VARIATIONS ON A CLASSIC TASK

HORVÁTH, Győző – HORVÁTH, Gyula – ZSAKÓ, László, HU

Abstract: Although the Floyd–Warshall algorithm is a remarkable algorithm in the world of graphs, it is undeservedly underrepresented at programming contests. Such contests tend to prefer more efficient solutions even when the Floyd-Warshall algorithm could also be applied. This paper aims to show that this algorithm does offer great opportunities for problem setters of programming contests.

Keywords: programming contest, graphs, Floyd–Warshall algorithm.

1 Introduction

There are some classical algorithms that students are taught at universities or when being prepared for IT contests, but later they are likely to be forgotten and not really used except for examinations.

A graph algorithm, the Floyd-Warshall algorithm [1,2] could be a good example. It is included in the set of algorithms to be used at Informatics Olympiads [3]. It is, however, hardly ever used.

Probably, the reason for this is that this algorithm looks like one that you just need to learn and be able to write it down. There seems to be no space for problem-solving and creative thinking when you actually use it. We believe that this is absolutely not true; there is something to be done about using the basic algorithm.

2 The original Floyd-Warshall algorithm

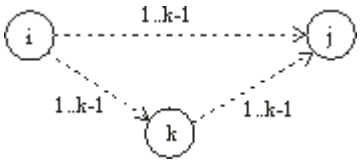
The basic task can be defined in two versions. In the first one, the edges of a graph are known, that is you know which pairs of vertices are connected with an edge. Now we want to determine which pairs of vertices have a path between them.

The transitive closure of a graph is:

$$E(i,j)=\begin{cases}\text{true} & \text{if} \quad \text{ExistPath } ?(i,j)\\ \text{false} & \text{otherwise}\end{cases}$$

An adjacency matrix contains the paths that do not have any intermediate vertices. This matrix is then transformed with N steps in such a way that we include more and more vertices as intermediate vertices. After the last step we will find E that we have been searching for, since thus all the vertices could have appeared as intermediate vertices.

Paths that cross maximum the first k vertices:



Besides setting the initial values, the point is:

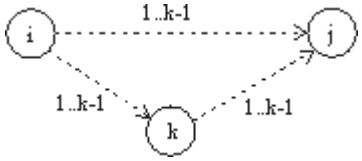
```
for k:=1 to N do
  for i:=1 to N do
    for j:=1 to N do
      E(i,j):=E(i,j) or (E(i,k) and E(k,j))
    Endfor
  Endfor
Endfor
```

In the second variant we know the length of each edge of the graph (weight, … any quantifiable value), and we would like to find the length of the shortest paths between all pairs of vertices.

Determining the distance matrix:

$$T(i,j)=\begin{cases}\text{Length\_of\_path}(i,j) & \text{if} \quad \text{ExistPath}?(i,j)\\ +\infty & \text{otherwise}\end{cases}$$

Paths that cross maximum the first k vertices: $T_{i,j}^{k}=\min\left(T_{i,j}^{k-1},T_{i,k}^{k-1}+T_{k,j}^{k-1}\right)$



The main point – apart from setting the initial values – is:

```
For k:=1 to N do
  For i:=1 to N do
    For j:=1 to N do
      T(i,j):=min(T(i,j),T(i,k)+T(k,j))
    Endfor
  Endfor
Endfor
```

The classic – and newer – algorithm books [4,5,6] end here.

3 Variations

Hardly ever can you find solutions that go beyond that [7,8]. Perhaps one of the nicest ones is the material by George Mason University [9], where 3 new problems are formulated:

Minimax - Minimax in graph problems involves finding a path between two nodes that minimizes the maximum cost along the path. (Example problems include finding feasible paths to take by car with a limited fuel tank and rest stations at every node).

Maximin - the other way around from Minimax - here you have problems where you need to find the path the maximizes the minimum cost along a path. (Example problems include trying to maximize the load a cargo truck can take when roads along a path may have a weight limit, or trying to find a network routing path that meets a minimum bandwidth requirement for some application).

Safest path - Similar in construction of Floyd-Warshall to minimax and maximin - Need to maximize the product of probabilities of survival along a path. Simply change the max to a min to find the most dangerous path.

