The Life and Times of Edsger Dijkstra

Madhavan Mukund
Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Brief Bio

- Born, Rotterdam, 1930
- Phd, Amsterdam, 1959
- Amsterdam, 1959-62
- TU Eindhoven, 1962-84
- UT Austin, 1984-99
- Died, Nuenen, 2002

Turing Award 1972

Citation

For fundamental contributions to programming as a high, intellectual challenge; for eloquent insistence and practical demonstration that programs should be composed correctly, not just debugged into correctness; for illuminating perception of problems at the foundations of program design.

Scientific contributions

- Shortest path algorithm
- Programming language design
- Program correctness and software engineering
- Operating systems
- Distributed computing

Shortest path algorithm

Numerische Mathematik 1, 269—271 (1959)

A Note on Two Problems in Connexion with Graphs

By

E. W. Dijkstra

Given a graph with non-negative edge weights, find the shortest path between a pair of vertices.

- Byproduct: shortest path to every vertex

Numerische Mathematik 1, 269—271 (1959)

A Note on Two Problems in Connexion with Graphs

By

E. W. Dijkstra

We consider n points (nodes), some or all pairs of which are connected by a branch; the length of each branch is given. We restrict ourselves to the case where at least one path exists between any two nodes. We now consider two problems.

Problem 1. Construct the tree of minimum total length between the n nodes. (A tree is a graph with one and only one path between every two nodes.)

In the course of the construction that we present here, the branches are subdivided into three sets:

I. the branches definitely assigned to the tree under construction (they will form a subtree);

II. the branches from which the next branch to be added to set I, will be selected;

III. the remaining branches (rejected or not yet considered).

The nodes are subdivided into two sets:

A. the nodes connected by the branches of set I,

B. the remaining nodes (one and only one branch of set II will lead to each of these nodes).

We start the construction by choosing an arbitrary node as the only member of set A, and by placing all branches that end in this node in set II. To start with, set I is empty. From then onwards we perform the following two steps repeatedly.

Step 1. The shortest branch of set II is removed from this set and added to set I. As a result one node is transferred from set B to set A.

Step 2. Consider the branches leading from the node, that has just been transferred to set A, to the nodes that are still in set B. If the branch under consideration is longer than the corresponding branch in set II, it is rejected; if it is shorter, it replaces the corresponding branch in set II, and the latter is rejected.

We then return to step 1 and repeat the process until sets II and B are empty. The branches in set I form the tree required.

The solution given here is to be preferred to the solution given by J. B. Kruskal [1] and those given by H. Loberman and A. Weinberger [2]. In their solutions all the — possibly $\frac{1}{2}n(n-1)$ — branches are first of all sorted according to length. Even if the length of the branches is a computable function of the node coordinates, their methods demand that data for all branches are stored simultaneously. Our method only requires the simultaneous storing of

the data for at most n branches, viz. the branches in sets I and II and the branch under consideration in step 2.

Problem 2. Find the path of minimum total length between two given nodes P and Q.

We use the fact that, if R is a node on the minimal path from P to Q, knowledge of the latter implies the knowledge of the minimal path from P to R. In the solution presented, the minimal paths from P to the other nodes are constructed in order of increasing length until Q is reached.

In the course of the solution the nodes are subdivided into three sets:

A. the nodes for which the path of minimum length from P is known; nodes will be added to this set in order of increasing minimum path length from node P;

B. the nodes from which the next node to be added to set A will be selected; this set comprises all those nodes that are connected to at least one node of set A but do not yet belong to A themselves;

C. the remaining nodes.

The branches are also subdivided into three sets:

I. the branches occurring in the minimal paths from node P to the nodes in set A;

II. the branches from which the next branch to be placed in set I will be selected; one and only one branch of this set will lead to each node in set B;

III. the remaining branches (rejected or not yet considered).

To start with, all nodes are in set C and all branches are in set III. We now transfer node P to set A and from then onwards repeatedly perform the following steps.

Step 1. Consider all branches r connecting the node just transferred to set A with nodes R in sets B or C. If node R belongs to set B, we investigate whether the use of branch r gives rise to a shorter path from P to R than the known path that uses the corresponding branch in set II. If this is not so, branch r is rejected; if, however, use of branch r results in a shorter connexion between P and R than hitherto obtained, it replaces the corresponding branch in set II and the latter is rejected. If the node R belongs to set C, it is added to set B and branch r is added to set II.

Step 2. Every node in set B can be connected to node P in only one way if we restrict ourselves to branches from set I and one from set II. In this sense each node in set B has a distance from node P; the node with minimum distance from P is transferred from set B to set A, and the corresponding branch is transferred from set II to set I. We then return to step 1 and repeat the process until node Q is transferred to set A. Then the solution has been found.

Remark 1. The above process can also be applied in the case where the length of a branch depends on the direction in which it is traversed.

