Name: Ananya Singh
Section: 004
Github URL: https://github.com/ananyasingh7/Graphs
Name of ALL collaborators:
URLs consulted: WilliamFiset on YouTube,

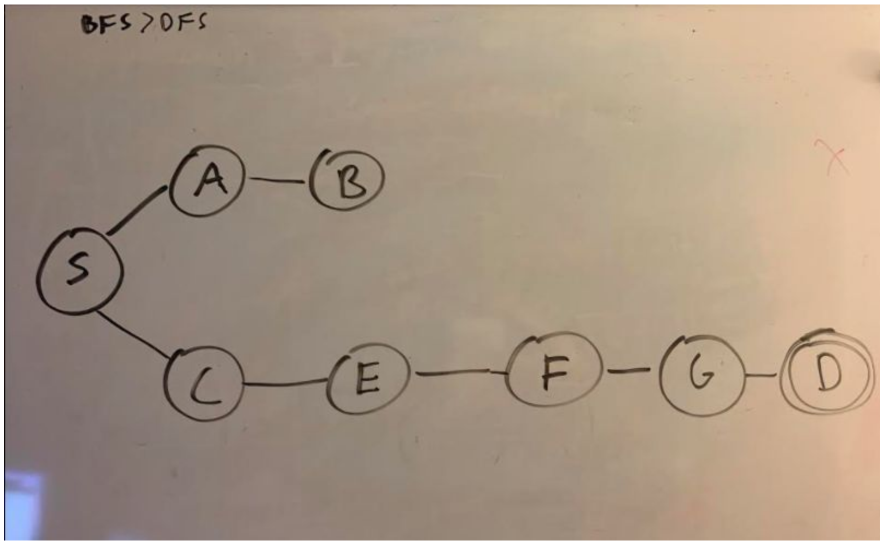
1. Gonna Take My Horse to the Old Town Node

a) [S, A, E, B, C, F, G, K, L, D]

b) We represent edges between the graphs by having an adjacency list for every edge in the graph.

S->neighbors = [A]
A->neighbors = [C, S, B, E]
C->neighbors = [A]
B->neighbors = [A, E]
E->neighbors = [B, A, F]
F->neighbors = [E, G]
G->neighbors = [F, K]
K->neighbors = [G, L]
L->neighbors = [K, D]
D->neighbors = [L]

c)



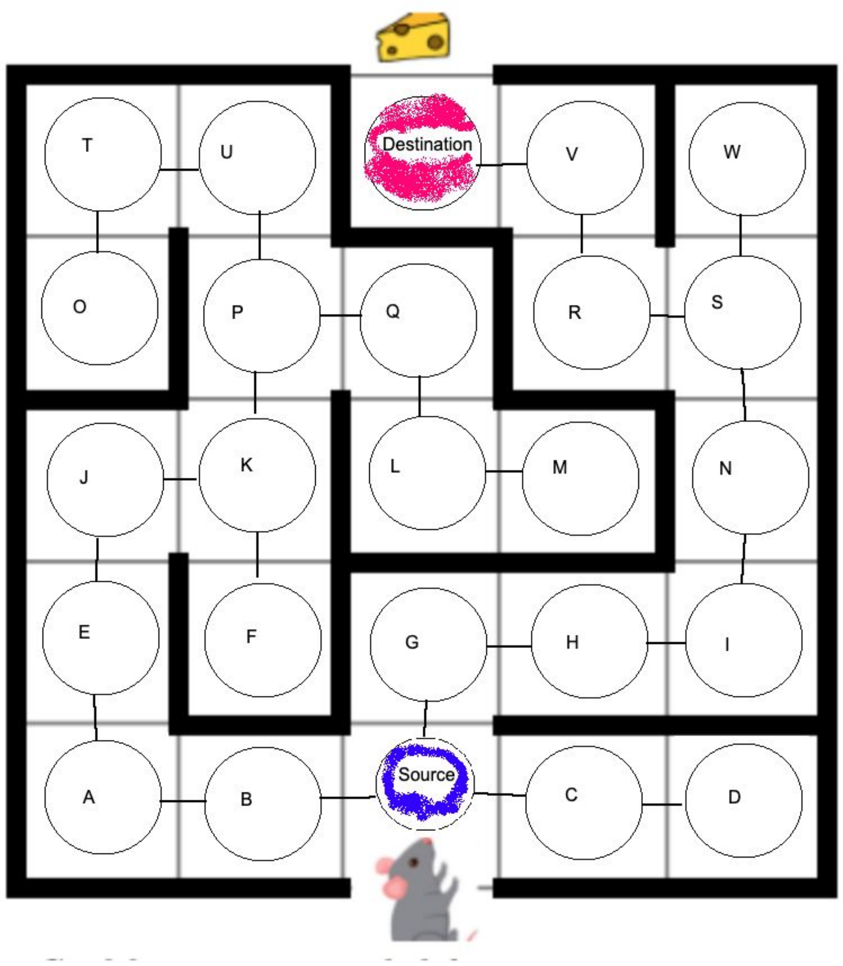
2. Boulevard of Broken Cheese

a) Since there are 25 squares in the maze, that means there are 25 nodes.