The recurrence relations of these tasks are very similar to the Floyd-Warshall relation.

Minimax: $A_{i,j}^{k}=\min\left(A_{i,j}^{k-1},\max\left(A_{i,k}^{k-1},A_{k,j}^{k-1}\right)\right)$

Maximin: $B_{i,j}^{k}=\max\left(B_{i,j}^{k-1},\min\left(B_{i,k}^{k-1},B_{k,j}^{k-1}\right)\right)$.

Safest path: $C_{i,j}^{k}=\max\left(C_{i,j}^{k-1},C_{i,k}^{k-1}*C_{k,j}^{k-1}\right)$.

Similarly, two more problems can be devised.

Minimin – finding the paths where the shortest edge is the shortest possible.

Minimin: $F_{i,j}^{k}=\min\left(F_{i,j}^{k-1},\min\left(F_{i,k}^{k-1},F_{k,j}^{k-1}\right)\right)$

Maximax – finding the paths where the longest edge is the longest possible.

Maximax: $G_{i,j}^{k} = \max\left(G_{i,j}^{k-1}, \max\left(G_{i,k}^{k-1}, G_{k,j}^{k-1}\right)\right)$

Then we will only need to find and write a text for the task. For instance, suppose there is a multi-day running race between cities where runners are to run from one town to the neighbouring one every day, and we want to measure the participants' endurance. Therefore, we are looking for routes (i.e. paths) between pairs of towns where there is a day which requires them maximum endurance i.e. that is the longest stage is the longest possible.

```
For k:=1 to N do
  For i:=1 to N do
    For j:=1 to N do
      A(i,j):=min(A(i,j),max(A(i,k),A(k,j)))
      B(i,j):=max(B(i,j),min(B(i,k),B(k,j)))
      C(i,j):=max(C(i,j),C(i,k)*C(k,j))
      F(i,j):=min(F(i,j),min(F(i,k),F(k,j)))
      G(i,j):=max(G(i,j),max(G(i,k),G(k,j)))
    Endfor
  Endfor
Endfor
```

Now we can create new tasks without changing the essential part:

- The number of edges of the path containing the fewest edges for every pair of vertices – we will compute a distance matrix where the length of the edges of the graph is 1;
- We will compute a distance matrix for each vertex in the shortest circular graph in such a way that initially $T(i,i)=+\infty$!

There are fewer opportunities regarding Warshall's original algorithm. However, there is an idea: let us find for all pairs of vertices if there are at least two routes between them where at least one of the vertices is different!

$$E2_{i,j}^{k} = E2_{i,j}^{k-1} \vee \left(E2_{i,k}^{k-1} \wedge E_{k,j}^{k-1}\right) \vee \left(E_{i,k}^{k-1} \wedge E2_{k,j}^{k-1}\right) \vee \left(E_{i,j}^{k-1} \wedge E_{i,k}^{k-1} \wedge E_{k,j}^{k-1}\right)$$

Another similar task would be to find a path for the pairs of vertices consisting of even (or odd) number of steps!

4 Combining two algorithms

In each of the above cases the Floyd-Warshall algorithm gives the values of the optimal solution for all pairs of vertices. Since there may be several optimal solutions, it seems sensible to combine the above basic cases.

Of the shortest routes, for example, you can choose the one where the longest edge is the shortest possible. Then, if the shortest path from i to j goes via k, A(i,j) must be calculated via k. If there exists a shortest route with or without k, A(i,j) must be the minimum of the two calculations!

```
For k:=1 to N do
  For i:=1 to N do
    For j:=1 to N do
      If T(i,j)>T(i,k)+T(k,j)
         then T(i,j):=T(i,k)+T(k,j); A(i,j):=max(A(i,k),A(k,j))
      elseif T(i,j)=T(i,k)+T(k,j)
         then A(i,j):=min(A(i,j),max(A(i,k),A(k,j)))
    Endfor
```

```
  Endfor
Endfor
```

You can similarly combine any two or more calculations.

5 Calculating vertices

It is possible to associate the conditions necessary for the task not to the edges of the graph, but to its vertices. Thus, for instance, you can talk about the number of vertices of (i, j) path containing the fewest vertices. On principle, this will not include any new features: depending on whether the end vertices of a path are included in the number of vertices of a path or not, the solution based on the distance matrix computed with the number of edges is T(i,j)+1 or T(i,j)-1.