Remark 2. For each branch in sets I and II it is advisable to record its two nodes (in order of increasing distance from P), and the distance between P and that node of the branch that is furthest from P. For the branches of set I this

is the actual minimum distance, for the branches of set II it is only the minimum thus far obtained.

The solution given above is to be preferred to the solution by L. R. Ford [3] as described by C. Berge [4], for, irrespective of the number of branches, we need not store the data for all branches simultaneously but only those for the branches in sets I and II, and this number is always less than n. Furthermore, the amount of work to be done seems to be considerably less.

References

[1] Kruskal jr., J. B.: On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem. Proc. Amer. Math. Soc. 7, 48—50 (1956).
[2] Loberman, H., and A. Weinberger: Formal Procedures for Connecting Terminals with a Minimum Total Wire Length. J. Ass. Comp. Mach. 4, 428—437 (1957).
[3] Ford, L. R.: Network flow theory. Rand Corp. Paper, P-923, 1956.
[4] Berge, C.: Théorie des graphes et ses applications, pp. 68—69. Paris: Dunod 1958.

Mathematisch Centrum
2e Boerhaavestraat 49
Amsterdam-O

(Received June 11, 1959)

Shortest path algorithm

- The paper is just over 2 pages long
- The first page is a rediscovery of Prim's algorithm
  - Minimum cost spanning tree in a graph
- The second page is the shortest path algorithm

Programming languages

- Involved in the design of Algol 60
- Laid down the syntax of most modern programming languages
  - Code blocks, begin ... end
  - Nested function definitions, lexical scope
- Backus-Naur form grammar

Algol 60

- Dijkstra "sneaked" recursion into the language
  - "Any other occurrence of the procedure identifier within the procedure body denotes activation of the procedure."
- Conservative committee members "tricked" into accepting this modification

Algol 60 ...

- Dijkstra's name does not appear in the final list of authors of the report
  - Disagreement with the "majority opinion"
- Developed the first Algol compiler in Amsterdam in 1960 with Jaap Zonneveld

Algol 68

- Designed by IFIP Working Group 2.1
- Abandoned a simpler design proposed by Niklaus Wirth for a more baroque set of features
  - Wirth's proposal came out as Pascal
- Dijkstra wrote a scathing open letter criticising the committee

To the EDITOR ALGOL 68
Mathematisch Centrum
2de Boerhaavestraat 49
AMSTERDAM(O)
The Netherlands

EWD230 - 0

[Copies to members of IFIP.WG.2.1]

Department of Mathematics
Technological University
Postbox 513
EINDHOVEN
The Netherlands

{Reader's reaction: "there are writings which are lovable although ungrammatical, and there are other writings which are extremely grammatical, but are disgusting. This is something that I cannot explain to superficial persons."

from the Epigram of Chang Ch'ao "On Flowers and Women" as quoted by Lin Yutang in "The Importance of Living".

Author's defence: "My tragic tale I won't prolong
sing rickety, tickety tin,
my tragic tale I won't prolong
and if you did not enjoy my song
you've yourselves to blame if it's too long:
you should never have let me begin....

from "The Irish ballad" by Tom Lehrer}

On Algol 68 ...

Reader's reaction:
"there are writings which are lovable although ungrammatical, and there are other writings which are extremely grammatical, but are disgusting. This is something that I cannot explain to superficial persons."

Author's defence:
"My tragic tale I won't prolong, sing rickety, tickety tin,
my tragic tale I won't prolong
and if you did not enjoy my song
you've yourselves to blame if it's too long:
you should never have let me begin....

On Algol 68 ...

- Size and complexity of the defining apparatus you needed terrify me.
- Being well-acquainted with your ingenuity ... I feel inclined to put the blame on the language you tried to define.
- WG.2.1 should return to its proper subject matter, viz. programming languages.

The birth of software engineering?

"The true problem, as [Dijkstra] now saw it, was the reliable creation of programs to perform specified tasks, rather than their expression in some language. I doubt if any programming language, as the term was (and still is) understood, would have satisfied him."

The Humble Programmer

- Dijkstra's Turing Award lecture 1972
- "... as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming had become an equally gigantic problem."
- "Is it a wonder that we found ourselves in a software crisis?"

Structured programming

- Advocated block structured constructs for writing manageable code
  - Build up assertion for the program from the constituent blocks
- Go To Statement Considered Harmful
  - Letter to CACM 1968
  - "... an invitation to make a mess of one's program."

