Jeremy Timperio
Unity Id: jmtimper

A* Search Algorithm

Different path from A to B and B to A

Explaination:

Could not find a path that would be different after using the A* search algorithm. Since the algorithm takes in both distance from start and distance from goal each path is the identical both ways.

Expands different number of nodes

Explaination:

The path from Fort Worth to Dallas has a different explored queue both ways since both are very close to each other but don't have a road connecting each other. Because of this, the explored queue from Fort Worth to Dallas is much smaller since there is only one path to Fort Worth where as Dallas has two possible paths.

Output:

Fort Worth to Dallas

```
Explored Nodes:

['ftWorth', 'oklahomaCity', 'tulsa', 'littleRock', 'kansasCity', 'wichita', 'memphis', 'stLouis',
'lincoln', 'nashville', 'omaha', 'desMoines', 'chattanooga', 'indianapolis', 'denver',
'coloradoSprings', 'atlanta', 'dallas']

explored length: 18


Solution Path:

['ftWorth', 'oklahomaCity', 'tulsa', 'kansasCity', 'wichita', 'denver', 'dallas']

Solution Length: 7

Solution Distance: 2068
```

Dallas to Fort Worth

```
Explored Nodes:

['dallas', 'mexia', 'houston', 'beaumont', 'austin', 'sanAntonio', 'lafayette', 'batonRouge', 'laredo',
'newOrleans', 'denver', 'coloradoSprings', 'pensacola', 'wichita', 'elPaso', 'santaFe',
'albuquerque', 'tallahassee', 'grandJunction', 'kansasCity', 'albanyGA', 'tulsa', 'albuquerque',
'oklahomaCity', 'ftWorth']

explored length: 25
```

Solution Path:

['dallas', 'denver', 'wichita', 'kansasCity', 'tulsa', 'oklahomaCity', 'ftWorth']

Solution Length: 7

Solution Distance: 2068

A* Search Algorithm vs. Greedy Search Algorithm

Longer Greedy path than A*

Explaination:

The path from Toroto to Albany, NY displays how A* Algorithm finds more efficient paths than greedy. The solution path for greedy has a longer distance because every node it chooses to explore is based on the distance from the goal and it doesn't take in to account to distance from the start. Since Ottawa and Montreal are closer to albany than Buffalo, the Greedy Algorithm explores those first without even checking out Buffalo.

Path from Toronto to Albany, NY

A Search Algorithm Output*

Explored Nodes:

['toronto', 'buffalo', 'rochester', 'albanyNY']

explored length: 4

Solution Path:

['toronto', 'buffalo', 'rochester', 'albanyNY']

Solution Length: 4

Solution Distance: 317

Greedy Search Algorithm Output

Explored Nodes:

['toronto', 'ottawa', 'montreal', 'albanyNY']

explored length: 4

Solution Path:

['toronto', 'ottawa', 'montreal', 'albanyNY']

Solution Length: 4

Jeremy Timperio
Unity Id: jmtimper

Solution Distance: 627

Expands more nodes

Explaination:

The path from Provo to Toledo is a great example to show the efficiency of the A* Algorithm versus the Greedy Algorithm. Both Provo and Toledo only have one path connecting them to the grid making it difficult for the algorithms to find the correct path connecting them. In the output displayed below, both algorithms find the same path but the Greedy Algorithm must search 57 city nodes while the A* algorithm only has to search 32 city nodes. This comes back to how A* uses both the distance from the start and end to find the best path that way even if it runs into a dead end it's expansion towards the goal has a better aim.

