Members:
Yang Xu (yxu4@wpi.edu)
Ying Lu (ylu6@wpi.edu)
Huyen Nguyen (hbnguyen@wpi.edu)

CS 4341 - 2016
PROJECT 1

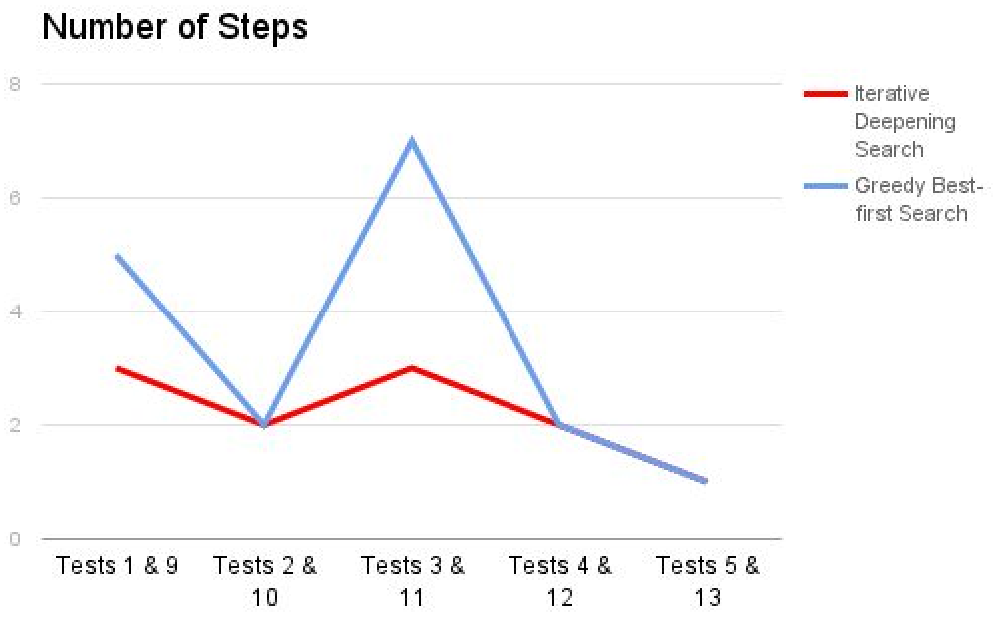
You should develop a suite of test problems and run both iterative deepening and greedy (best first) search on them. You should be sure to have at least one test problem where greedy search returns a suboptimal solution. Similarly, develop at least one test problem where iterative deepening is not able to search deep enough to get a solution but greedy search can.

There are 16 test cases, where tests 1-8 are for iterative deepening search, and tests 9-16 are for greedy best-first search. Tests 1 and 9 are basically identical, except for the name of the algorithm being used. Similarly, tests 2 and 10 are identical, etc.

Tests 1 and 9 are basically the sample test case Professor Beck gave in the project specification.
Tests 2 and 10 check to see if the implementation are optimal. In fact, 81 = 3 * 9 * 3. This is the optimal solution, but we can also do 81 = 3 * 3 * 3 * 3.
Tests 3 and 11 check to see if the program can apply subtraction to get to the goal.
Tests 4 and 12 check the power operator.
Tests 5 and 13 is a simple test case. It is used to confirm that the output statistics (number of nodes expanded, maximum depth, and number of steps) are correct.
Tests 6 and 14 is a tricky test case, where iterative deepening search is supposed to fail because of the time limit, but greedy best-first search can find a solution.
Tests 7 and 15 is another tricky test case, where iterative deepening search can find a solution, but greedy best-first search fails because of time limit.
Tests 8 and 16 check the behavior of both algorithms when it is impossible to reach the goal.

In general, how did the number of steps required by greedy search compare to iterative deepening? How does the number of nodes expanded vary for the two techniques? A graph may be helpful to compare them.