b) Edges would be represented as an adjacency list that has nodes that are neighbors and have no wall in between them.

c) 1) NOT directed
   2) NOT cyclic
   3) ARE connected
   4) NOT weighted

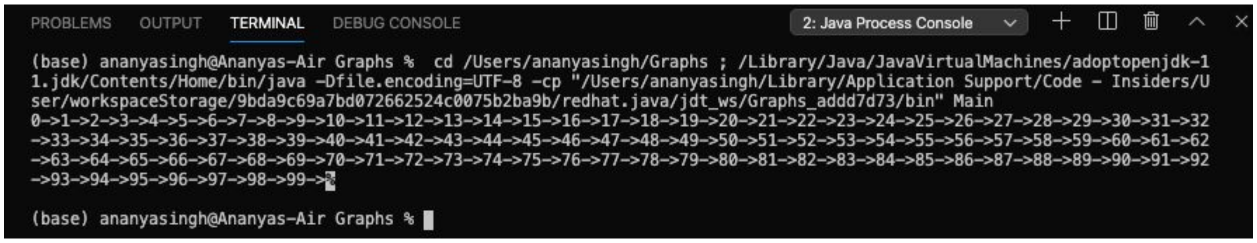
d)



Every letter is a node in the graph!
Source and Destination are labeled and color coded.

3. Traverse This Town

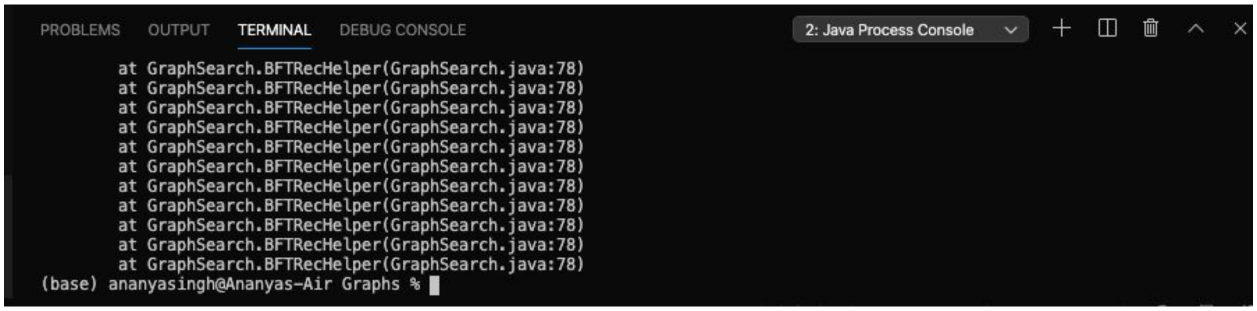
h) This is the output after inserting 100 nodes with BFTRec:



```
(base) ananyasingh@Ananyas-Air Graphs %  cd /Users/ananyasingh/Graphs ; /Library/Java/JavaVirtualMachines/adoptopenjdk-1
1.jdk/Contents/Home/bin/java -Dfile.encoding=UTF-8 -cp "/Users/ananyasingh/Library/Application Support/Code - Insiders/U
ser/workspaceStorage/9bda9c69a7bd072662524c0075b2ba9b/redhat.java/jdt_ws/Graphs_addd7d73/bin" Main
0->1->2->3->4->5->6->7->8->9->10->11->12->13->14->15->16->17->18->19->20->21->22->23->24->25->26->27->28->29->30->31->32
->33->34->35->36->37->38->39->40->41->42->43->44->45->46->47->48->49->50->51->52->53->54->55->56->57->58->59->60->61->62
->63->64->65->66->67->68->69->70->71->72->73->74->75->76->77->78->79->80->81->82->83->84->85->86->87->88->89->90->91->92
->93->94->95->96->97->98->99->%

(base) ananyasingh@Ananyas-Air Graphs %
```