Using recurrence relation:

$$P_{i,j}^{k} = \min\left(P_{i,j}^{k-1}, P_{i,k}^{k-1} + P_{k,j}^{k-1} + 1\right) \text{ or } Q_{i,j}^{k} = \min\left(Q_{i,j}^{k-1}, Q_{i,k}^{k-1} + Q_{k,j}^{k-1} - 1\right).$$

In 2016 a problem was presented at a contest in Hungary where goods had to be transported between towns, and the intermediate towns imposed a duty on the shipment. It meant that those shortest paths had to be found where the sum of duties to be paid at the intermediate vertices (D_i) is the smallest possible (initial setting for H(i,j)=0 if there is an edge between i and j):

$$H_{i,j}^{k} = \min\left(H_{i,j}^{k-1}, H_{i,k}^{k-1} + H_{k,j}^{k-1} + D_{k}\right)$$

We believe that if in a graph model you assign weights to vertices instead of edges (e.g. to the activities – machines or phases of work – of a production line) and the edges denote only sequence relationships, the variations of the basic algorithm can also be re-interpreted for the vertices.

6 If the path also has to be given

For every (i,j) pair of vertices you should store a vertex k of the path that follows i, or precedes or is an intermediate vertex. Now we have chosen the one following i, but you could also choose the one preceding j, like in a teaching material by MIT [10].

```
Path:
  Start(T,First)
  For k:=1 to N do
    For i:=1 to N do
      For j:=1 to N do
        If T(i,j)>T(i,k)+T(k,j)
           then T(i,j):=T(i,k)+T(k,j); First(i,j):= First(i,k)
      Endfor
    Endfor
  Endfor
End.
Start(T,First)
  For i:=1 to N do
    For j:=1 to N do
      If T(i,j)=+∞ then First(i,j):=0 else First(i,j):=j
    Endfor
  Endfor
End.
```

7 Applications

Sometimes you can find tasks which require a variation of the Floyd-Warshall algorithm as the first step to the solution, and then the matrix resulting is to be worked with. Such an example could be the problem of finding a vertex for which the distance to the farthest neighbour is minimal.

```
Floy-Warshall(T); p:=1
For i:=1 to N do
  Max(i):=1
  For j=2 to N do
    If T(i,j)>T(i,Max(i)) then Max(i):=j
  Endfor
  If T(i,Max(i))<T(p,Max(p)) then p:=i
Endfor
```

A similar example is that of the central point, which can be defined as the vertex from which the average distance of other vertices, or the distance of the farthest one away from it is the shortest. [11]

8 Summary

The problems discussed above are based on applying the strategy of the classical Floyd-Warshall algorithm. When can this strategy be applied? In their book, Aho, Hopcroft and Ullmann [11] introduced the concept of the so-called closed semiring, which provides a general framework to tackle the problem of paths. Now we will describe only those qualities and conditions that are necessary in order that the strategy of the Floyd-Warshall algorithm can be applied to solve the problems discussed.

The input of the problem is a G=(V,E,w) directed weighted graph where the weight function w assigns w(p,q) from the value set K to each (p,q) edge. There are two binary operations, the min and the +, defined for the value set K. The weight function w is extended to paths in a way that it is the sum of the weight of edges under operation +. The distance δ(p,q) between any two – p and q – vertices of the graph is defined as the weight of the shortest path, i.e. the minimum under the min operation of the weight of all paths from p to q. The conditions are as follows:

1. The value set K has a key element; and let ∞ denote it. If there is no edge between vertices p and q, w(p,q)= ∞.
2. The value set K has a key element; let 0 denote it. The weight function is considered satisfied if w(p,p)=0 for every vertex p.
3. For operations + the following equalities are valid:
   3.1. x+y=y+x
   3.2. x+0=x, 0+x=x
   3.3. (x+y)+z=x+(y+z)
4. 4. For operations min the following equalities are valid:
   4.1. min(∞,x)=x=min(x, ∞)
   4.2. min(x,min(y,z))=min(min(x,y),z)
5. min(x,y)+z = min(x+z,y+z), x+min(y,z)=min(x+y, x+z)
6. ∞+x=∞, x+∞=∞

The above means that in any path problem – if the shortest path can be expressed by operations min or + where the above conditions are satisfied – the formula

$$T_{i,j}^{k} = \min\left(T_{i,j}^{k-1}, T_{i,k}^{k-1} + T_{k,j}^{k-1}\right)$$

will return the length of the shortest path from i to j that crosses maximum the first k vertices.

