Problem Set 5

All parts are due Tuesday, November 24, 2015 at 11:59PM.

Name: David Walter

Collaborators: None.

Part A

Problem 5-1.

Every vertex has at most 4 outward edges with weights $p(x+1,y), p(x-1,y), p(x,y+1)$, and $p(x,y-1)$ (if they exist). Each vertex also has at most 4 inward edges each with weight $p(x,y)$, so each vertex has at most 8 edges. $|V| = O(n^2)$ and $|E| = O(n^2)$, so $|V| + |E| = O(n^2)$.

We will create a new graph G as a copy of the original graph representation of the video game. While there are still edges with an edge weight $p(x,y)$ greater than 1 in G, we will check if the edge weight is at least Bens health M, and if so, we will simple delete this edge because there is no shortest path through this edge. Otherwise, we will split up each edge into $edge - weight$ number of weight 1 edges connecting new vertices, with these new edges having the same direction as the original edge. Since $|E| = O(n^2)$ and we will do this splitting of edges at most M times per edge, we know that making G an unweighted graph will be $O(Mn^2)$. Now, we can run BFS on this new unweighted, directed graph G. In G, $|V| = O(Mn^2)$ and $|E| = O(Mn^2)$. The runtime for BFS is $O(|V| + |E|)$, so this algorithm is $O(Mn^2)$. If while we are running BFS, the weight of the shortest path is at least M, then we return $None$, Otherwise we will return the shortest path result from running BFS.

Problem 5-2.

Keeping in mind how many edges and vertices we have, as discussed in 5-1, we will run the Djikstra algorithm to help Ben escape. We will run Djikstra on the start, and on each heath pack, so $K+1 = O(K)$ times. So running Djikstra from the start node, every vertex will be augmented with the shortest path distance to the start node, and likewise will be done, from the shortest path from each health pack. As we do this we will be constructing a graph G' containing only the start node, the health pack nodes, and the end point. The edges in G' will represent there being an at most M-1 length path between vertices, which means Ben can survive the journey along that edge. If any path exists in G' from the start node to the end node (the start node and end node are connected in the same graph) then we know that ben can saftely make it to the end. If the start node and the end node in G' are not connected, then Ben cannot make it and we return that Ben is

trapped for good. Djikstra runs in $O(|V|log|V| + |E|) = O(n^2log(n^2) + n^2) = O(n^2log(n))$ run time. And we run djikstra O(K) times, and making G' and adding edges happens in constant time while we run djikstra, so our algorithm runs in $O(Kn^2log(n))$ time.

Part B

Problem 5-3.

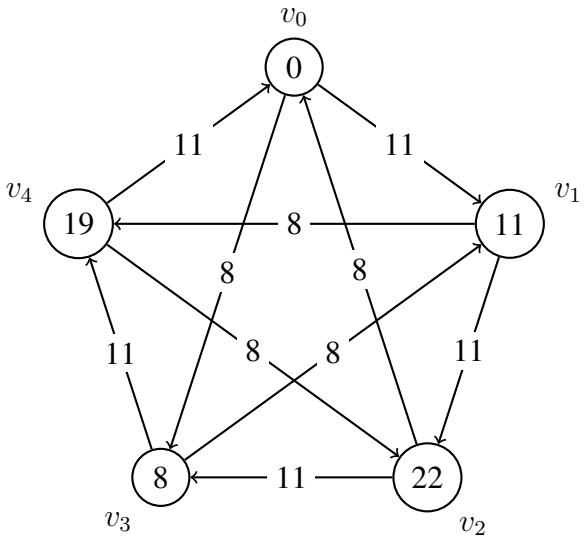
(a) Proof of Claim 1.

The definition of congruence: $a \equiv b \mod n$ if and only if, n is a divisor of $(a - b)$ if and only if the remainder of $a/n =$ remainder of b/n. So if $l' \equiv l \mod a$ and $l' > l$, if this congruence holds, then we know that $l' = l + c * a$ where c is any non-negative integer. Since l is a whimsical number and l' is l plus some positive number multiple of some number a, which is a member of A, then l' must also be a whimsical number.

Proof of Claim 2.

The $\mod n$ congruence relation is transitive and symmetric. If $a \equiv b \mod n$ and $b \equiv c \mod n$ then $a \equiv c \mod n$. So if $r \equiv x_{r'} \mod a$ and $r \equiv l \mod a$, then $x_{r'} \equiv l \mod a$, and if $l >= x_{r'}$, then $l = c * x_{r'}$ where c is any non-negative integer. Since we know that $x_{r'}$ is a whimsical number, then we know, just like in claim 1, that l must be a whimsical number if the congruence holds and $x_{r'}$ exists for some r'.

(b) Graph Implementation:



(c) The run time of the constructor is $O(MlogM + kM)$. In the constructor, to create the graph, like the one in part b, we create a blank adjacency matrix in $O(M)$ time and we finish creating the adjacency matrix in $O(kM)$ time. Next, we run djikstra on the

graph. The runtime for djikstra is $O(|V|log|V| + |E|)$, and $|V| = a - 1 = O(M)$, and $|E| = |V|(k-1) = (a-1)(k-1) = O(kM)$, so the runtime for djikstra is $O(MlogM + kM)$, which is less than the $O(kMlog(kM))$ run time that was asked for.

The runtime of the $is\_whimsical$ is $O(1)$. The function checks $r' = l(mod(a))$ in constant time, it checks if $x_{r'}$ exists by checking in the dictionary of weights in constant time, and it checks if $l > x_{r'}$ in constant time.

(d) *Python script submitted.*