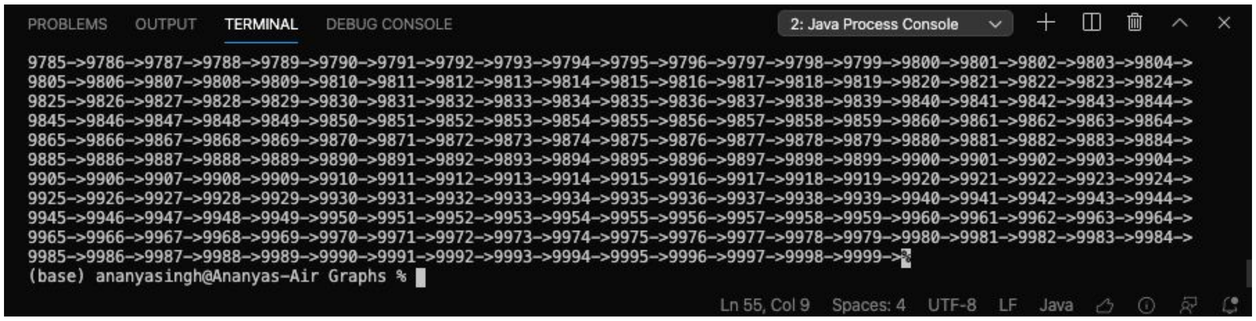
This is the output after inserting 10000 nodes with BFTRec:



```
        at GraphSearch.BFTRecHelper(GraphSearch.java:78)
        at GraphSearch.BFTRecHelper(GraphSearch.java:78)
        at GraphSearch.BFTRecHelper(GraphSearch.java:78)
        at GraphSearch.BFTRecHelper(GraphSearch.java:78)
        at GraphSearch.BFTRecHelper(GraphSearch.java:78)
        at GraphSearch.BFTRecHelper(GraphSearch.java:78)
        at GraphSearch.BFTRecHelper(GraphSearch.java:78)
        at GraphSearch.BFTRecHelper(GraphSearch.java:78)
        at GraphSearch.BFTRecHelper(GraphSearch.java:78)
        at GraphSearch.BFTRecHelper(GraphSearch.java:78)
        at GraphSearch.BFTRecHelper(GraphSearch.java:78)
(base) ananyasingh@Ananyas-Air Graphs %
```

The problem with inserting 10000 nodes recursively is that I ran out of space because of all the implicit space that keeps built up and on top of that, a linkedlist requires space as well.

i) This is the output after inserting 10000 nodes with BFTIter:



```
9785->9786->9787->9788->9789->9790->9791->9792->9793->9794->9795->9796->9797->9798->9799->9800->9801->9802->9803->9804->
9805->9806->9807->9808->9809->9810->9811->9812->9813->9814->9815->9816->9817->9818->9819->9820->9821->9822->9823->9824->
9825->9826->9827->9828->9829->9830->9831->9832->9833->9834->9835->9836->9837->9838->9839->9840->9841->9842->9843->9844->
9845->9846->9847->9848->9849->9850->9851->9852->9853->9854->9855->9856->9857->9858->9859->9860->9861->9862->9863->9864->
9865->9866->9867->9868->9869->9870->9871->9872->9873->9874->9875->9876->9877->9878->9879->9880->9881->9882->9883->9884->
9885->9886->9887->9888->9889->9890->9891->9892->9893->9894->9895->9896->9897->9898->9899->9900->9901->9902->9903->9904->
9905->9906->9907->9908->9909->9910->9911->9912->9913->9914->9915->9916->9917->9918->9919->9920->9921->9922->9923->9924->
9925->9926->9927->9928->9929->9930->9931->9932->9933->9934->9935->9936->9937->9938->9939->9940->9941->9942->9943->9944->
9945->9946->9947->9948->9949->9950->9951->9952->9953->9954->9955->9956->9957->9958->9959->9960->9961->9962->9963->9964->
9965->9966->9967->9968->9969->9970->9971->9972->9973->9974->9975->9976->9977->9978->9979->9980->9981->9982->9983->9984->
9985->9986->9987->9988->9989->9990->9991->9992->9993->9994->9995->9996->9997->9998->9999->%
(base) ananyasingh@Ananyas-Air Graphs %
```

There are no problems with inserting 10000 nodes iteratively because there is no extra implicit space being built up, like there was with recursion, so the stack never runs out of memory.

4. Thank U, Vertext

a) Direct Acyclic Graph (or DAG) has two properties. First, it is a directed graph. Suppose we have node A and node B. If there is a direct edge existing between node A and node B, that doesn't necessarily imply that a directed edge exists between node B and A. Second, it does not have any cycles. The nodes in the new graph, if an edge is formed, is only one one direction compared to as before, where we assumed that it was both directions.

5. Uno, Do, Tre, Cuatro, I Node You Want Me

a) The properties that make it possible to use Dijkstra's on this graph are 1) it has weighted edges and 2) it has connected nodes. There is also a good chance that the graph could have cycles which will be fine for Dijkstra's as well.

6. When You Wish Upon A*

c) An admissible and consistent heuristic that I can use to help solve the maze using A* is the Manhattan Distance. It is admissible because it always underestimates the distance to the destination node, which also makes it consistent.