for one. One of those 1000 (or more) neighbors has another neighboring node, the only node in the graph

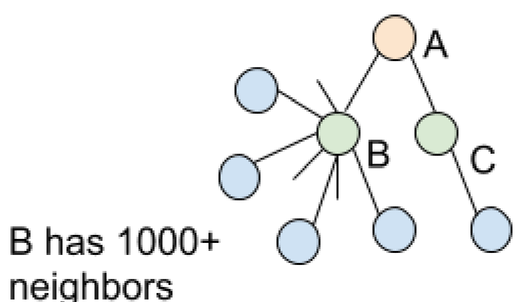
Please specify alphabetical order, otherwise cannot determine.

-0.1

that does not have an edge connected to Node 1.



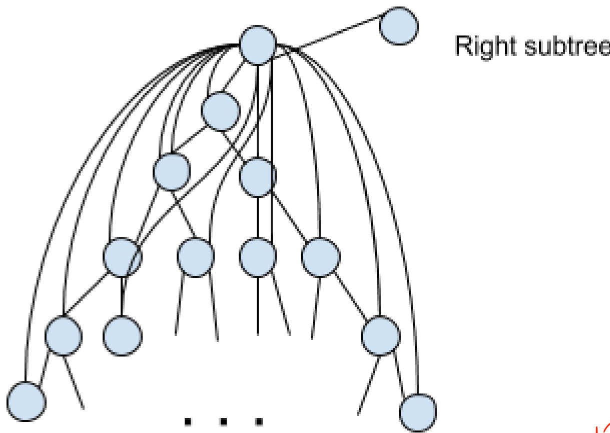
b. One instance of the Best Vertex Cover where doing a DFS will construct 1000 (or more) times as many states as doing a BFS is a tree graph in which the left child of the root node, B, has 1000 (or more) neighbors. The right child of the root node, however, only consists of vertex C and C's sole neighbor. The solution to the Best Vertex Cover problem in this instance is a set S containing the root node A and it's children, so it is S={A,B,C}. However, DFS would create new states as it visits all the vertices in the left subtree looking for the missing edges, causing DFS to construct 1000, or more, times as many states as doing BFS. BFS would reach the goal state by being run from the root node, while DFS would travel through the left subtree.



-0.6

Idea direction right, but pay attention to the rule of BVC,
If traverse in alphabetical order, DFS will complete in 3 steps,
while BFS will first create state for each single node before
going to two nodes then three nodes

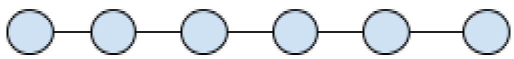
Problem Set 1



4.

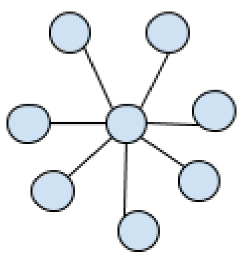
Max depth D = n $n-1$ -0.3

If all n vertices are in a line (connected) and each vertex has a maximum of two neighbors, and two vertices have strictly one neighbor



Branching factor B = n

In the case where n-1 vertices are children or neighbors of the root or n^{th} vertex



Bound on number of states in the state space: 2^n

In a set of n elements, there are 2^n possible subsets. N^n is the upper bound, $2^n < n^n$

5. { D, E } → Error(S) = 9

Potential neighbors: {D}, {E}, {D,E,C}, {D,E,B}, {D,E,A}

For {D}, Error(S) = Max(0, (10-20)) + 3 + 6 + 3 = 12

For {E}, Error(S) = Max(0, (13-20)) + 5 + 3 + 3 = 11

For {D,E,C}, Error(S) = Max(0, (29-20)) + 3 = 12

For {D,E,B}, Error(S) = Max(0, (28-20)) + 3 = 11

For {D,E,A}, Error(S) = Max(0 , (26-20)) = 6 → best option

{D,E,A}

Potential neighbors: {E,A}, {D, A}, {E,D}, {E,A,D,B}, {E,A,D,C}

For {E,A}, Error(S) = Max(0, (16-20)) + 5 = 5 → best option

For {D,A}, Error(S) = Max(0, (13-20)) + 6 = 6

For {E,D}, Error(S) = Max(0,10+13- 20)+3+3 = 9
For {E,A,B,D}, Error(S) = Max(0, (31-20)) = 11
For {E,A,D,C}, Error(S) = Max(0, (27-20)) = 7

{E,A}

Potential neighbors: {E}, {A}, {E,A,D}, {E,A,B}, {E,A,C}
For {E}, Error(S) = Max(0, (13-20)) + 5 + 3 + 3 = 11
For {A}, Error(S) = Max(0, (3-20)) + 10 + 5 + 6 = 21
For {E,A,D}, Error(S) = Max(0, (26-20)) = 6
For {E,A,B}, Error(S) = Max(0, (21-20)) = 1 → best option
For {E,A,C}, Error(S) = Max(0, (22-20)) + 5 = 7

{E,A,B}

Potential neighbors: {E,A}, {A,B}, {E,B}, {E,A,B,D}, {E,A,B,C}
For {E,A}, Error(S) = Max(0, (16-20)) + 5 = 5
For {A, B}, Error(S) = Max(0, (8-20)) + 10 + 6 = 16
For {E,B}, Error(S) = Max(0, (18-20)) + 3 = 3
For {E,A,B,D}, Error(S) = Max(0, (31-20)) = 11
For {E,A,B,C}, Error(S) = Max(0, (27-20)) = 7

At this point, there is no better option. {E,A,B} has a cost of 1, so the hill climbing function terminates at that state. However, this returned solution is not a viable solution as it goes over the budget, T=20. Starting at state {D,E} and using the hill-climbing method, a non-solution is returned.