The German Collegiate Programming Contest 2016

The GCPC 2016 Jury

04.06.2016

Judges' Solutions

Problem	Min LOC	Max LOC
Dwarves	36	113
Correcting Cheeseburgers	43	145
Knapsack in a Globalized World	19	111
Matrix Cypher	22	340
Model Railroad	47	127
One-Way Roads	51	557
Formula	17	67
Celestial Map	27	148
Common Knowledge	1	49
Selling CPUs	13	64
Routing	40	120
Maze	34	91
total	350	1932

G: Formula - Sample Solution

Easiest problem in the set.

Problem

Given a triangle Δabc and a number r.
How much differs the incircle radius of Δabc from r_m?

Solution

- You are given formulas to compute
  - the incircle radius, given the area
  - the area, given the side lengths
- Rearranging the formulas leads to

$$r_{\Delta abc} = \frac{\sqrt{4a^2b^2 - (a^2 + b^2 - c^2)^2}}{2(a + b + c)}$$

- The answer is $\frac{r_{\Delta abc} - r_m}{r_m}$

I: Common Knowledge - Sample Solution

Problem

For two segment displays of length n where on one both players see the top half and on the other one player sees the top and one the bottom half, how many numbers do they both know?

Solution

- Digits recognizable by seeing the top half: 0,1,4,7.
- Digits recognizably by seeing the bottom half: 0,2,4.
- If both see the top half, there are four possible digits, thus 4^n numbers
- Otherwise there are only two possible digits (0 and 4), thus 2^n numbers.
- In total, there are $4^n \cdot 2^n = 8^n$ numbers which is the solution.

A: Dwarves - Sample Solution

Problem

Given various statements about the relative heights of the dwarves, decide whether there is a contradiction in their statements.

Solution

- Read input as directed graph with an edge from the smaller to the larger dwarf.
- Check if the graph is acyclic (e.g. with DFS).

D: Matrix Cypher - Sample Solution

Problem

Decode a message (represented as bitstring) that has been encoded by repeatedly multiplying different matrixes onto the identity matrix depending on zeroes and ones.

Solution

- Note that the matrix for the bit 0 corresponds to adding first row to the second. If the bit is 1, we add the second row to the first.
- For decoding simply check which row is greater and undo the action by subtraction. This gives you the last bit of the message. Repeat this, until we have the identitiy matrix.

J: Selling CPUs - Sample Solution

Problem

You have c identical objects and there are m ordered merchants. Each merchant i each has his own price p^i_j for each amount j of objects you can sell him.
How much money can you make by selling your objects to the merchants?

Insights

- If $m < c$ it might not be optimal to sell all objects
- To determine how much to sell to merchant j it is only relevant how much you sold to the merchants $i < j$ **in total**, not how much you sold to the individual merchant

J: Selling CPUs - Sample Solution

Solution

- Use Dynamic Programming over (#merchants,#CPUs sold)

$$max\_price(m, c) = \begin{cases} 0 & \text{, if } m = 0 \\ \min(max\_price(m-1, c), \\ \qquad \min_{k=1}^{c} max\_price(m-1, c-k) + p_k^i) & \text{, else} \end{cases}$$

- Setting all $max\_price(0, c) = 0$, means that we don't have to sell all objects
- Runtime $\mathcal{O}(mc^2)$

J: Selling CPUs - Sample Solution

Solution

- Use Dynamic Programming over (#merchants,#CPUs sold)

$$max\_price(m,c) = \begin{cases} 0 & \text{, if } m = 0 \\ \min(max\_price(m-1,c), \\ \qquad \min_{k=1}^{c} max\_price(m-1,c-k) + p_k^i) & \text{, else} \end{cases}$$

  - Setting all $max\_price(0,c) = 0$, means that we don't have to sell all objects
  - Runtime $\mathcal{O}(mc^2)$

J: Selling CPUs - Sample Solution

Solution

- Use Dynamic Programming over (#merchants,#CPUs sold)

$$max\_price(m, c) = \begin{cases} 0 & , \text{ if } m = 0 \\ \min(max\_price(m-1, c), & \\ \qquad \min_{k=1}^{c} max\_price(m-1, c-k) + p_k^i) & , \text{ else} \end{cases}$$

  - Setting all $max\_price(0, c) = 0$, means that we don't have to sell all objects
  - Runtime $\mathcal{O}(mc^2)$

E: Model Railroad - Sample Solution

Problem

Given a railroad network with already existing connections and possible connections, is it possible to destroy some connections and build others of at most the same total length such that the new network is connected?

