Efficiency/Complexity- Dijkstra’s Algorithm

December 11, 2013

1 Efficiency

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 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 $\bigcirc(|E| + |V| \log |V|)$, which is the best bound. The DeleteMins operation takes $\bigcirc(\log |V|)$.

If a Standard binary heap is used then the complexity is $\bigcirc(|E| \log |E|)$, $|E| \log |E|$ term comes from $|E|$ updates for the standard heap.

If the set used is a priority queue then the complexity is $\bigcirc(|E| + |V|^2)$. $\bigcirc(|V|^2)$ term comes from $|V|$ scans of the unordered set New Frontier to find the vertex with least sDist value.

2 Pitfalls

There’s 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.
3 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 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.

4 Improvements

Use a d-way heap(Johnson, 1970s)

- easy to implement.
- reduces cost to $Ed \log_d(V)$.
- 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 $E + V \log V$
- not quite linear
- practical utility questionable.

5 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.
6 Limitations

Once 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.