Ways of Computing

Jaikumar Radhakrishnan

ICTS
International Centre for Theoretical Sciences
Tata Institute of Fundamental Research

26 Feb 2023

The plan: three examples

- Computing shortest paths in networks
- Matching nuts and bolts
- A zero-knowledge proof involving *Sudoku*

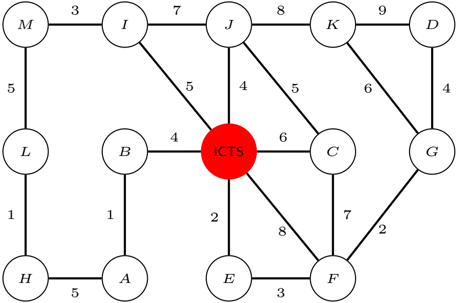
The problem

Nodes: They represent locations.

Links: They represent paths between locations.

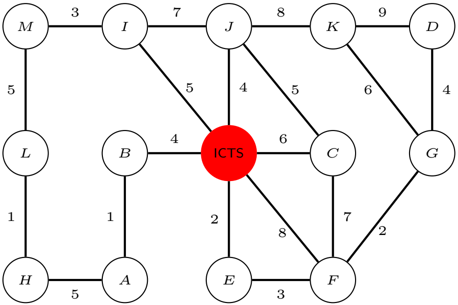
Costs: For each link we have a postive number, which represents the cost of using the link, e.g., its length, or the time it takes to travese it.

Goal: Find the best route from ICTS to all the other locations.



Why not try all possibilities?

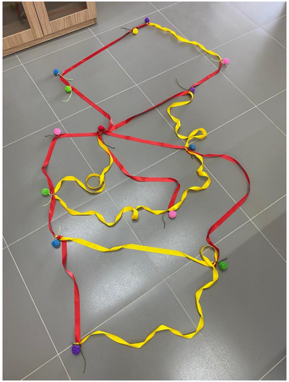
- Our network has 14 nodes and 19 links.
- There are at least 25 different routes from ICTS to node *D*.
- In general, even for moderately sized networks the number of possible paths from source to destination is enormous. But many of them can be systematically eliminated. How?



A physics experiment to find the best route

Balls: They correspond to nodes in our network. Place the balls on the floor.

Strings: They correspond to links in our network. Connect the balls with a string that is as long as the cost of the link.



Are you ready?

Step 1: Place everything on the floor in a heap.

Step 2: Lift the ball representing ICTS. (Let ICTS rise!)

Step 3: After the ball representing a location has risen, measure its distance from the ball representing ICTS.

- Did nature examine all possibilities and come up with the path?
- Could we have predicted the outcome with pen and paper?
- Then, how long will this computation take for a network on *n* nodes and *m* links?

Are you ready?

Step 1: Place everything on the floor in a heap.

Step 2: Lift the ball representing ICTS. (Let ICTS rise!)

Step 3: After the ball representing a location has risen, measure its distance from the ball representing ICTS.

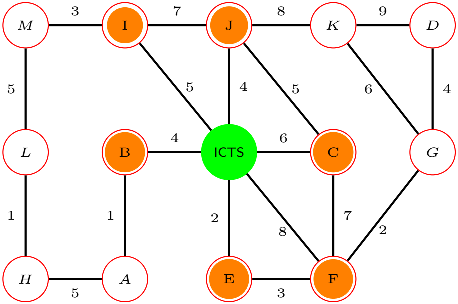
- Did nature examine all possibilities and come up with the path?
- Could we have predicted the outcome with pen and paper?
- Then, how long will this computation take for a network on *n* nodes and *m* links?

Dijkstra's method

Edsger Dijkstra (1930-2002): A note on two problems in connexion with graphs. In Numerische Mathematik, 1 (1959), S. 269–271.

At each step ...

- Initially, ICTS is green, all other nodes are red. Give the neighbours of ICTS a tentative cost equal to the length of the link from ICTS and mark them orange
- Pick the orange node with minimum cost.
- Colour it green. Its tentative cost is now final. (It has risen!)
- Update the information on its neighbouring nodes.