Solution

- Sum up the length of all existing edges.
- Run the MST algorithm of your choice.
- Output "possible" if the weight of the MST is at most the sum of the existing edges.

L: Maze - Sample Solution

Problem

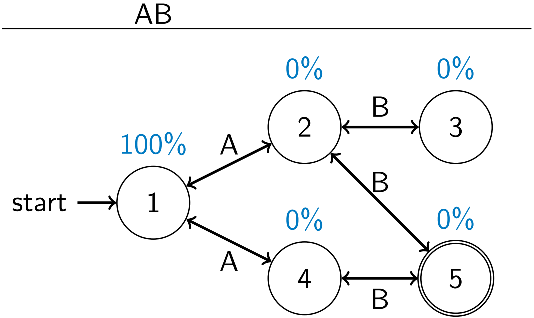
Given a word w and a nondeterministic finite automaton A, compute the chance that any prefix of w is accepted by A.

Solution

- Simulation/Dynamic Programming
- Store an array $P[n]$ with probabilities for each node, starting with 100% at the start node
- For every letter l in w:
  - $P'[n]$ is the sum over the probability of all incoming edges.
  - Probability to take edge $n_0 \xrightarrow{l} n$ is $P[n_0] * \frac{1}{|out(n_0, l)|}$

L: Maze - Sample Solution

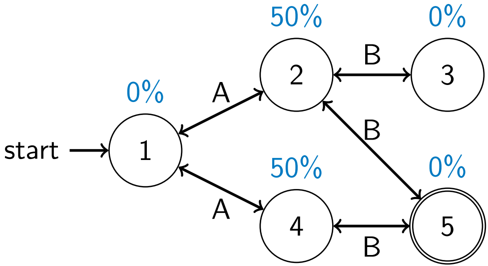
Example



L: Maze - Sample Solution

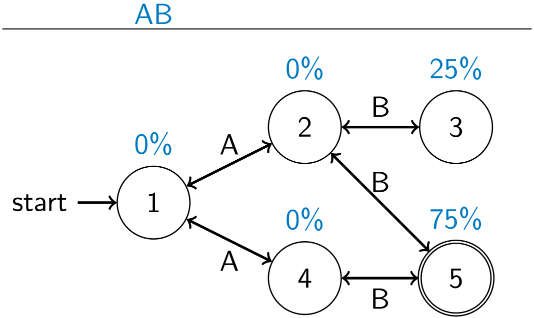
Example

AB



L: Maze - Sample Solution

Example



H: Celestial Map - Sample Solution

Problem

You are given multiple stars' location and trajectory, furthermore a plane and a distance d. For every star, decide whether it was in the plane and had distance d to you when it sent the message which you recieved just now.

Insight

- Compute normal of p_1 and p_2 to represent the plane (cross product).

H: Celestial Map - Sample Solution

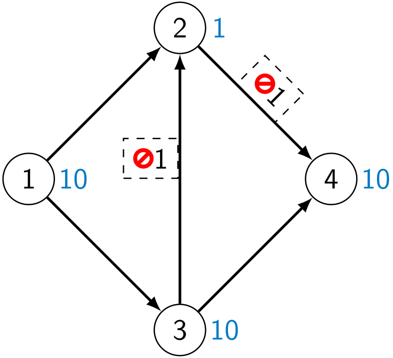
Solution

- To test whether a star is viable:
  - Compute intersection of plane and star's trajectory.
  - Distance t from Bob to intersection = time it took for the message to reach Bob.
  - intersection + $t\cdot$ trajectory = star's current position iff star is viable.
- or (without computing intersections)
  - If the star had distance d to Bob, it took d lightyears for the message to arrive.
  - Take stars current position $(s_x\ s_y\ s_z)$ and remove $d \cdot (t_x\ t_y\ t_z)$.
  - Check if the point we got is in plane (using the normal vector) and has the correct distance to Bob. If this is the case the star is viable.

