PATH FINDING - Dijkstra's Algorithm

Venkata Chaitanya Chunduri

Indiana State University

vchunduri@sycamores.indstate.edu

December 13, 2014

Abstract

Dijkstra Algorithm is one of the most famous algorithms in computer science. Back before computers were a thing, around 1956, Edsger Dijkstra came up with a way to find the shortest path within a graph whose edges were all nonnegative values. To this day, almost 50 years later, his algorithm is still being use d in things such as link-state routing. It has been extended by others to create more advanced path finding algorithms such as A*.As far as optimal solutions (paths) are concerned, Algorithms designed for the shortest path problems should be capable of handling three cases. An optimal solution exists.No optimal solution exists because there are no feasible solutions. No optimal solution exists because the length of feasible paths from city 1 to city n is unbounded from below.

Keywords: Dijkstra Algorithm, Shortest Path, Link-State Routing, Path Finding Algorithms.

1 Dijkstra's - A Greedy Approach

Approach of the algorithm is iterative and also maintains shortest path with each intermediate nodes.
Greedy algorithms use problem solving methods based on actions to see if there is a better long term strategy. Dijkstra algorithm uses the greed y approach to solve the single source shortest problem. It repeatedly selects from the unselected vertices, vertex v nearest to source s and declares the distance to be the actual shortest distance from s to v. The edges of v are then checked to see if their destination can be reached by v followed by the relevant t outgoing edges.

Here is the greedy idea of Dijkstra algorithm:

- Maintain a set S of vertices whose shortest-path from s are known ($s \in S$ initially).
- At each step add vertex v from the set V-S to the set S. Choose v that has minimal distance from s (be greedy).
- Update the distance estimates of vertices adjacent to v.

Edsger Wybe Dijkstra
1930 –2002

"Computer Science is no more about computers than astronomy is about telescopes."

2 History on Dijkstra‘s

Edsger Dijkstra is Dutch.
He is one of the big names in computer science. He is known for his handwriting and quotes such as:

- Simplicity is prerequisite for reliability.
- The question of whether machines can think is about as relevant as the question of whether submarines can swim.

Dijkstra‘s algorithm was created in 1959 by Dutch Computer Scientist Edsger Dijkstra. While employed at the Mathematical Center in Amsterdam, Dijkstra was asked to demonstrate the powers of ARMAC, a sophisticated computer system developed by the mathematical Center. Part of his presentation involved illustrating the best way to travel between two points and in doing so, the shortest path algorithm was created. It was later on renamed as Dijkstra‘s Algorithm in recognition of its creator.
In particular, we are reminded that this famous algorithm was strongly inspired by Bellman‘s principle of optimality and that both conceptually and technically it constitutes a dynamic programming successive approximation procedure par excellence.

3 Implementation with example

Consider the following example:

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 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] equals Infinity; The nodes not processed yet.

STEP 1 :

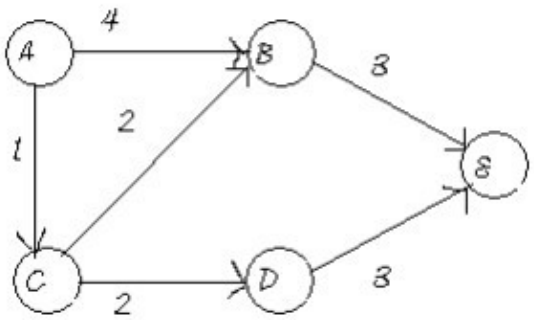

Figure: Weighted-directed graph

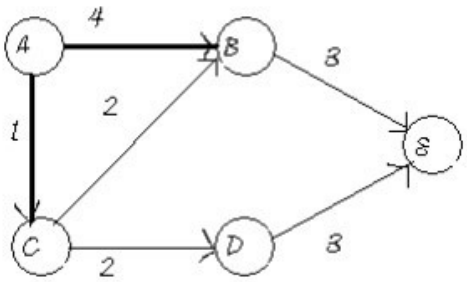

Figure: shortest path to vertices B, C from A

sDist[B] = 4; sDist[C] = 2;

STEP 2 :
sDist[B] = 3; sDist[D] = 2;

STEP 3 :
sDist[E] = 6;

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 :

