Dijktra’s shortest path algorithm essay

| | Kingdom of Saudi Arabia Ministry of Higher Education Jazan University, Jazan | | A DIJKSTRA SHORTEST PATH ALGORITHM Faculty of Computer Science & Information system ABSTRACT Dijkstra Algorithm is utilized to compute shortest path in a network. It is essential to explain the different types of connected, directed and weighted graphs. Tree is also discussed with the help of graph.

Dijkstra shortest path is shown and described with the example. Pseudo code and algorithm are also included along with their efficiency and applications. The different aspects of related algorithms were discussed, such as A* algorithm, Bellman–Ford algorithm and Prim’s algorithm Keywords – Algorithm, Network, Tree, Pseudo code INTRODUCTION Graphs ? G (V, E) where: – V is vertex set – E is edge set: every e in E connects two vertices -If (i, j) connected they are said adjacent -Edge e = (i, j) is said incident to i and j -| V| = cardinality of the vertex set ? A graph G(V, E) can be represented by | V| x | V| adjacency matrix ? A path between vertices i and j is a sequence of vertices and edges starting with vertex i and ending in j, such that every edge is incident to the preceding and following vertices. ? A path is simple if every node/edge appears only once. ? Distance between vertices i and j: minimum number of edges along a path from i to j ? Cycle: simple path where starting vertex coincides with ending vertex ? A graph is connected if there exists a path between any possible vertex pair i and j ? A graph is directed if edges have a direction.

In this case, (i, j) belonging to E does not imply that (j, i) belongs to E. Edges are called arcs in this case ? A graph is weighted if every edge (or arc) comes with a number . Tree Graph T is a tree if: ? One and only one simple

path between every vertex pair (i, j) ? If N = | V|: Exactly N1 edges (arcs) Connected, no cycles ? Every vertex of a tree can be assumed to be the ? root of the tree ? A tree can be represented by arranging vertices in sequential layers of increasing distance from root ? In a layered representation of a tree Every vertex (except for the root), has only one parent vertex – Every vertex has 0 or more children – A node with 0 children is a leaf Dijkstra’s Algorithm: Dijkstra’s algorithm is called the single-source shortest path. It is also known as the single source shortest path problem. It computes length of the shortest path from the source to each of the remaining vertices in the graph. The single source shortest path problem can be described as follows: Let G= {V, E} be a directed weighted graph with V having the set of vertices.

The special vertex s in V, where s is the source and let for any edge e in E, EdgeCost(e) be the length of edge e. All the weights in the graph should be non-negative. Before going in depth about Dijkstra’s algorithm let’s talk in detail about directed-weighted graph. Directed graph can be defined as an ordered pair G: = (V, E) with V is a set, whose elements are called vertices or nodes and E is a set of ordered pairs of vertices, called directed edges, arcs, or arrows. Directed graphs are also known as digraph. pic] Figure of Directed graph DESCRIPTION OF THE ALGORITHM Before going into details of the pseudo-code of the algorithm it is important to know how the algorithm works.

Dijkstra’s algorithm works by solving the sub problem k, which computes the shortest path from the source to vertices among the k closest vertices to the source. For the dijkstra’s algorithm to work it should be directed- weighted

graph and the edges should be non-negative. If he edges are negative then the actual shortest path cannot be obtained. At the kth round, there will be a set called Frontier of k vertices that will consist of the vertices closest to the source and the vertices that lie outside frontier are computed and put into New Frontier. The shortest distance obtained is maintained in sDist[w]. It holds the estimate of the distance from s to w. Dijkstra’s algorithm finds the next closest vertex by maintaining the New Frontier vertices in a priority-min queue.

The algorithm works by keeping the shortest distance of vertex v from the source in an array, sDist. The shortest distance of the source to itself is zero. Dist for all other vertices is set to infinity to indicate that those vertices are not yet processed. After the algorithm finishes the processing of the vertices sDist will have the shortest distance of vertex w to s. two sets are maintained Frontier and New Frontier which helps in the processing of the algorithm.

Frontier has k vertices which are closest to the source, will have already computed shortest distances to these vertices, for paths restricted up to k vertices. The vertices that resides outside of Frontier is put in a set called New Frontier. PSEUDO-CODE OF THE ALGORITHMThe following pseudo-code gives a brief description of the working of the Dijkstra’s algorithm. Procedure Dijsktra (V: set of vertices 1… n {Vertex 1 is the source} Adj[1…n] of adjacency lists; EdgeCost(u, w): edge – cost functions;) Var: sDist[1…n] of path costs from source (vertex 1); {sDist[j] will be equal to the length of the shortest path to j} Begin: Initialize {Create a virtual set Frontier to store i where sDist[i] is already fully solved} Create empty Priority Queue New

Frontier; sDist[1] forall vertices w in V – {1} do {no edges have been explored yet} sDist[w]<> nd for; Fill New Frontier with vertices w in V organized by priorities sDist[w]; endInitialize; repeat v if sDist[w]; sDist[v] +EdgeCost(v, w) then sDist[w] update w in New Frontier {with new priority sDist[w]} endif endfor until New Frontier is empty end Dijkstra; The algorithm illustrates Best-First-Breadth-First-Search. It is the Best-First because the best vertex is New Frontier is selected to be processed next.