K: Routing - Sample Solution

Problem

Find a shortest path, but whether an edge can be used depends on the last edge that was used.

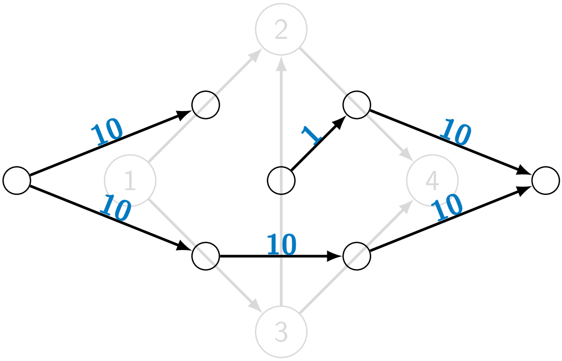


K: Routing - Sample Solution

Insights

- Consider the dual graph instead:
- Edges become nodes.
- Nodes become edges connecting edges if they can be used together.
- The weights of the edges are the processing times of the corresponding server.
- Add artificial nodes for the source and the target.

K: Routing - Sample Solution



K: Routing - Sample Solution

Solution

- Search for a shortest path on the dual graph instead.
- Use for instance Dijkstra's Algorithm.
- The dual graph has at most n^2 vertices and n^3 edges.
- Running time: $\mathcal{O}(n^2 \log n^2 + n^3) = \mathcal{O}(n^3)$
- Different idea: Work on the original graph, but use as Dijkstra state not only the node and the distance, but also the last edge you used.

F: One-Way Roads - Sample Solution

Problem

Given an undirected graph, find a number d such that there is an orientation of the edges where every node has in-degree of at most d.

Solution

- For a given d we can decide whether there exists an orientation that fulfils the constraint using maximum flow (see next slide).
- Use binary search to find the minimal d.

F: One-Way Roads - Sample Solution

Find an orientation

- First solution:
  - Start with an arbitrary orientation $\vec{G}$.
  - Nodes with indegree greater than d have to flip edges.
  - Nodes with smaller indegree may increase their indegree.
  - Add a source with edges to all nodes v, capacity $\max(0, indeg_{\vec{G}}(v) - d')$.
  - Add sink with edges to all nodes, capacity $\max(0, d' - indeg_{\vec{G}}(v))$.
  - Add edges between the nodes in reverse direction of $\vec{G}$.
  - A maximum flow now tells you whether you should flip an edge (if it is used by the flow) and whether there was a solution (if all source edges are used to full capacity).

F: One-Way Roads - Sample Solution

Find an orientation

- Second solution:
  - Compute a matching between roads and the cities they connect.
  - The graph has one node per city and one per road, as well as source and sink.
  - The source is connected to all roads with capacity 1, each road to its endpoints with capacity 1.
  - The cities are connected to the sink with capacity d.
  - A maximum flow now tells you whether there is a matching of roads to cities (if the flow is equal to the number of roads).

B: Correcting Cheeseburgers - Sample Solution

Problem

Given a number of up to 10 digits find the minimum number of *bit-shuffles* to sort the digits in ascending order.

Idea

- Construct the graph G of possible permutations.
- There is a directed edge between permutations a and b iif there is some shuffle such that bit-shuffle$(a) = b$.

B: Correcting Cheeseburgers - Sample Solution

Naive Solution

- BFS on graph to find minimum number of steps from starting number s to sorted number t.
- Graph G has a manageable amount of nodes ($|V| = N!$).
- Each node has about N^3 edges ($|E| \approx N! * N^3$).

$\Rightarrow$ BFS results in TLE.

B: Correcting Cheeseburgers - Sample Solution

Insights

- The maximum number of steps required is at most 6. (Can be confirmed by naive exploration in a few minutes.)
- BFS bounded by a maximum depth of 3 is fast.
- The *bit-shuffle* can be reversed. We can search backwards.

Solution - Bidirectional Search

- 2-depth BFS from s and 3-depth BFS with the reversed *bit-shuffle* from t.
- Check for overlaps to get the minimum steps required.
- No overlap $\Rightarrow$ result must be greater than 5 and combined with our previous insight it must be 6.

