

Please enter your name and uID below.

Name: Devin White

uID: U0775759

Collaborators, if any, and how you collaborated:

Submission notes

- Due at 11:59 pm (midnight) on Thursday, Oct 27.
- Solutions must be typeset using one of the template files. For each problem, your answer must fit in the space provided (e.g. not spill onto the next page) **without** space-saving tricks like font/margin/line spacing changes.
- Upload a PDF version of your completed problem set to Gradescope.
- Teaching staff reserve the right to request original source/tex files during the grading process, so please retain these until an assignment has been returned.
- Please remember that for this problem set, you are allowed to collaborate in detail with your peers, as long as you cite them. However, you must write up your own solution, alone, from memory. If you do collaborate with other students in this way, you must identify the students and describe the nature of the collaboration. You are not allowed to create a group solution, and all work that you hand in must be written in your own words. Do not base your solution on any other written solution, regardless of the source.

1. (Bipartite Graphs)

Definitions:

- A graph $G = (V, E)$ is bipartite if the vertices V can be partitioned into two subsets L and R , such that every edge has one vertex in L and the other in R .
- A Cycle is V_n vertices (with a minimum of 3 connected in a closed loop/chain e.g. $(V_1, V_2, V_3) V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_1$)

Assumptions:

- Graph G is a bipartite graph if the vertices V can be partitioned into two subsets L and R , such that every edge has one vertex in L and the other in R in other words, for any distance (v, i) either $v \in L, i \in R$ **OR** $v \in R, i \in L$
- The graph is connected (There is a path from any vertex to any other vertex)
- A Connected graph is bipartite iff every cycle has an even number of edges

Proof

- (a) **prove that if G is bipartite, then every cycle in G must have even length.**

Assume for the sake of argument that the graph G has at minimum one cycle $C = (V_1, V_2, V_3, V_4 \dots V_n)$ where $n \% 2 = 1$, or put another way, where cycle n is of odd length. Letting V_1 be in the subset L , then by the definition of a bipartite graph given above, it **MUST** be the case that V_2 is in the subset R , V_3 Must be in L , V_4 in R , continuing this inductively we can quickly see can see that for any given vertex V_i , if $i \% 2 = 1$ (Odd) then V_i is in L and if $i \% 2 = 0$ (Even) V_i is in R . As n is odd as stated above, then it is the case that V_n is vertex L . This is a contradiction, as V_1 and V_n are connected and both in subset L , this conflicts with the definition given above of a bipartite graph that for every edge one vertex must be in L , and the other must be in R . This means that if G is a bipartite graph, then every cycle in G must be an even length

- (b) **prove that if every cycle in G has even length, then G must be bipartite.**

Assume then, as shown per part 1 that every Cycle in G **must** be an even length. Picking an arbitrary Vertex defined as V , we will define L as all vertices an odd distance from V , and R as all Vertices an even distance from V V itself is R .

Assuming $\text{dist}(i, v)$ represents the shortest path between any 2 vertices in graph G , and we put our arbitrary vertex V in R , V 's direct neighbors in L , and their direct neighbors in R , and so on, we know intuitively since G is a connected graph, as well as by definition 2 above, the subsets R and L make up the entire graph (e.g. $R \cup L = G$) and therefore $R \cap L$ is empty because a vertex can't be in both L and R . Assuming, for the sake of contradiction that there does exist an edge $(j, v) \in E$ where $j, v \in R$, then by constructing the graph we can see that it is the case that $\text{dist}(i, v)$ and $\text{dist}(i, j)$ must both be even (due our arbitrary v being in R) Assuming that $\text{SP}(i, j)$ and $\text{SP}(v, i)$ are the shortest paths which connect i to j and i to v , then the length of the cycle of $\text{SP}(i, j) + \text{SP}(v, i)$ has a total length of $1 + \text{dist}(i, j) + \text{dist}(i, v)$ which is a contradiction. Therefore no edge like $E(j, v)$ may exist meaning that if every cycle in G is even length, then G must be a bipartite graph

2. (Getting Gold)

- (a) **What are the vertices? What does each vertex represent? For example, in Flood Fill, the vertices represent pixels in the input.**

The vertices in the graph represent the "tiles" in the map. In other words, a vertex is reachable from another vertex if and only if a tile is directly adjacent (directly up, down, left, or right) to another tile.

- (b) **What are the edges and what do they represent (defining your edges carefully simplifies the problem)? In Flood Fill, the edges are undirected and connect neighboring pixels of the same color.**

In "getting gold" the edges are undirected and connect each tile to each adjacent tile ONLY in the left, right, up and down directions (no diagonals). A given vertex can have **AT MOST** 4 edges. (Possible to map directions as +1 X +1 Y -1 X -1 Y). Walls ('#') and traps ('T') represent the "end of the path" for any given path on the graph, or put another way, a tile directly adjacent to a Trap ('T') is the last Vertex visited on a given path. No edges connect a valid tile to a trap.

- (c) **If the vertices and/or edges have associated values, what are they? In Flood Fill, vertices store the color of the associated pixel.**

The vertices consist of the following information: The X and Y coordinates where the tile is "set", which can be represented within a 2D array, the thing contained (or the "value") within the tile (#:wall, .:open, G:gold, T:trap). For example, the top left tile would have coordinates X:0 Y:0 or (0,0) while the bottom right tile would have the coordinates X:n Y:n or (n,n). X and Y can also be thought of as indexes 0 ≤ i < H and 0 ≤ j < W.

- (d) **What problem do you need to solve on this graph? How does WFS solve this problem? Solving Flood Fill is equivalent to turning i, j's connected component to the provided color. WFS allows us to reach and mark all elements in i, j's connected component.**

The problem to be solved with this graph is to find the maximum amount of possible gold that can be reached without running into a trap or passing through impassable walls, starting from the initial player position 'P', as the player doesn't know what is in a given tile until they are inside a said tile, and the player doesn't know what direction/how many traps there are when sensing a "draft", then **any space adjacent to a trap is considered a "dead end" and thus is the last vertex on a given path.** (nothing is reachable from that vertex aside from the previous vertex (or vertices))

WFS allows us to reach/traverse into every tile/vertex that is connected to the initial player position 'P', increasing the amount of gold every time the player moves into a tile/vertex with 'G' while accounting for and avoiding traps and walls by considering them "disconnected" from the graph. In other words, a wall and a Trap are not considered "reachable" tiles and so can not be traversed, and as such tiles on the other side of a wall/trap can not be reached from a tile on the opposite side of a wall/trap as the exact location of traps are unknown, vertices directly adjacent to (up, down, left, or right from) a trap are the last vertices on a path and so the path ends there, on the other hand, a wall is only the end at a given direction, and simply would reduce the possible edges at a vertex by 1.

- (e) **What is the running time of your entire algorithm, including the construction of the graph, in terms of the input parameters W and H? The running time of this Algorithm is $O(2(W*H))$ which reduces to $O(W*H)$.**