Application of dynamic programming and graph algorithms

Introduction

Shortest path algorithms are very important for computer networks, like the Internet.Shortest path algorithms have wide applications and so wide are no of algorithms to find shortest path.Mapping software like Google or Apple maps makes use of the shortest path algorithms. They are also important for road network, operations, and logistics research.But the most important factor among all such algorithms are time and space complexity of the algorithms.

Algorithms to find Shortest Path

There are mainly two algortihms for finding shortest paths :

1. Dijkstra’s Algorithm
2. Astar Algorithm

Dijkstra’s Algorithm

Dijkstra’s Algorithm is a greedy algorithm for solving single source shortest path problems that provides us with the shortest path from one particular given node to all other nodes in the graph.This algorithm finds the path with lowest cost(i.e. the shortestpath) between that vertex and every other vertex .For example , if the vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road , Dijkstra’s algorithm can be used to find the shortest route between one city and all other cities.

Pseudocode of Dijkstra’s Algorithm

1) Create a Min Heap of size V where V is the number of vertices in the given graph. Every node of min heap contains vertex number and distance value of the vertex.

2) Initialize Min Heap with source vertex as root (the distance value assigned to source vertex is 0). The distance value assigned to all other vertices is INF (infinite).

3) While Min Heap is not empty, do following.

…..a) Extract the vertex with minimum distance value node from Min Heap. Let the extracted vertex be u.

…..b) For every adjacent vertex v of u, check if v is in Min Heap. If v is in Min Heap and distance value is more than weight of u-v plus distance value of u, then update the distance value of v.

Astar Shortest Path Algorithm

A* is a graph traversal and path search algorithm, which is considered to have "brains" , it is known as intelligent algorithm as it chooses very optimal choice at every step.

The secret to its success is that it combines the pieces of information that Dijkstra’s Algorithm uses (favoring vertices that are close to the starting point) and information that Greedy Best-First-Search uses (favoring vertices that are close to the goal). In the standard terminology used when talking about A*, g(n) represents the exact cost of the path from the starting point to any vertex n, and h(n) represents the heuristic estimated cost from vertex n to the goal. Each time through the main loop, it examines the vertex n that has the lowest f(n) = g(n) + h(n).

What A* Search Algorithm does is that at each step it picks the node according to a value-‘f’ which is a parameter equal to the sum of two other parameters – ‘g’ and ‘h’. At each step it picks the node having the lowest ‘f’, and process that node.

Heuristic for Astar Algorithm

A* always requires a heuristic, it is defined using heuristic values for distances. A* in principle is just the ordinary Dijkstra algorithm using heuristic guesses for the distances.The heuristic function should run fast, in O(1) at query time. Otherwise we won't have much benefit from it. As heuristic we can select every function h for which:

Straight-line heuristic:

The straight-line distance (or as-the-crow-flies) is straightforward and easy to compute. For two nodes v, u we know the exact location, i.e. Longitude and Latitude.

We then just need to compute the straight-line distance by defining h as the Euclidean distance.

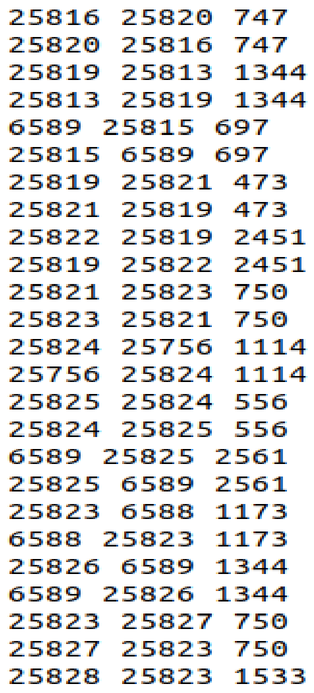
This methods run in O(1).

Dataset

Large dataset of road networks has been taken for efficient testing of the project .The dataset contains almost 5 lac+ nodes and 20 lac+ edges.This dataset represents the road network of california state.The format of dataset is represented by following image.

The first column of the dataset represents the source node , second column represents the destination node and third column represents the weight of the given edge.

Screenshot of Dataset



```
25816 25820 747
25820 25816 747
25819 25813 1344
25813 25819 1344
6589 25815 697
25815 6589 697
25819 25821 473
25821 25819 473
25822 25819 2451
25819 25822 2451
25821 25823 750
25823 25821 750
25824 25756 1114
25756 25824 1114
25825 25824 556
25824 25825 556
6589 25825 2561
25825 6589 2561
25823 6588 1173
6588 25823 1173
25826 6589 1344
6589 25826 1344
25823 25827 750
25827 25823 750
25828 25823 1533
```

Working of the project

This project contains three runnable files:

1.Dijkstra.cpp

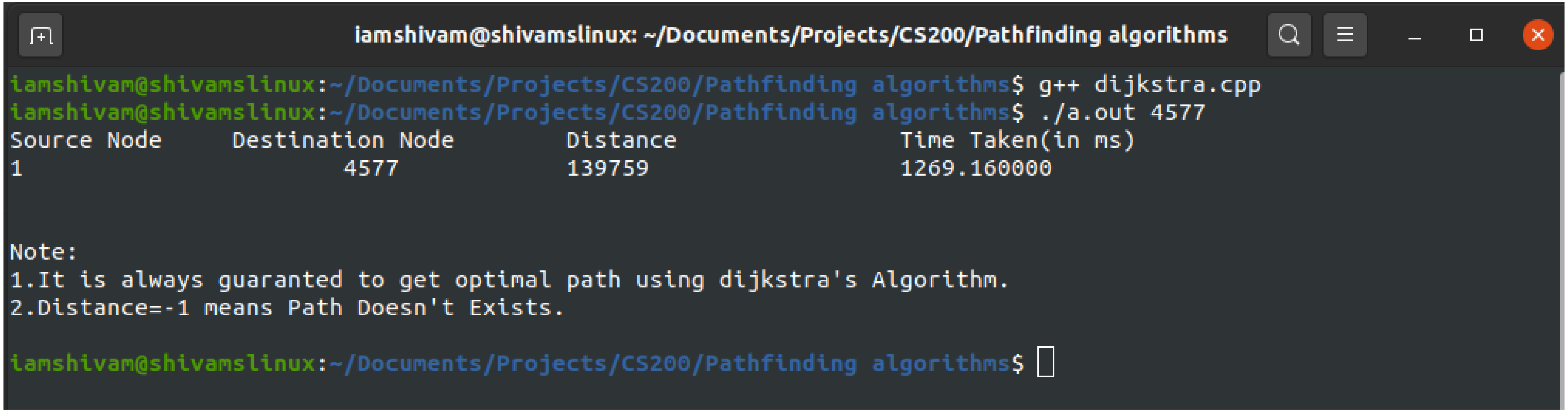
2.Astar.cpp

3.run.sh

Running dijkstra.cpp

This file reads the dataset of nodes and edges and form the corresponding graph.This file also takes the destination city (destination node) to which path is to be found.And then this file displays the time taken by the algorithm to find the path and distance of the resultant shortest path.

OUTPUT:

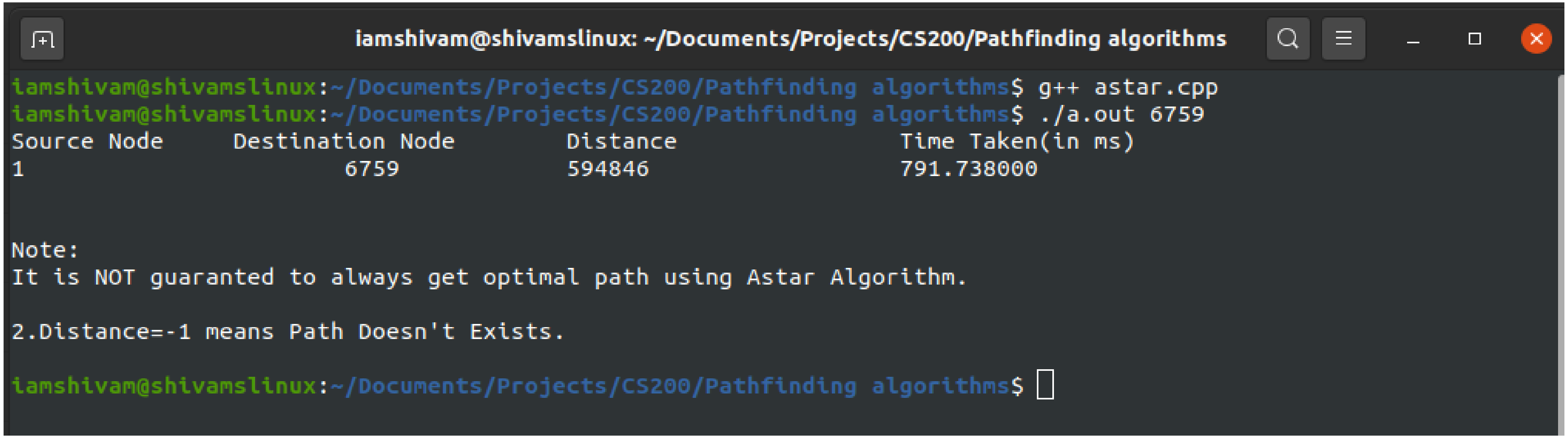


```
iamshivam@shivamslinux:~/Documents/Projects/CS200/Pathfinding algorithms$ g++ dijkstra.cpp
iamshivam@shivamslinux:~/Documents/Projects/CS200/Pathfinding algorithms$ ./a.out 4577
Source Node     Destination Node        Distance                Time Taken(in ms)
1                       4577            139759                  1269.160000


Note:
1.It is always guaranted to get optimal path using dijkstra's Algorithm.
2.Distance=-1 means Path Doesn't Exists.

iamshivam@shivamslinux:~/Documents/Projects/CS200/Pathfinding algorithms$ 
```

Astar.cpp

This file reads the dataset of nodes and edges and form the corresponding graph.This file also takes the destination city (destination node) to which path is to be found.And then this file displays the time taken by the algorithm to find the path and distance of the resultant shortest path.Astar algorithm doesn't always guarantees the shortest path as it depends on the heuristic function taken.