Path from Provo to Toledo

A Search Algorithm Output*

Explored Nodes:

['provo', 'grandJunction', 'denver', 'wichita', 'kansasCity', 'stLouis', 'indianapolis', 'cincinnati', 'dayton', 'coloradoSprings', 'columbus', 'lincoln', 'omaha', 'desMoines', 'cleveland', 'dallas', 'tulsa', 'minneapolis', 'greenBay', 'milwaukee', 'santaFe', 'mexia', 'littleRock', 'memphis', 'pittsburgh', 'nashville', 'chicago', 'albuquerque', 'oklahomaCity', 'buffalo', 'midland', 'toledo']

explored length: 32

Solution Path:

['provo', 'grandJunction', 'denver', 'wichita', 'lincoln', 'omaha', 'desMoines', 'minneapolis', 'greenBay', 'milwaukee', 'chicago', 'midland', 'toledo']

Solution Length: 13

Solution Distance: 2577

Greedy Search Algorithm Output

Explored Nodes:

['provo', 'grandJunction', 'denver', 'wichita', 'kansasCity', 'stLouis', 'indianapolis', 'cincinnati', 'dayton', 'columbus', 'cleveland', 'pittsburgh', 'buffalo', 'toronto', 'saultSteMarie', 'rochester', 'philadelphia', 'baltimore', 'washington', 'richmond', 'ottawa', 'norfolk', 'raleigh', 'greensboro', 'charlotte', 'augusta', 'thunderBay', 'newYork', 'albanyNY', 'savannah', 'montreal', 'montreal', 'jacksonville', 'lakeCity', 'tallahassee', 'albanyGA', 'macon', 'atlanta', 'chattanooga', 'nashville', 'memphis', 'littleRock', 'pensacola', 'boston', 'providence', 'newHaven', 'stamford', 'daytonaBeach', 'lincoln', 'omaha', 'desMoines', 'minneapolis', 'greenBay', 'milwaukee', 'chicago', 'midland', 'toledo']

explored length: 57

Solution Path:

['provo', 'grandJunction', 'denver', 'wichita', 'lincoln', 'omaha', 'desMoines', 'minneapolis', 'greenBay', 'milwaukee', 'chicago', 'midland', 'toledo']

Solution Length: 13

Solution Distance: 2577

A* Search Algorithm vs. Dynamic Programming Search Algorithm

Longer Dynamic Programming path than A*

Explaination:

The path from Sacramento to Yuma demonstrates the efficiency of the Dynamic Programming Search Algorithm versus A* Search Algorithm. The DP Algorithm's path is slightly longer than A* Algorithm's path because it only takes distance from start point into consideration. Without any estimation of the endpoint, the DP algorithm must explore more nodes to find the goal. That's why the DP's explored queue is much larger than the explored queue of the A* algorithm.

Path from Sacramento to Yuma

A Search Algorithm Output*

Explored Nodes:

['sacramento', 'stockton', 'modesto', 'fresno', 'bakersfield', 'losAngeles', 'sanDiego', 'yuma']

explored length: 8

Solution Path:

['sacramento', 'stockton', 'modesto', 'fresno', 'bakersfield', 'losAngeles', 'sanDiego', 'yuma']

Solution Length: 8

Solution Distance: 704

Dynamic Programming Search Algorithm Output

Explored Nodes:

['sacramento', 'stockton', 'modesto', 'sanFrancisco', 'oakland', 'pointReyes', 'reno', 'sanJose', 'salinas', 'fresno', 'bakersfield', 'sanLuisObispo', 'redding', 'losAngeles', 'losAngeles', 'medford', 'sanDiego', 'sanDiego', 'eugene', 'saltLakeCity', 'salem', 'lasVegas', 'lasVegas', 'portland', 'yuma']

explored length: 25

Solution Path:

['sacramento', 'stockton', 'modesto', 'fresno', 'bakersfield', 'losAngeles', 'sanDiego', 'yuma']

Solution Length: 8

Solution Distance: 774

Expands more nodes

Explaination:

Tha path from Saramento to Portland produces the same solution path in Dynamic Programming Search Algorithm and in A* Search Algorithm. However, the explored paths are very different because DP lasks information about the goal node. The DP algorithm explored path must explore 24 node while the A* Algorithm only must explore 13 to come to the same conclusion.

