This paper appeard in: Information Processing Letters 110 (2010), 279-281

The Floyd-Warshall Algorithm on Graphs with Negative Cycles

Stefan Hougardy

Research Institute for Discrete Mathematics, University of Bonn, Lennéstr. 2, 53113 Bonn, Germany

Abstract

The Floyd-Warshall algorithm is a simple and widely used algorithm to compute shortest paths between all pairs of vertices in an edge weighted directed graph. It can also be used to detect the presence of negative cycles. We will show that for this task many existing implementations of the Floyd-Warshall algorithm will fail because exponentially large numbers can appear during its execution.

Keywords: Floyd-Warshall algorithm, design of algorithms, graph algorithms

1. Introduction

The Floyd-Warshall algorithm [3] (see Figure 1) is a simple and widely used algorithm to compute shortest paths between all pairs of vertices in an edge weighted directed graph. This algorithm has a worst-case runtime of $O(n^3)$ for graphs with n vertices. There exist several algorithms with a better worst-case runtime [4, 11, 6, 12, 16, 13, 2, 7, 1, 5], the best of these algorithms currently achieve a runtime of $O(n^3 \log\log n/\log^2 n)$ [5] respectively $O(mn+n^2 \log\log n)$ [9]. However, these algorithms are much more complicated than the Floyd-Warshall algorithm and involve complicated data structures. Therefore, in many cases the Floyd-Warshall algorithm is still the best choice.

The Floyd-Warshall algorithm outputs the correct result as long as no negative cycles exist in the input graph. In case that a negative cycle exists, computing a shortest (simple) path is an NP-hard problem (see e.g. [8]) and the Floyd-Warshall algorithm will not output the correct result. Rather, it will detect the presence of a negative cycle by checking that there is a negative entry in the diagonal of the matrix M (lines 8 and 9 in Figure 1). In many widely used implementations this is done as shown in Figure 1. See for example the books [14, 10] or the implementation and the comments given in the section "Behaviour with negative cycles" in Wikipedia [15]. We will show that for such implementations there exist simple examples in which the matrix M can have

Email address: `hougardy@or.uni-bonn.de` (Stefan Hougardy) exponentially large entries and thus an overflow may occur during the execution of the Floyd-Warshall algorithm. As we will see the numbers in M not only can double but can grow by a factor of six in each iteration.

Floyd-Warshall-Algorithm

Input: A digraph G with $V(G) = \{1, \ldots, n\}$ and weights $c : E(G) \to \mathbb{R}$

Output: An $n \times n$ matrix M such that $M[i,j]$ contains the length of a shortest path from vertex i to vertex j.

```
1 $M[i,j] := \infty \ \forall i \neq j$
2 $M[i,i] := 0 \ \forall i$
3 $M[i,j] := c((i,j)) \ \forall (i,j) \in E(G)$
4 for $i := 1$ to $n$ do
5     for $j := 1$ to $n$ do
6         for $k := 1$ to $n$ do
7             if $M[j,k] > M[j,i] + M[i,k]$ then $M[j,k] := M[j,i] + M[i,k]$
8 for $i := 1$ to $n$ do
9     if $M[i,i] < 0$ then return('graph contains a negative cycle')
```

Figure 1: The Floyd-Warshall algorithm for computing the lengths of shortest paths between all pairs of vertices in a directed graph with possibly negative edge weights.

2. Exponentially large numbers in the Floyd-Warshall algorithm

In the following let $G = (V, E)$ be a directed graph with $|V| = n$ and let $c : E(G) \to \mathbb{R}$ be an arbitrary weight function on the edges of G. By $c_{\max}$ we denote the largest absolute value that an edge weight in G has, i.e., $c_{\max} := \max_{e \in E(G)}\{|c(e)|\}$. Using this value one can easily bound the largest number that appears in the matrix M during the execution of the Floyd-Warshall algorithm under the assumption that no negative cycle exists in the input graph. We denote by $||M||_{\max}$ the maximum norm of the matrix M where we ignore entries with the symbolic value of ∞, i.e., $||M||_{\max} := \max_{i,j}\{|M[i,j]| \text{ with } M[i,j] \neq \infty\}$.

Proposition 1. *If the input graph of the Floyd-Warshall algorithm does not contain a negative cycle then*

$$||M||_{\max} \ \leq \ n \cdot c_{\max}$$

at any time during the execution of the algorithm.

Proof. If no negative cycle exists in the input graph then at any time of the execution of the Floyd-Warshall algorithm an entry $M[i,j] \neq \infty$ contains the length of some path from vertex i to vertex j. As such a path can consist of at most $n-1$ edges the result follows.

In contrast to the above result we will now prove that the entries of the matrix M can become exponentially large, if the input graph is allowed to contain negative cycles.

Theorem 2. *There exist graphs such that*

$$||M||_{\max} \;=\; 2\cdot 6^{n-1}\cdot c_{\max}$$

during the execution of the Floyd-Warshall algorithm.