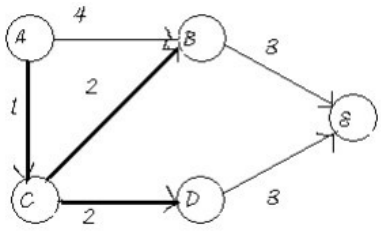

Figure: Shortest path from B, D using C as intermediate vertex

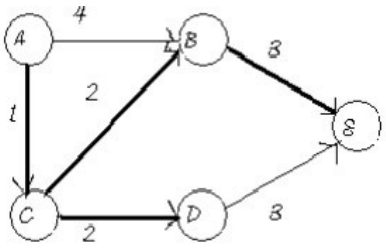

Figure: Shortest path to E using B as intermediate vertex

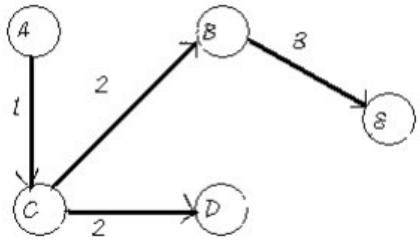

Figure: the path obtained using Dijkstra's Algorithm

4 Pseudocode for Dijkstra‘s

function Dijkstra(Graph, source):
dist[source] := 0 // Distance from source to source
for each vertex v in Graph: // Initializations
if v ≠ source
dist[v] := infinity // Unknown distance function from so previous[v] := undefined // Previous node in optimal path from source
end if
add v to Q // All nodes initially in Q (unvisited nodes)
end for
while Q is not empty: // The main loop
u := vertex in Q with min dist[u] // Source node in first case
remove u from Q
for each neighbor v of u: // where v has not yet been removed from Q.
alt := dist[u] + length(u, v)
if alt ¡ dist[v]: // A shorter path to v has been found
dist[v] := alt
previous[v] := u
end if
end for
end while
return dist[], previous[]
end function

5 Efficiency of Dijkstra

The complexity/efficiency can be expressed in terms of Big-O notation. Big-O gives another way of talking about the way inputs affects the algorithm‘s running time. It gives an upper bound of the running time.
In Dijkstra‘s algorithm, the effciency 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 stand and 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 least sDist value.

6 Advantages

- Once it has been carried out you can find the least weight path to all permanently labelled 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.

7 Disadvantages

- There is a problem with this algorithm - it can only see the neighbors of the immediate node. The issue that can arise is if you choose a short node that is forked. Since the algorithm is not backtracking, it can potentially degrade into an infinite loop, especially since it will eventually run out of suitable neighbors to inspect all while knowing that not all nodes have been visited.
- The major problem of the algorithm is the fact that it does a blind search there by consuming a lot of time waste of necessary resources.
- Another problem is that it cannot handle negative edges. This leads to acyclic graphs and most often cannot obtain the right shortest path.

8 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 weighted digraph. It uses the same concept as that of Dijkstra‘s algorithm m but can handle negative edges as well. It has a better running time than that of Dijkstra‘s algorithm.

- **Prims'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 or jarnik algorithm.

9 Improvements

- Easy to implement.
- Reduces cost to Edlog(V).
- Indistinguishable from linear for huge sparse graphs found in practice.
- Use a Fibonacci heap (Sleator-Tarjan, 1980s)
- Very difficult to implement.
- Reduces worst-case costs(in theory) to E + V logV
- Not quite linear
- Practical utility questionable.

10 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.
- Robot path planning.
- Logistics Distribution lines.
- IS-IS (Intermediate system to intermediate system)

11 Limitations

One thing we haven't looked at is the problem of finding shortest paths that must go through certain points. This is a hard problem and is reducible to the Travelling Salesman problem, what this means in practice is that it can take a very long time to solve the problem even for very small inputs.

12 References

- http://en.wikipedia.org/wiki/Pathfinding
- http://en.wikipedia.org/wiki/Dijkstra's algorithm
- http://math.mit.edu/řothvoss/18.304.3PM/Presentations/1-Melissa.pdf
- http://www.cs.nyu.edu/courses/summer07/G22.2340-001/Presentations/Puthuparampil.pdf
- http://courses.csail.mit.edu/6.006/spring11/lectures/lec16.pdf
- https://www.youtube.com/watch?v=0nVYi3o161A
- https://www.youtube.com/watch?v=8Ls1RqHCOPw
- http://www.ijaiem.org/volume2issue7/IJAIEM-2013-07-23-079.pdf
- http://www.thestudentroom.co.uk/wiki/Revision:The_Shortest_Path