Homework 4

CSE 250 - Fall 2015

Introduction

In this homework, we will playing a board game called JetPorter in which your player must reach the end of a linear board in the fewest number of turns. Each turn, the maximum number of spaces that can be travel can be determined based on the current square and the statistics of the player. Specifically, each square has both a teleporter and a cannon that can be used to advance to a later square. Each square has a given amount of gun powder for the cannon, and energy for the teleporter. In addition, each player has a specific weight affecting the maximum travel distance for the cannons, and chi affecting the maximum range of the teleporters. Computing these maximum distances is determined by the function in part 1.

Note that a player can choose to travel fewer than the maximum number of squares each turn. For part 3, you will need to efficiently determine the best path through the board to minimize the number of turns needed to reach the end of the board.

Relevant code can be found in the course repository:
https://bitbucket.org/hartloff/cse250-fall2015/src/c68d56e1bfd7/homework/?at=master

Part 1

Due: Wednesday, October 14 @ 11:59pm

Debug the function

```
int GameUtil::compute(Square*, Player*)
```

which, given a square and a player, will return the maximum number of squares from that square that the player can travel. The maximum is calculated based on whichever is higher: the distance the player can travel using the teleporter at that square, the distance the player can travel using the cannon at that square, or simply the value 1 if both of them are less than 1. The distance a player can travel using the teleporter at a square is given by:

$$d_t = \frac{1}{1+c} * \sum_{i=0}^{floor(c)} \sqrt{i * c * e}$$

Where `c` is the player's chi, `floor(c)` is the chi rounded down to the nearest integer, and `e` is the teleporter energy at that square.

The distance a player can travel using the cannon at a square is given by:

$$d_c = \frac{(\frac{p^{1.7}}{w^{1.5}})^2}{9.8}$$

Where `p` is the `cannonPowder` at the square and `w` is the player's weight.

Part 2

Due: Thursday, October 15 @ 11:59pm

Implement the function

```
bool GameUtil::isValidPath(std::vector<int>, Player*, Game*)
```

that for the given player and game, will return `true` or `false` depending on whether or not the path is a valid path. A valid path is a vector of integers such that each element corresponds to the index of each subsequently visited square in the game's board. Furthermore, a valid path must start at the first square of the board, end at the last square of the board, move forward on the board with every step, and each step cannot exceed the maximum distance that the player can travel from its current square calculated in `int GameUtil::compute(Square*, Player*)`.

Part 3

Due: Friday, October 16 @ 11:59pm

Implement a function

```
int GameUtil::shortestPathDistance(Game*, Player*)
```

that, given a game and player, will return the distance of the shortest possible valid path the player can take to reach the end of the game's board. The distance of a path is the number of jumps the player must make to reach the last square. This function will be graded on correctness as well as efficiency. To get full credit, your function must have a polynomial runtime in terms of number of squares on the board (so, for example, $\Theta(n^2)$ would be an acceptable runtime, while $\Theta(2^n)$ would be too slow).

Submission

Submit all your code in the file named `gameUtil.cpp`. Submissions must be made on the CSE servers using the `submit_cse250` command.