Path from Sacramento to Portland

A Search Algorithm Output*

Explored Nodes:

['sacramento', 'stockton', 'reno', 'pointReyes', 'sanFrancisco', 'modesto', 'oakland', 'redding', 'medford', 'sanJose', 'eugene', 'salem', 'portland']

explored length: 13

Solution Path:

['sacramento', 'pointReyes', 'redding', 'medford', 'eugene', 'salem', 'portland']

Solution Length: 7

Solution Distance: 755

Dynamic Programming Search Algorithm Output

Explored Nodes:

['sacramento', 'stockton', 'modesto', 'sanFrancisco', 'oakland', 'pointReyes', 'reno', 'sanJose', 'salinas', 'fresno', 'bakersfield', 'sanLuisObispo', 'redding', 'losAngeles', 'losAngeles', 'medford', 'sanDiego', 'sanDiego', 'eugene', 'saltLakeCity', 'salem', 'lasVegas', 'lasVegas', 'portland']

explored length: 24

Jeremy Timperio
Unity Id: jmtimper

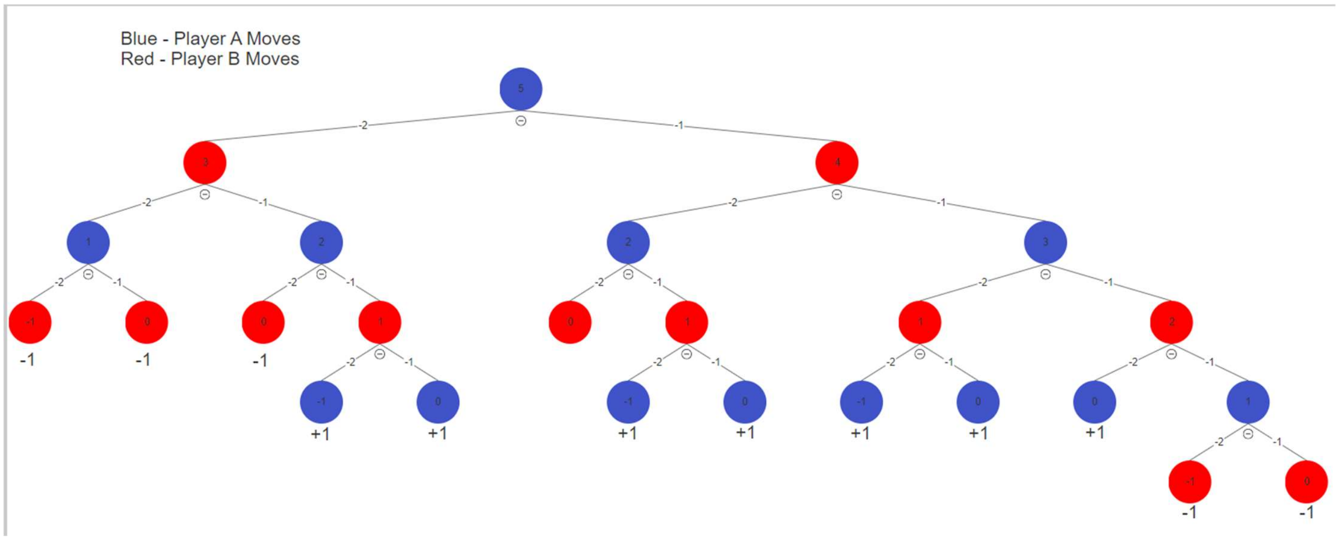
Solution Path:

['sacramento', 'pointReyes', 'redding', 'medford', 'eugene', 'salem', 'portland']

Solution Length: 7

Solution Distance: 755

Question 2: Game Tree



If both players play optimally, player A would win every time because Player A would first take one coin. Next, player B would take one Coin. Then, player A would take two coins. This would force player B to take at least one coin which would be the last remaining coin. If player A takes one coin first, it forces player B to take either one coin or two coins. If player B takes two coins, then player A would only take one coin forcing player B to draw the last coin. If player A follows these rules, there is no way player A loses the game.