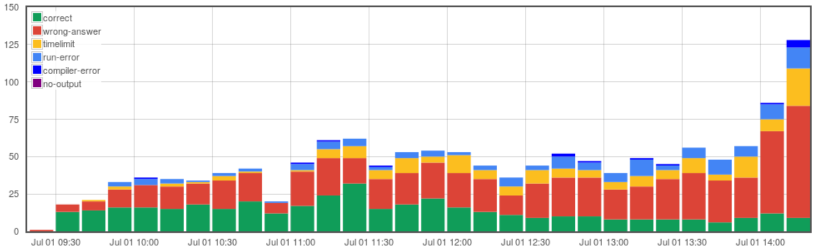
German Collegiate Programming Contest 2017

The GCPC 2017 Jury

01.07.2016

Statistics



150
125
100
75
50
25
0
correct
wrong-answer
timelimit
run-error
compiler-error
no-output
Jul 01 09:30
Jul 01 10:00
Jul 01 10:30
Jul 01 11:00
Jul 01 11:30
Jul 01 12:00
Jul 01 12:30
Jul 01 13:00
Jul 01 13:30
Jul 01 14:00

Statistics

Problem	Min LOC	Max LOC
Borders	48	337
Buildings	26	90
Joyride	46	84
Pants on Fire	30	97
Plug It In	51	206
Perpetuum Mobile	30	142
Water Testing	17	269
Ratatöskr	56	131
Überwatch	14	72
Word Clock	79	168
You are fired	21	91
total	418	1687

K: You Are Fired! - Sample Solution

Problem

Given a list of employees with their respective salary, decide if there is a way to save d dollars by firing at most k of the employees.

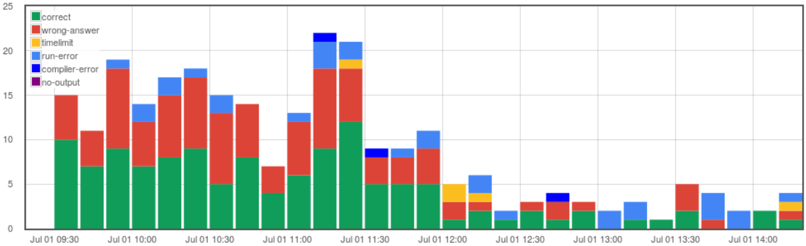
Solution

- Sort all employees by their salary.
- If the accumulated salary of the k best earning employees is smaller than d, print out "impossible".
- Otherwise: Fire employees in descending order of their salary. Stop if their accumulated weight is $>= d$.

K: You Are Fired! - Statistics

- Tried by 125 teams (97%), solved by 120 teams (93%)
- C++: Tried by 58 teams (45%), solved by 56 teams (43%), 110 submissions (51% correct)
- Java: Tried by 59 teams (46%), solved by 56 teams (43%), 131 submissions (43% correct)
- Python: Tried by 10 teams (8%), solved by 8 teams (6%), 20 submissions (40% correct)
- Fastest: 10 minutes, written by oachkatzlschwoaf
- Best runtime: 0% of the given time, written by 42 teams
- Shortest: 393 characters, written by FAU wie 🐦

K: You Are Fired! - Statistics



D: Pants on Fire - Sample Solution

Problem

You are given N statements of the form `A are worse than B` and one further such statement $S =$`X are worse than Y`.
Decide whether

- S logically follows
- $\neg S$ logically follows
- none of the two

Solution

- `are worse than` induces an ordering relation $\prec$
- Use Floyd-Warshall to compute the transitive hull of the N given statements, which is $\prec$

D: Pants on Fire - Sample Solution

Solution

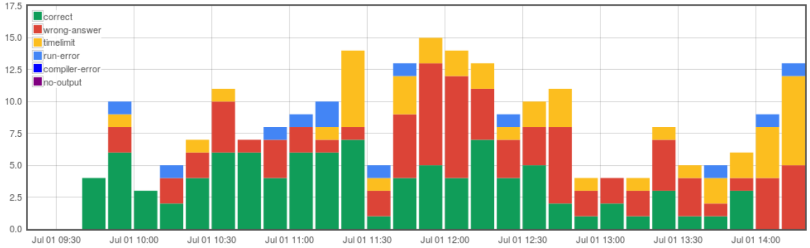
- Check whether the order S or its inverse (`Y are worse than X`, i.e. $\texttt{Y} \prec \texttt{X}$) are in $\prec$.
- If $S \in \prec$: `Fact`
- Else If $\overline{S} \in \prec$: `Alternative Fact`
- Else: `Pants on Fire`

- Using DFS was also possible

D: Pants on Fire - Statistics

- Tried by 116 teams (90%), solved by 98 teams (76%)
- C++: Tried by 54 teams (42%), solved by 52 teams (40%), 92 submissions (57% correct)
- Java: Tried by 53 teams (41%), solved by 41 teams (32%), 119 submissions (34% correct)
- Python: Tried by 9 teams (7%), solved by 5 teams (4%), 26 submissions (19% correct)
- Fastest: 20 minutes, written by goto considered_harmful;
- Best runtime: 0% of the given time, written by 34 teams
- Shortest: 554 characters, written by It compiles, submit it!