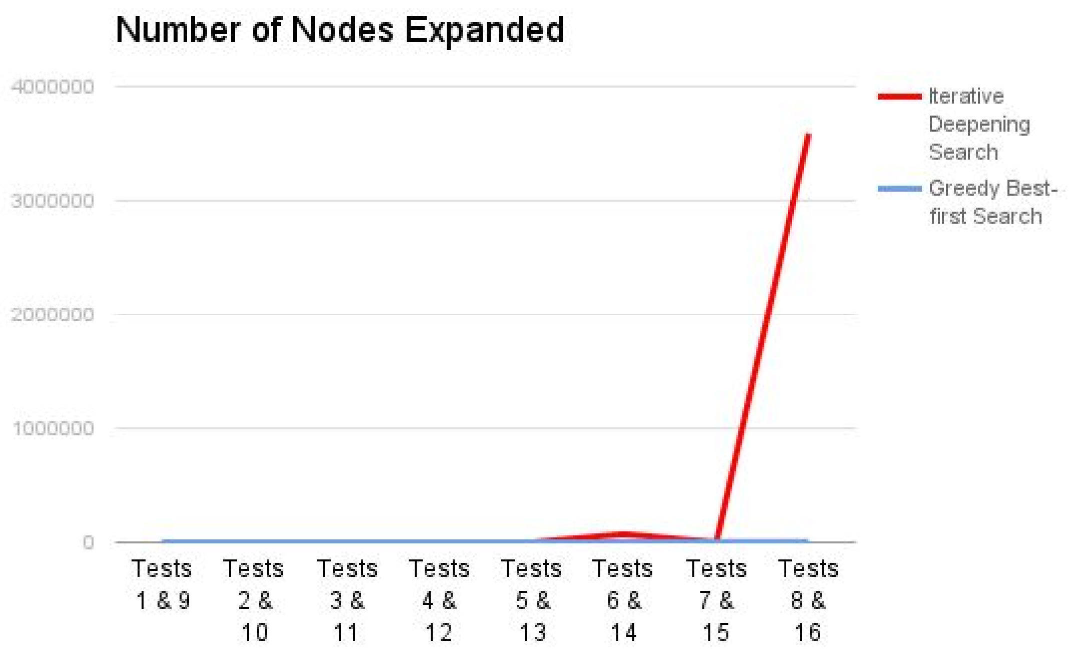
<u>Compare the number of steps:</u>



Note: We left out the remaining test cases because the number of steps for those test cases are ill-defined. Specifically, one (or both) of the algorithms cannot reach the goal.

In general, iterative deepening search takes less number of steps. This makes sense because iterative deepening search explores all the paths, and it can choose the best path if there is one and if there is enough time. The greedy best-first search will go with the most promising path, but there's no guarantee this is the shortest path. In fact, greedy best-first search has to choose a different path when it realizes it's moving further from the goal.

<u>Compare the number of nodes expanded:</u>



Iterative deepening search expands more nodes. This makes sense because iterative deepening search explores all the paths, whereas greedy best-first search doesn't take the paths it don't find promising, and it doesn't expand the nodes along those paths either.

Compute the effective branching factor of iterative deepening. Compare it to the effective branching factor of greedy search.

$$effective\ branching\ factor \approx \sqrt[depth]{number\ of\ nodes\ expanded}$$

	Iterative Deepening Search	Greedy Best-first Search
Tests 1 & 9	$\sqrt[3]{28} \approx 3$	$\sqrt[5]{30} \approx 2$
Tests 2 & 10	$\sqrt{3} \approx 1.41$	$\sqrt{9} = 3$
Tests 3 & 11	$\sqrt[3]{8} = 2$	$\sqrt[7]{24} \approx 1.57$
Tests 4 & 12	$\sqrt{3} \approx 1.41$	$\sqrt{15} \approx 3.87$
Tests 5 & 13	1	8
Tests 6 & 14	$\sqrt[4]{72410} \approx 16.4$	$\sqrt[12]{2911} \approx 1.9$
Tests 7 & 15	$\sqrt{3} \approx 1.41$	$\sqrt[1412]{4239} \approx 1$
Tests 8 & 16	$\sqrt[14]{3587219} \approx 2.9$	$\sqrt[432]{3891} \approx 1$

In general, iterative deepening search has a bigger effective branching factor. This is because is has to expand all the nodes. In contrast, greedy best-first search only expands the nodes along the most promising paths.