Conclusion

We believe that Floyd-Warshall algorithm variations and their applications are undeservedly underrepresented in programming contests, although they could be well used there, especially by those who are just getting started with graph algorithms. At contests this type of problem is rarely set (e.g. UVA online judge: 534 – Frogger, 10048 – Audiphobia [12]); and when it is, there usually would be a more efficient solution, as well.

The tasks and examples discussed here demonstrate the there are many possibilities for problem setters and the real difficulties are to create appropriate story of a real word problem that can be solved by this strategy.

Bibliography

[1] ROBERT W. FLOYD*: Algorithm 97: shortest path.* Communications of the ACM 5 (6): 345. June 1962.

[2] STEPHEN WARSHALL*: A theorem on boolean matrices.* Journal of the ACM 9 (1): 11–12. January 1962.

[3] TOM VERHOEFF, GYULA HORVÁTH, KRZYSZTOF DIKS, GORDON CORMACK, MICHAL FORISEK*: The international olympiad in informatics syllabus, 2013.* Http://ksp.sk/~misof/ioi-syllabus/ [Accessed on: Apr. 28, 2016]

[4] THOMAS H. CORMEN, CHARLES E. LEISERSON, RONALD L. RIVEST, CLIFFORD STEIN*: Introduction to algorithms.* The MIT press Cambridge, Massachusetts London, England, 2009.

[5] S. DASGUPTA, C.H. PAPADIMITRIOU, and U.V. VAZIRANI*: Algorithms. 2006.*

[6] MICHAEL SOLTYS*: An introduction to the analysis of algorithms.* World scientific publishing co. Pte. Ltd. Singapore, 2012.

[7] MAURICE POLLACK*: "The maximum capacity through a network",* Operations Research 8 (5): 733–736, 1960.

[8] VENKAT SASANK DONAVALLI*: Algorithms for the widest path problem. Rochester institute of technology, project proposal.* Https://people.rit.edu/svd2867/propsal.pdf [Accessed on: Apr. 28, 2016]

[9] *George Mason University motion and shape computing (masc) group's: Floyd-Warshall algorithm* http://masc.cs.gmu.edu/wiki/floydwarshall/ [Accessed on: Apr. 28, 2016]

[10] CHANDLER BURFIELD*: Floyd-Warshall algorithm,* february 20, 2013, http://math.mit.edu/~rothvoss/18.304.1pm/presentations/1-chandler-18.304lecture1.pdf [Accessed on: Apr. 28, 2016]

[11] ALFRED V. AHO, JOHN E. HOPCROFT, JEFFREY D. ULLMAN: *The Design and Analysis of Computer Programs.* Addison-Wesley Publishing Co., Inc. Boston, MA, USA, 1974

[12] *UVA Online Judge.* Http://uva.onlinejudge.org/ [Accessed on: Apr. 28, 2016]

Lectured by: Sándor Király, Dr. PhD.

Contact address:

Győző Horváth, Dr. Ph.D.,
Department of Media and Educational Informatics, Faculty of Informatics, Eötvös Loránd Univer-

sity, H-1117 Budapest, Pázmány P. sétány 1/C, Hungary,
phone: +36-1-372-2500 , e-mail: gyozke@elte.hu

Gyula Horváth, Dr.,
Department of Media and Educational Informatics, Faculty of Informatics, Eötvös Loránd University, H-1117 Budapest, Pázmány P. sétány 1/C, Hungary,
phone: +36-1-372-2500 , e-mail: horvath@inf.elte.hu

László Zsakó, Doc. Dr. hab. Ph.D,
Department of Media and Educational Informatics, Faculty of Informatics, Eötvös Loránd University, H-1117 Budapest, Pázmány P. sétány 1/C, Hungary,
phone: +36-1-372-2500 , e-mail: zsako@caesar.elte.hu