D: Pants on Fire - Statistics



I: Uberwatch - Sample Solution

Problem

A game of Überwatch is played over n time slices. The player can defeat all currently visible opponents. The attack must be charged for m time slices after every use and at the beginning of the game. What is the maximum number of opponents that can be defeated?

Observations

The greedy strategy of firing as fast as possible might skip a time slice needed for the optimal solution. $\rightarrow$ WA

Only two choices in every time slice:

- Fire
  Must not have fired for atleast m time slices. Score current number of opponents.
- Wait
  Score does not change.

I: Uberwatch - Sample Solution

Problem

A game of Überwatch is played over n time slices. The player can defeat all currently visible opponents. The attack must be charged for m time slices after every use and at the beginning of the game. What is the maximum number of opponents that can be defeated?

Solution

Solution for time slice t:

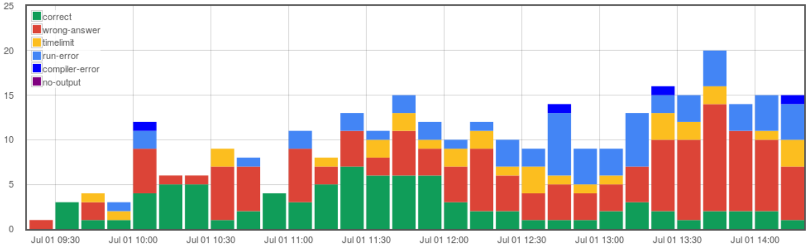
$$solution(t) = \max(\underbrace{opponents(t) + solution(t-m)}_{\text{Fire}}, \underbrace{solution(t-1)}_{\text{Wait}})$$

One dimensional dynamic programming to get $O(n)$ run time.
Special case the intial charging.

I: Uberwatch - Statistics

- Tried by 111 teams (86%), solved by 81 teams (63%)
- C++: Tried by 55 teams (43%), solved by 49 teams (38%), 143 submissions (34% correct)
- Java: Tried by 53 teams (41%), solved by 28 teams (22%), 153 submissions (18% correct)
- Python: Tried by 6 teams (5%), solved by 4 teams (3%), 13 submissions (31% correct)
- Fastest: 11 minutes, written by Hello KITty 😼
- Best runtime: 1% of the given time, written by 💥🔥💥 PanzerFAUst 💥🔥💥, FAU wie 🐦, 🐢🐉#Irgendwas🐉🐢, The Three Wizards, ,') DROP TABLE Teams; –, Hello KITty 😼, Lambda, Spezialexpertenkomitee, oachkatzlschwoaf, Team H
- Shortest: 230 characters, written by veni, vidi, pizzapanes cenavi

I: Uberwatch - Statistics



C: Joyride - Sample Solution

Problem

Given a graph $G = (V, E)$, per node a time t_v and a cost c_v, and a time t^* for all edges. You are traversing the graph starting at 1 and at every node you pass you have stay kt_v time and spend kc_v money with $k \geq 1$

What is the least amount of money you need to spend s.t. you arrive at 1 after X time?

Solution

The problem is a variant of Knapsack.

The time is the "weight", X is the size of the bucket, and c_v is the "gain" of an edge.

Extend standard Knapsack-DP to handle the graph strcuture.

C: Joyride - Sample Solution

Use DP over the current node v and the time t you can move freely from e.