C: Knapsack in a Globalized World - Sample Solution

Problem

Good old KNAPSACK with a twist: The size of the knapsack is too large to fit into the memory, but every of n items can be put multiple times into the knapsack.

Idea

- Do calculation modulo the size of an arbitrary item, let's say G_1.
- The most crucial insight:

$$\exists a_i \in \mathbb{N}_{\geq 0}: \; K = \sum_{1 \leq i \leq n} a_i \cdot G_i \Leftrightarrow$$

$$\exists a_i \in \mathbb{N}_{\geq 0}: \; K \mod G_1 = \sum_{2 \leq i \leq n} a_i \cdot G_i \text{ and } \sum_{2 \leq i \leq n} a_i \cdot G_i \leq K$$

C: Knapsack in a Globalized World - Sample Solution

Solution - shortest path

- Every modulo class is a node in the graph, there are G_1 nodes.
- There is an edge from node i to node j, iff there is an item k with $i + G_k = j \mod G_1$. There are at most $n \cdot G_1$ edges.
- The shortest path from 0 to $K \mod G_1$ must not be longer than K.
- Dijkstra is fast enough - $O\left(n \cdot G_1 \log(n \cdot G_1)\right)$.
- We also accepted any other $O\left(G_1^2 \cdot n\right)$ shortest path algorithm.

C: Knapsack in a Globalized World - Sample Solution

Solution - number theory

- Let G_{max} be the largest item.
- Every number K greater than $G_{max} \cdot G_{max}$ is reachable iff it is divisible by the GCD of $\{G_1, \ldots, G_n\}$.
- Solve KNAPSACK normally for $K <= G_{max} \cdot G_{max}$, do the GCD calculations for larger numbers - results in $O\left(G_{max}^2 \cdot n\right)$.

Nota bene

- This problem is also known as Money Changing Problem (MCP).
- It's NP-complete.
- A pseudo-polynomial $O\left(G_1 \cdot n\right)$ solution is known.

Thanks

We thank all organizers, problem setters, jury members, contest site organizers and other helpers for their work.

Contest Director

Stefan Toman, Technische Universität München

Main Organization

Moritz Fuchs, Technische Universität München
Philipp Hoffmann, Technische Universität München
Christian Müller, Technische Universität München
Chris Pinkau, Technische Universität München

Thanks

Jury

Markus Blumenstock, Johannes Gutenberg-Universität Mainz
Egor Dranischnikow, Johannes Gutenberg-Universität Mainz
Moritz Fuchs, Technische Universität München
Philipp Hoffmann, Technische Universität München
Christian Müller, Technische Universität München
Martin Schuster, Universität zu Lübeck
Ben Strasser, Karlsruher Institut für Technologie
Martin Tillmann, Karlsruher Institut für Technologie
Stefan Toman, Technische Universität München

Thanks

Proofreaders and Testers

Michael Baer, Miriam Polzer, Tobias Werth, Paul Wild, Thorsten Wißmann

Helpers and Coaches

Nourhene Bziouech, Doina Logofatu, Marinus Gottschau, Stefan Jaax, Markus Mock, Alexander van Renen, Melanie Strauss

Thanks

Contest Site Organizers

Heiko Röglin (Bonn) Tobias Brinkjost (Dortmund) Michael Baer (Erlangen), Daniela Novac(Erlangen), Dominik Paulus Erlangen), Miriam Polzer (Erlangen), Tobias Polzer (Erlangen), Paul Wild (Erlangen), Thorsten Wißmann (Erlangen), Bakhodir Ashirmatov (Göttingen), Ben Strasser (Karlsruhe), Martin Tillmann (Karlsruhe), Max Bannach (Lübeck), Tim Kunold (Lübeck), Maciej Liskiewicz (Lübeck), Martin Schuster (Lübeck),Markus Blumenstock (Mainz), Egor Dranischnikow (Mainz), Jochen Saalfeld (Osnabrück), Julian Pätzold (Osnabrück), Tobias Friedrich (Potsdam), Christoph Kessler (Potsdam), Christian Baldus (Saarland), Julian Baldus (Saarland), Gregor Behnke (Ulm), Marianus Ifland (Würzburg)

Thanks