Ethan Nguyen
ethantn04@csu.fullerton.edu
CPSC 335 Algorithm Engineering
Project 3 - Algorithm 1

Project Report - Algorithm 1

Pseudocode

Function minDaysBurnHealthyTrees(forest):

- define variable rows and cols for the number of rows and columns in the forest
- this will simulate a 2d grid of the entire forest

```
function mindaysburnhealthytree(forest):
        rows = rows of forest
        column = columns of forest
```

Initialize the queue and set the number count of healthy trees to 0.

Create a for loop for each cell of i and j which represent the rows and columns in the forest.
This loop will count the number of healthy trees and enqueue the burning ones.
The program will return 0 if there are no trees to burn.
The program will return -1 if there are no burning trees to continue the fire.

```
for each cell(i, j) in forest:
        if forest[i][j] == 2:
                enqueue(i, j) into q
        else if forest[i][j] == 1:
                healthytree += 1
```

Breadth-first search is used to simulate the burning of the trees.
A while loop should be used to continue as long as there are burning trees.
It is important to create a flag to check if any new healthy trees did catch fire so that the number of days can be incremented.
The coordinates of the burning tree should also be recorded so that if it is neighboring a burning tree, the healthy tree will also catch on fire.
The number of healthy trees should then be decremented.
The program will then return the number of days it took to burn all of the healthy trees as well as whether some of the healthy trees were unreachable.

```
if healthyTrees == 0:
        return days
else:
        return -1
```

Sample

```
//vectors simulate 2d grid of the forest
//0 = empty, 1 = healthy, 2 = burning
vector<vector<int>> forest1 =
   {
      {2,1,1},//standard example
      {1,1,0},
      {0,1,1}

   };

   vector<vector<int>> forest2 =
   {
      {2,1,1},//not all trees can burn
      {0,1,1},
      {1,0,0}

   };

   vector<vector<int>> forest3 =
   {
      {0,2}//no healthy trees

   };
```

Results :
Test Case 1 Output: 4
Test Case 2 Output: -1
Test Case 3 Output: 0

Mathematical Analysis & Efficiency Class

Let m be the num of rows.
Let n be the num of columns.
The total number of cells can be depicted as V = m * n.
To traverse the cells to count the healthy trees and burning trees, it will take O(m * n) time.
The breadth-first search traversal through the cells is O(n) time.

So because there are no further indications of an increase of complexity, the total time complexity is O(m x n)