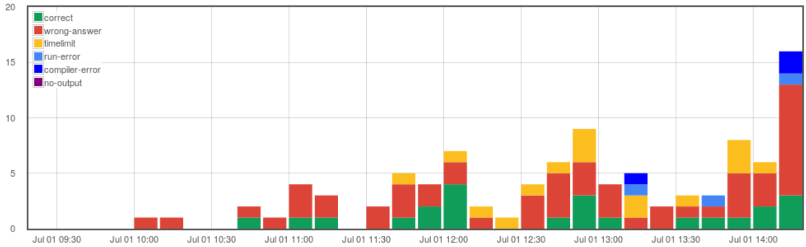
$$dp(v,t) = \begin{cases} \infty & t < 0 \\ c_v + min\Big(\{dp(v, t - t_v)\} \cup & \\ \quad \{dp(v', t - t^* - t_v) \mid (v', v) \in E\}\Big) & \text{if } t >= 0 \end{cases}$$

Additional special treatment for entering the park:
The answer is $dp(1, X - t_1) + c_v$

C: Joyride - Statistics

- Tried by 35 teams (27%), solved by 23 teams (18%)
- C++: Tried by 26 teams (20%), solved by 18 teams (14%), 69 submissions (26% correct)
- Java: Tried by 8 teams (6%), solved by 5 teams (4%), 30 submissions (17% correct)
- Python: Tried by 1 teams (1%), solved by 0 teams (0%), 1 submissions (0% correct)
- Fastest: 83 minutes, written by oachkatzlschwoaf
- Best runtime: 0% of the given time, written by PSpace-Orakel, ApfeLMUs, Hexaflexagons, Knoblauch, Ingwer und Thymian, goto considered_harmful;, perfect-zero-knowledge, vO)>, Lambda, <(Ov
- Shortest: 958 characters, written by 7sen

C: Joyride - Statistics



G: Water Testing - Sample Solution

Problem

Given a polygon $P = (p_1, \ldots, p_n)$ s.t. all p_i are on integer coordinates.
How many integer points are strictly inside P?

Solution

Use Pick's theorem.

$$A = I + \frac{R}{2} - 1$$

where

- A - Area of P
- I - Integer points strictly inside P
- R - Integer points in the border of P

G: Water Testing - Sample Solution

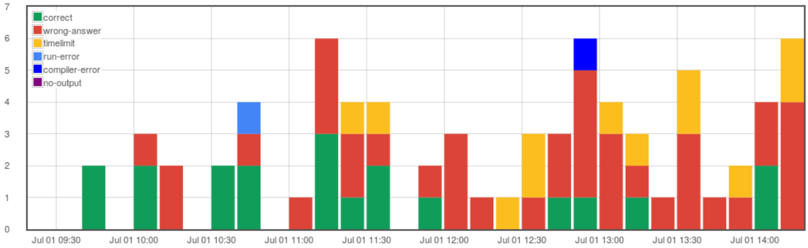
$$A = I + \frac{R}{2} - 1$$

- Compute A with any interger-safe method
- R can be computed using gcd
  - each edge of the polygon is defined by a $(\Delta_x, \Delta_y) = (|p_1^x - p_2^x|, |p_1^y - p_2^y|)$ pair
  - we have to determine how many $k \in \mathbb{N}^+$ there are s.t. $\frac{\Delta_x}{k}$ and $\frac{\Delta_y}{k}$ are integer.
  - but this is just $gcd(\Delta_x, \Delta_y)$

G: Water Testing - Statistics

- Tried by 33 teams (26%), solved by 19 teams (15%)
- C++: Tried by 23 teams (18%), solved by 18 teams (14%), 51 submissions (35% correct)
- Java: Tried by 10 teams (8%), solved by 1 teams (1%), 23 submissions (4% correct)
- Python: Tried by 0 teams (0%), solved by 0 teams (0%), 0 submissions (0% correct)
- Fastest: 22 minutes, written by 💥🔥💥 PanzerFAUst 💥🔥💥
- Best runtime: 0% of the given time, written by But wait, there's more!
- Shortest: 590 characters, written by Hello KITty 😼

G: Water Testing - Statistics



E: Perpetuum Mobile - Sample Solution

Problem

Given a directed graph with arc weights in $[0.0001, 5]$, decide whether the graph cointains a circle with total product > 1.

Solution

- Given a circle in the graph, let c_i be the arc weight of the ith arc in the circle. We know: $c_i > 0$ for all arcs i. Then

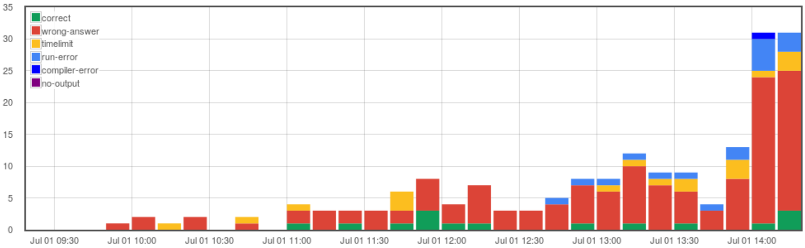
$$\Pi_i c_i > 1 \iff -\Sigma_i \log(c_i) < 0$$

- Convert arc weights c_i to $-\log(c_i)$ and use Bellman-Ford or Floyd-Warshall to decide whether a negative cycles exists in the transformed graph. This is precisely the case if there is a cycle with product > 1 in the original graph.