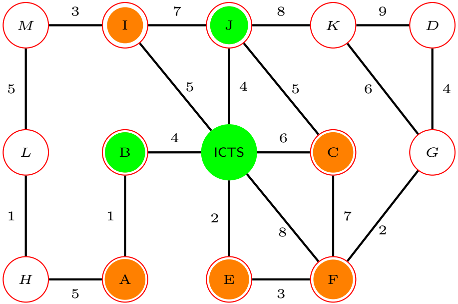


Dijkstra's method

Edsger Dijkstra (1930-2002): A note on two problems in connexion with graphs. In Numerische Mathematik, 1 (1959), S. 269–271.

At each step ...

- Initially, ICTS is green, all other nodes are red. Give the neighbours of ICTS a tentative cost equal to the length of the link from ICTS and mark them orange.
- Pick the orange node with minimum cost.
- Colour it green. Its tentative cost is now final. (It has risen!)
- Update the information on its neighbouring nodes.



Nuts and bolts (or keys and locks)

The problem

- We are given a large number of nuts and bolts, of different sizes.
- Each nut matches a unique bolt.
- When we try to match nut with a bolt, we know if the nut is too big or too small or just right.
- How do we match them up? Must we try every nut against every bolt?

If only we could compare nuts with nuts and

. . . bolts with bolts

- Sort the nuts in the increasing order of their sizes.
- Sort the bolts similarly.
- Match the largest nut with the largest bolt, the second largest nut with the second largest bolt, . . .
- How long does it take?

If only we could compare nuts with nuts and

. . . bolts with bolts

- Sort the nuts in the increasing order of their sizes.
- Sort the bolts similarly.
- Match the largest nut with the largest bolt, the second largest nut with the second largest bolt, . . .
- How long does it take?

. . . but we can't!

Two strategies

Strategy I: Ignore the sizes

- Pick a random nut.
- Try it against every bolt.
- Put the matching pair aside and repeat with the rest.
- To match n pairs, it will take about $n^2/4$ comparisons on average.

Strategy II: Divide and conquer

- Pick a random nut.
- Partition the bolts into too small, just right and too big.
- Using the matching bolt, partition the bolts, similarly.
- Put the matching pair aside and solve the two subproblems independently.
- To match n pairs, it will take about $4n \ln n$ comparisons on average.

Two strategies

Strategy I: Ignore the sizes

- Pick a random nut.
- Try it against every bolt.
- Put the matching pair aside and repeat with the rest.
- To match n pairs, it will take about $n^2/4$ comparisons on average.

Strategy II: Divide and conquer

- Pick a random nut.
- Partition the bolts into too small, just right and too big.
- Using the matching bolt, partition the bolts, similarly.
- Put the matching pair aside and solve the two subproblems independently.
- To match n pairs, it will take about $4n \ln n$ comparisons on average.

Sudoku and zero-knowledge proofs

- You are given Sudoku puzzle. You suspect that perhaps the problem is unsolvable.
- I have a solution. I want to convince you that the problem is solvable without revealing the solution.
- Can it be done?
  Yes! With randomness. . . . and paper and scissors.

	9		1		5		2	
6			3		2			8
	7			6			3	
		6				1		
9	1						4	5
		7				2		
	8			3			9	
7			4		8			2
	6		7		9		1	

LaTeX source: Roberto Bonvallet

(due to Gradwohl, Naor, Pinkas, Rothblum, see `https://www.wisdom.weizmann.ac.il/~naor/PAPERS/sudoku_abs.html`)

Summary: the many ways of computing

- Dijkstra's algorithm as a physics experiment.
- A randomized divide and conquer solution for the Nuts and Bolts problem. Deterministic methods (theoretically) matching the randomized solution are known, but are complicated (Bradford, Komlós, Ma, Szeméredi, 1995).
- Zero-knowledge proof for Sudoku. Randomness was key!

The three types of constraints

- Each row must have all nine digits.
- Each column must have all nine digits.
- Partition the rows and columns into nine 3×3 blocks. Each block must have all nine digits.

Thank you!

Thank you!

Thanks to Ramprasad Saptharishi for the comparison software, and for suggestions for this talk.