Deriving correct programs

- "Testing shows the presence, not the absence of bugs"
- Weakest precondition
  - R is a desired property after statement S
  - wp(S,R) is the weakest pre-condition from which the execution of S guarantees that R holds
- Programs as predicate transformers

Guarded commands

Avoiding the asymmetry of if-then-else

- ```
  if   x >= y → m := x
    [] y >= x → m := y
  fi
  ```

- ```
  q1, q2, q3, q4 := Q1, Q2, Q3, Q4
  do   q1 > q2 → q1,q2 := q2,q1
    [] q2 > q3 → q2,q3 := q3,q2
    [] q3 > q4 → q3,q4 := q4,q3
  od
  ```

Concurrent programming

- Cooperating Sequential Processes, 1968
  - Consolidated earlier work
- First to identify the mutual exclusion problem
- Identified the problem of deadlock ("deadly embrace")
- Proposed semaphores (P,V) for synchronisation
- Invented the Dining Philosopher's Problem

Concurrent programming

- Invented self-stabilising distributed algorithms to deal with fault tolerance
- First concurrent garbage collection algorithm
- Edsger W. Dijkstra Prize in Distributed Computing : awarded alternately at PODC and DISC

Writing style

- Always aimed for elegance and simplicity
- Avoided complicated notation, but also pictures!
- Cavalier approach to references
- "For the absence of a bibliography, I offer neither explanation nor apology"

A Discipline of Programming

Personality

- Often perceived as rude and arrogant
- "In Austin, he said 'Thank God' in reaction to a comment 'I am losing my voice' uttered by a renowned computer scientist from the MIT toward the end of her lecture. (A letter with apologies from the department chairman promptly followed.)"
- "You probably know that arrogance, in computer science, is measured in nanodijkstras."

Alan Kay, keynote speech at OOPSLA 1997

Extremely quotable

- In the good old days physicists repeated each other's experiments, just to be sure. Today they stick to FORTRAN, so that they can share each other's programs, bugs included.
- The question of whether Machines Can Think... is about as relevant as the question of whether Submarines Can Swim.

World's first blogger?

- EWD manuscripts written over 4 decades
- Technical notes, trip reports, observations, commentaries
- Earlier ones typewritten, later ones handwritten

Guarded commands, non-determinacy and a calculus for the derivation of programs.

by Edsger W.Dijkstra *)

*) Author's address: BURROUGHS
Plataanstraat 5
NUENEN - 4565
The Netherlands.

Abstract. So-called "guarded commands" are introduced as a building block for alternative and repetitive constructs that allow non-deterministic program components for which at least the activity evoked, but possibly even the final state, is not necessarily uniquely determined by the initial state. For the formal derivation of programs expressed in terms of these constructs, a calculus will be shown.

The threats to computing science

Let me describe to you by way of introduction the observation that induced me to choose "The threats to computing science" as my topic for this keynote address. The observation is simply that, when browsing through a computing science journal or conference proceedings, one usually sees at a first glance the geographical origin of the work described.

Now, in any area we have always had national differences, but their obviousness in computing becomes surprising as soon as we remember all the homogenizing forces at work during the few decades in which the topic emerged, and allow me to mention a few of them, just to underline the strength of this argument.

The Dijkstra font

- This presentation!
- Created by Luca Cardelli, now at Microsoft Research, Cambridge, UK

A Parable

- A long time ago there was a railway company
- Passenger coaches had toilets at one end
- Board of Directors decide to cut costs
- Introduce 50% coaches without toilets

Uproar!

- Instructions not communicated to yard
- Some trains have no coaches with toilets!

First fix

- Add a marker to each coach with toilet
- Instruct yard to compose trains with half the coaches marked
- Complicates the process of shunting coaches to make up a train, but manageable

Dissatisfaction

- Coaches with toilets bunched at one end
- Passengers at other end angry

Second fix

- Ensure that coaches with toilets alternate with those without
- More work for the yard
- Grumbling ...

More unhappiness

- Coaches with toilets could be oriented in opposite directions
- Toilets may be almost three coaches apart
- Parents of small children not amused!

Third fix

- Ensure that all coaches with toilets point the same way
- Coaches with toilets now have a direction
- Lots of coach rotation on turntables
- Yard workers not happy at all

Still some complaints

- Where is the toilet?
- Passengers in non-toilet coaches may walk two coach lengths
- Add an arrow to nearest toilet
- Now non-toilet coaches are also oriented!

Chaos

- Yard workers in a tizzy about ensuring trains are composed correctly
- Systems overloaded
- Schedules out of whack

Enlightenment

- Combine coaches with/without toilets in pairs, with toilet in the middle
  - Pairs are now non-oriented, identical units
  - Some compromises, but life goes on again!

According to Dijkstra

- Anonymous hero who found the solution deserves to be called the world's first competent programmer
- Programmers are delighted by this story
- Managers get more and more annoyed as the story progresses
- Mathematicians don't see the point!

Thank you

References

- "Edsger Wybe Dijkstra (1930-2002): A Portrait of a Genius", Krzysztof R. Apt, Formal Aspects of Computing, Volume 14, Issue 2, pp. 92-98 (2002)
- Dijkstra quotes
  https://en.wikiquote.org/wiki/Edsger_W._Dijkstra
- E. W. Dijkstra Archive, the manuscripts of Edsger W. Dijkstra, 1930-2002
  http://www.cs.utexas.edu/~EWD/