The best shortest path algorithm

Ask Question

What is the difference between the **"Floyd-Warshall algorithm"** and **"Dijkstra's Algorithm"**, and which is the best for finding the shortest path in a graph?

I need to calculate the shortest path between all the pairs in a net and save the results to an array as follows:

```
**A     B     C     D      E**
A 0     10    15    5      20
B 10     0    5     5      10
C 15     5    0     10     15
D 5      5    10    0      15
E 20     10    15   15     0
```

algorithm shortest-path

edited Apr 17 '17 at 7:47

iled

1,556 2 20 33

asked Dec 4 '09 at 13:06

ricardo

2,801 8 20 37

but the other one was closed, mostly because of the user's bad english, and one of the solutions named these exact two algorithms as alternatives. If we close this as dup, how will the author find out more about the previous question? Will we really all be nice enough to go over there and vote to reopen? – Will Dec 4 '09 at 13:15

hi sorry, but wanted to add an array example with respect to a picture but I did not do – ricardo Dec 4 '09 at 13:17

thanks, SilentGhost for re-edit my question – ricardo Dec 4 '09 at 13:20

2 Shouldn't DE in that graph be 15?

7 Answers

Dijkstra's algorithm finds the shortest path between a node and every other node in the graph. You'd run it once for every node. Weights must be non-negative, so if necessary you have to normalise the values in the graph first.

Floyd-Warshall calculates the shortest routes between all pairs of nodes in a single run! Cycle weights must be non-negative, and the graph must be *directed* (your diagram is not).

Johnson's algorithm is using Dijkstra's algorithm to find all pairs in a single pass, and is faster for sparse trees (see the link for analysis).

edited Jan 10 '13 at 19:22

answered Dec 4 '09 at 13:22

Will
47.4k 30 137 207

From the wikipedia link you cite for Dijkstra: "the algorithm finds the path with lowest cost between that vertex and **every** other vertex" (my emphasis). You thus don't need to run it for every pair of vertex but only for every vertex. – Andreas Brinck Dec 4 '09 at 13:43

thx Andreas, fixed – Will Dec 4 '09 at 13:46

9 You can convert an undirected graph to a directed graph by replacing every edge uv with two edges (u,v) and (v,u) with the same weight. Then presumably Floyd-Warshall should work just fine? – Nick Lewis Dec 4 '09 at 13:50

2 err .. floyd-warshall does not require it to have non-negative edges, from wikipedia "is a graph analysis algorithm for finding shortest paths in a weighted graph

Terms of Service.

Main Purposes:

Dijkstra's Algorithm is one example of a single-source shortest or SSSP algorithm, i.e., given a source vertex it finds shortest path from source to all other vertices. Floyd Warshall Algorithm is an example of all-pairs shortest path algorithm, meaning it computes the shortest path between all pair of nodes.

Time Complexities :

Time Complexity of Dijkstra's Algorithm: O(E log V)

Time Complexity of Floyd Warshall: O(V3)

Other Points:

We can use Dijskstra's shortest path algorithm for finding all pair shortest paths by running it for every vertex. But time complexity of this would be O(VE Log V) which can go (V3 Log V) in worst case.

Another important differentiating factor between the algorithms is their working towards distributed systems. Unlike Dijkstra's algorithm, Floyd Warshall can be implemented in a distributed system, making it suitable for data structures such as Graph of Graphs (Used in Maps).

Lastly Floyd Warshall works for negative edge but no negative cycle, whereas Dijkstra's algorithm don't work for negative edges

answered Jun 22 at 6:07

Codemaker

637 3 12

In the meanwhile better algorithms for the single source shortest path

Hagerup. The algorithm has the same worst case complexity as Djikstra's, but in the average case the expected runtime is linear in the size of the graph, which is much faster than the pure Dijkstra. The idea of the algorithm is based on the idea, that there is no need to always poll the minimum edge from the queue. It is possible poll an edge from the queue, whose weight is `1+k` times as large as the minimum edge weight, where `k` is some number larger `0`. Even if such an edge is chosen, the algorithm will still find the shortest path.

answered Oct 17 '12 at 5:34

SpaceTrucker
8,497 5 37 78

Dijkstra's is mainly for single pair shortest path finding i.e. from one node to all other nodes, where as Floyd-Warshall is for all-pair shortest path i.e. shortest path between all pair of vertices. The Floyd-Warshall algorithm has a worst case performance of O(|V|3), where as Dijkstra's has a worse case performance of O(|E| + |V|log |V|) Also Dijkstra's cannot be used for negative weights (we use Bellmann Ford for the same). but for Floyd-Warshall we can use negative weights but no negative cycles

answered Jul 11 '12 at 8:09

piyush_raman
172 1 5

Floyd Warshall find the paths between all pairs of vertices, but Dijkstra only finds the path from one vertex to all others.

Floyd Warshall is $O(|V|^3)$ and Dikstra is O(|E| + |V| log |V|) but you'll have to

possibly faster to use Dijsktra repeatedly than the FW algorithm, I would try both approaches and see which one is fastest in the actual case.

edited Dec 4 '09 at 14:22

u0b34a0f6ae
31.2k 10 77 95

answered Dec 4 '09 at 13:11

Andreas Brinck
37.2k 13 70 106

Francis Haart's comment: "@Andreas Brinck, in a complete graph, E=(V^2-V)/2, and dijkstra's would be no faster." – Peter O. Feb 15 '12 at 0:15

Dijkstra finds the shortest path from only **one** vertex, Floyd-Warshall finds it between **all** of them.

answered Dec 4 '09 at 13:13

Gergely Orosz
5,365 3 40 58

Use the Floyd-Warshall algorithm if you want to find the shortest path between **all** pairs of vertexes, as it has a (far) higher running time than Dijkstra's algorithm.

The Floyd-Warshall algorithm has a worst case performance of $O(|V|^3)$, where as Dijkstra's has a worse case performance of $O(|E| + |V|\log |V|)$

answered Dec 4 '09 at 13:11

Yacoby
44.1k 11 96 112