OUTPUT:



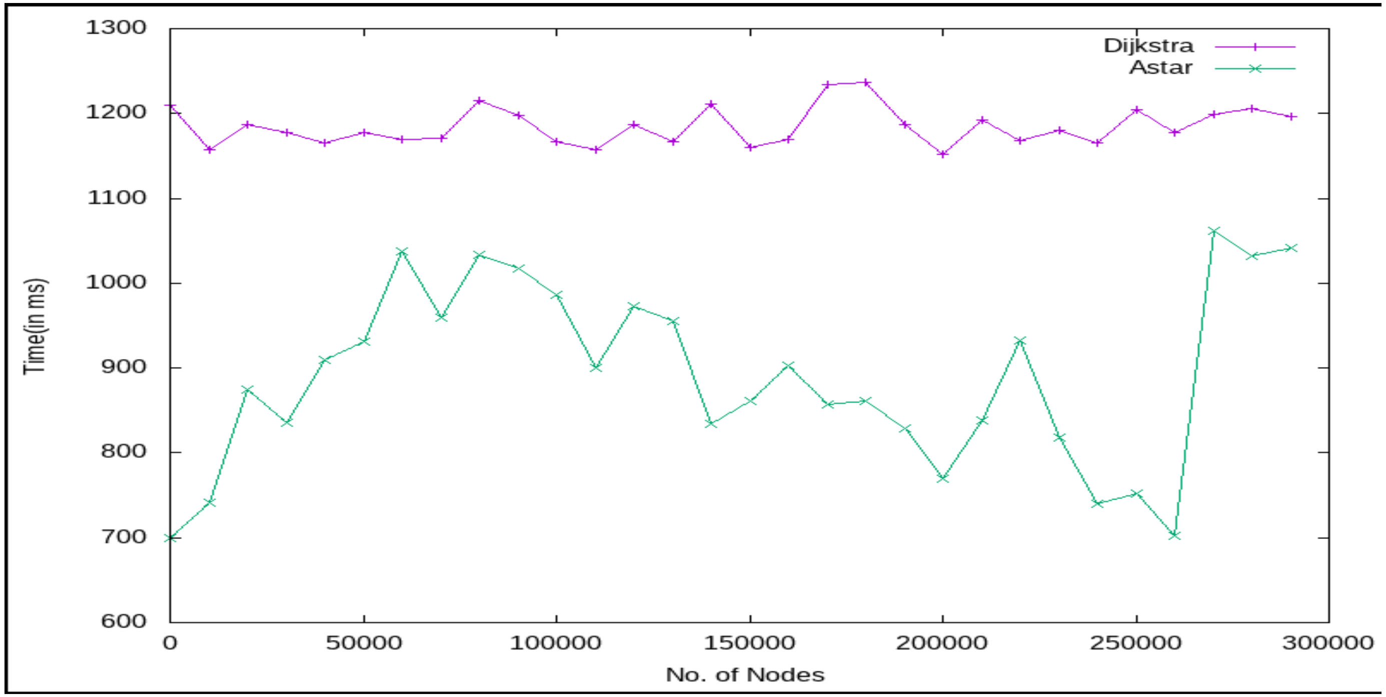
iamshivam@shivamslinux: ~/Documents/Projects/CS200/Pathfinding algorithms

```
iamshivam@shivamslinux:~/Documents/Projects/CS200/Pathfinding algorithms$ g++ astar.cpp
iamshivam@shivamslinux:~/Documents/Projects/CS200/Pathfinding algorithms$ ./a.out 6759
Source Node     Destination Node        Distance                Time Taken(in ms)
1                       6759            594846                  791.738000


Note:
It is NOT guaranted to always get optimal path using Astar Algorithm.

2.Distance=-1 means Path Doesn't Exists.

iamshivam@shivamslinux:~/Documents/Projects/CS200/Pathfinding algorithms$ 
```

Run.sh

This file analyse the Dijkstra's Shortest path algorithm and Astar shortest path algorithm over a large dataset of road network of california state for time taken by both the algorithms to find the shortest path between two given cities(two nodes) of the graph.This file collects the time taken by both the algorithms and plot the graph between time taken and no. of nodes.

Output of run.sh



Bibliography

1.T.H.Cormen,C.E.Leiserson,R.L.Rivest and C Stein,*Introduction to Algorithms*.

2.https://theory.stanford.edu/~amitp/GameProgramming/AstarComparison.html,*Introduction to A*.*

THANK YOU!