The search used is Breadth-First, since New Frontier consists of vertices that can be tried next and these vertices are one edge away from the explored vertices. EXAMPLE The above algorithm can be explained and understood better using an example. The example will briefly explain each step that is taken and how shortest Distance is calculated.

Consider the following example: [pic] Figure: Weighted-directed graph The above weighted graph has 5 vertices from A-E. The value between the two vertices is known as the edge cost between two vertices. For example the edge cost between A and C is 1. Using the above graph the Dijkstra’s algorithm is used to determine the shortest path from the source A to the remaining vertices in the graph. The example is solved as follows: ? Initial step sDist[A]= 0 ; the value to the source itself sDist[B]= ? , sDist[C]= ? , sDist[D]= ? , sDist[E]= ? ; the nodes not processed yet ? Step 1 Adj[A]={B, C}; computing the value of the adjacent vertices of the graph sDist[B]= 4; sDist[C]= 2; [pic] Figure: shortest path to vertices B, C from A ? Step 2 Computation from vertex C Adj[C] = {B, D}; sDist[B] ; sDist[C]+ EdgeCost[C, B] 4 ; 1+2 (True) Therefore, sDist[B]= 3; sDist[D]= 2; [pic] Figure: Shortest path from B, D using C as intermediate vertex Adj[B]={E}; sDist[E]= sDist[B]

+EdgeCost[B, E] 3+3= 6; [pic] Figure: Shortest path to E using B as intermediate vertex Adj[D]={E}; sDist[E]= sDist[D]+EdgeCost[D, E] = 3+3= 6 This is same as the initial value that was computed so sDist[E] value is not changed. ? Step 4 Adj[E]= 0; means there is no outgoing edges from E And no more vertices, algorithm terminated.

Hence the path which Follows the algorithm is [pic] Figure: the path obtained using Dijkstra’s Algorithm EFFICIENCY The complexity/efficiency can be expressed in terms of Big-O Notation. Big-On gives another way of talking about the way input affects the algorithm’s running time. It gives an upper bound of the running time. In Dijkstra’s algorithm, the efficiency varies depending on | V|= n DeleteMins and | E| updates for priority queues that were used. If a Fibonacci heap was used then the complexity is O(| E | + | V | log | V |) , which is the best bound. The DeleteMins operation takes O(log| V|). If a standard binary heap is used then the complexity is O(| E | log | E|),| E | log | E| term comes from| E| updates for the standard heap .

If the set used is a priority queue then the complexity is O(| E|+| V| 2). O(V| 2) term comes from | V| scans of the unordered set New Frontier to find the vertex with the least sDist value. ADVANTAGE of DIJKSTRA Once it has been carried out you can find the least weight path to all permanently labeled nodes. You don’t need a new diagram for each pass. Dijkstra’s algorithm has an order of n2 so it is efficient enough to use for relatively large problems. DIS-ADVANTAGES The major disadvantage of the algorithm is the fact that it does a blind search there by consuming a lot of time waste of necessary resources. Another disadvantage is that it cannot handle negative edges.

This leads to cyclic graphs and most often cannot obtain the right shortest path.

RELATED ALGORITHMS ? A* algorithm is a graph/tree search algorithm that finds a path from a given initial node to a given goal node It employs a “ heuristic estimate” h(x) that gives an estimate of the best route that goes through that node. It visits the nodes in order of this heuristic estimate. It follows the approach of best first search. ? The Bellman–Ford algorithm computes single-source shortest paths in a digraph. It uses the same concept as that of Dijkstra’s algorithm but can handle negative edges as well. It has a better running time than that of Dijkstra’s algorithm. ? Prim’s algorithm finds a minimum spanning tree for a connected weighted graph.

It implies that it finds a subset of edges that form a tree where the total weight of all the edges in the tree is minimized. it is sometimes called the DJP algorithm. APPLICATIONS ? Traffic information systems use Dijkstra’s algorithm in order to track the source and destinations from a given particular source and destination ? OSPF- Open Shortest Path First, used in Internet routing. It uses a link-state in the individual areas that make up the hierarchy. The computation is based on Dijkstra’s algorithm which is used to calculate the shortest path tree inside each area of the network.

REFERENCES 1. http://tide4javascript.

com/? s= Dijkstra 2. http://www. cs. sunysb.

edu/~skiena/combinatorica/animations/dijkstra. html 3. http://www. unf.

edu/~wkloster/foundations/DijkstraApplet/DijkstraApp l et. htm 4.

http://www. cs. auckland. ac. nz/software/AlgAnim/dijkstra.

html 5. http://www. cs. mcgill. ca/~cs251/OldCourses/1997/topic29/ 6.

http://en. wikipedia.

org/wiki/Dijkstra’s_algorithm