Proof. Consider the following graph on vertices $1,\dots,n$ with edge set $E = \{(1,i)|1 \le i \le n\} \cup \{(i,1)|2 \le i \le n\}$. Let $c(e) := -1$ for all $e \in E$. We now prove by induction on the number i of executions of the outer for-loop in the Floyd-Warshall algorithm (line 4 in Figure 1) that the following equations hold:

$$\begin{aligned} M[i,i] &= -\frac{1}{3}\cdot 6^i && (1)\\ M[i,j] &= -\frac{1}{2}\cdot 6^i, \quad \text{for } j > i && (2)\\ M[j,i] &= -\frac{1}{2}\cdot 6^i, \quad \text{for } j > i && (3)\\ M[i+1,j] &= -6^i, \quad \text{for } j > i. && (4)\end{aligned}$$

For $i = 1$ this is easily seen to be true: For $j = 1$ line 7 of the algorithm sets $M[1,k]$ to the value $M[1,1]+M[1,k]$ for $k = 1,\dots,n$. This results in $M[1,1] = -1+(-1) = -2 = -\frac{1}{3}\cdot 6^1$ and $M[1,k] = -2+(-1) = -3 = -\frac{1}{2}\cdot 6^1$. Similarly we have $M[j,1] = M[j,1]+M[1,1] = -1+(-2) = -3 = -\frac{1}{2}\cdot 6^1$ for $2 \le j \le n$. Finally for $j > 1$ we have $M[2,j] = M[2,1]+M[1,j] = -3+(-3) = -6^1$.

For $i > 1$ we use (4) to obtain $M[i,i] = M[i,i]+M[i,i] = -6^{i-1}+(-6^{i-1}) = -\frac{1}{3}\cdot 6^i$ which proves (1). From (4) we also get $M[i,j] = M[i,i]+M[i,j] = -\frac{1}{3}\cdot 6^i + (-6^{i-1}) = -\frac{1}{2}\cdot 6^i$ for $j > i$. Similarly $M[i,j] = M[j,i]+M[i,i] = -6^{i-1}+(-\frac{1}{3}\cdot 6^i) = -\frac{1}{2}\cdot 6^i$ for $j > i$. Finally we have $M[i+1,j] = M[i+1,i]+M[i,j] = -\frac{1}{2}\cdot 6^i + (-\frac{1}{2}\cdot 6^i) = -6^i$ for $j > i$.

Thus we get from (1): $M[n,n] = -\frac{1}{3}\cdot 6^n = -2\cdot 6^{n-1}$. As we have $c_{\max} = 1$ in our graph the result follows.

The graphs that are constructed in Theorem 2 contain one negative loop, i.e., a negative cycle of length 1. However, it is easily seen that exponentially large numbers can appear even if there do not exist short negative cycles: Simply subdivide all negative edges a suitable number of times. By doing so one can easily see that the largest entry of M still can grow as $\Omega(6^n\cdot c_{\max})$.

3. Conclusion

We have shown that during the execution of the Floyd-Warshall algorithm exponentially large numbers may occur, if the input graph contains negative cycles. Theorem 2 implies that even for graphs with less than 30 vertices and 60 edges with weight -1 it may happen that during the execution of the Floyd-Warshall algorithm numbers with absolute value larger than 2^{64} occur. This shows that for larger graphs it can be quite likely to have subgraphs causing an overflow.

Of course, there is a simple way to avoid this pitfall: Instead of checking for negative cycles at the end of the algorithm one can include lines 8 and 9 in Figure 1 in the for-loop in line 6 without increasing the worst-case runtime. Another possibility is to first use the Bellman-Ford algorithm [8] to detect negative cycles in $O(mn)$ and to start the Floyd-Warshall algorithm only if the input graph has no negative cycles. Most implementations of the Floyd-Warshall algorithm we have seen apply neither of these two solutions and therefore might fail if negative cycles exist in the input graph. With this note we want to make aware of this potential pitfall.

[1] Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. In *STOC07*, pages 590–598, 2007.

[2] Timothy M. Chan. All-pairs shortest paths with real weights in $O(n^3/\log n)$ time. *Algorithmica*, 50:236–243, 2008.

[3] Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962.

[4] Michael L. Fredman. New bounds on the complexity of the shortest path problem. *SIAM Journal on Computing*, 5(1):83–89, 1976.

[5] Yijie Han. An $O(n^3 \log\log n/\log^2 n)$ time algorithm for all pairs shortest paths. Manuscript, 2009.

[6] Yijie Han. A note of an $O(n^3/\log n)$ time algorithm for all pairs shortest paths. *Information Processing Letters*, 105:114–116, 2008.

[7] Yijie Han. An $O(n^3(\log\log n/\log n)^{5/4})$ time algorithm for all pairs shortest paths. *Algorithmica*, 51:428–434, 2008.

[8] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer-Verlag Berlin Heidelberg, fourth edition, 2008.

[9] Seth Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theoretical Computer Science*, 312:47–74, 2004.

[10] Edward M. Reingold, Jurg Nievergelt, and Narsingh Deo. *Combinatorial Algorithms: Theory and Practice*. Prentice-Hall, Inc., 1977.

[11] Tadao Takaoka. A new upper bound on the complexity of the all pairs shortest path problem. *Information Processing Letters*, 43:195–199, 1992.

[12] Tadao Takaoka. A faster algorithm for the all-pairs shortest path problem and its application. In K.-Y. Chwa and J.I. Munro, editors, *COCOON 2004*, volume 3106 of *Lecture Notes in Computer Science*, pages 278–289. Springer-Verlag Berlin Heidelberg, 2004.

[13] Tadao Takaoka. An $O(n^3 \log\log n/\log n)$ time algorithm for the all-pairs shortest path problem. *Information Processing Letters*, 96:155–161, 2005.

[14] Mark Allen Weiss. *Data Structures and Algorithm Analysis*. The Benjamin/Cummings Publishing Company, Inc., second edition, 1995.

[15] Wikipedia. Floyd-Warshall algorithm — Wikipedia, The Free Encyclopedia, 2009. [Online; accessed 20-November-2009].

[16] Uri Zwick. A slightly improved sub-cubic algorithm for the all pairs shortest paths problem with real edge lengths. *Algorithmica*, 46:181–192, 2006.