E: Perpetuum Mobile - Statistics

- Tried by 53 teams (41%), solved by 15 teams (12%)
- C++: Tried by 34 teams (26%), solved by 13 teams (10%), 108 submissions (12% correct)
- Java: Tried by 17 teams (13%), solved by 2 teams (2%), 68 submissions (3% correct)
- Python: Tried by 2 teams (2%), solved by 0 teams (0%), 7 submissions (0% correct)
- Fastest: 104 minutes, written by Hello KITty 😼
- Best runtime: 0% of the given time, written by FAU wie 🐦, Spezialexpertenkomitee, Hexaflexagons
- Shortest: 767 characters, written by oachkatzlschwoaf

E: Perpetuum Mobile - Statistics



F: Plug it in - Sample Solution

Problem

- Adam placed his electronics randomly into his room.
- Not every device can be plugged into every socket.
- How many devices can Adam power using 1 powerbar which triples 1 socket?

Solution

- Naive idea: Compute a maximum bipartite matching (possibly using a maximum flow algorithm) for each of the n possible locations of the powerbar
- This is, however, too time-consuming!

F: Plug it in - Sample Solution

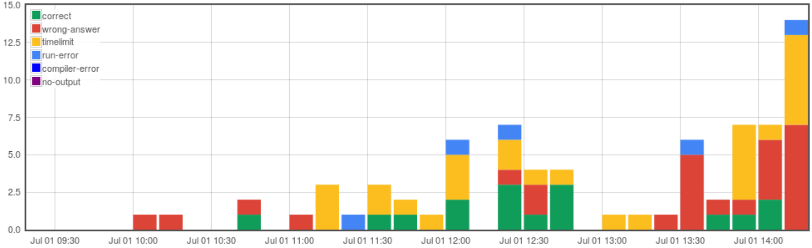
Solution

- Create a bipartite graph with sockets on the left, devices on the right and a connection between a socket and a device if the device can be plugged into the socket.
- Compute a maximum bipartite matching without the powerbar
- For each matched socket with more than 1 neighbor:
  - Triple the socket (i.e. add 2 unoccupied copies of the socket / raise the capacity in the flow network).
  - Compute the maximum bipartite matching / maximum flow in the resulting graph.
  - Use the bipartite matching of the original graph as a starting point for the matching of the new graph!
- Take the maximum across all computed matchings.

F: Plug it in - Statistics

- Tried by 29 teams (22%), solved by 16 teams (12%)
- C++: Tried by 21 teams (16%), solved by 14 teams (11%), 58 submissions (24% correct)
- Java: Tried by 7 teams (5%), solved by 2 teams (2%), 15 submissions (13% correct)
- Python: Tried by 1 teams (1%), solved by 0 teams (0%), 3 submissions (0% correct)
- Fastest: 88 minutes, written by Hello KITty 😼
- Best runtime: 1% of the given time, written by Lambda
- Shortest: 1450 characters, written by Knoblauch, Ingwer und Thymian

F: Plug it in - Statistics



B: Buildings - Sample Solution

Problem

How many possible houses are there (up to rotation) with m walls consisting of $n \times n$ bricks, each colored in one of c colors?

Solution

For each wall, there are $c^{(n \times n)}$ possible designs.
But how to deal with rotation?

B: Buildings - Sample Solution

Burnside's Lemma

The number of orbits *(distinct results)* for a group of operations *(possible rotations)* acting on a set X *(the set of walls)* is exactly the average number of elements *(wall designs)* fixed by each group element *(rotation by a fixed n)*.

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$$

B: Buildings - Sample Solution

Application

Consider all rotations rot_i rotating the house by i.

1. Since the house has m walls, all designs have to have periodicity m (i.e. repeat after m walls).
2. Rotation by i fixes all wall designs every i walls, i.e. it fixes all designs with periodicity i.
3. $\Rightarrow$ Rotation by i fixes a design iff the design has periodicity $gcd(i, m)$.
4. There are exactly $\{\text{number of wall designs}\}^{gcd(i,m)}$ possible designs with periodicity $gcd(i, m)$.

B: Buildings - Sample Solution

Problem

How many possible houses are there (up to rotation) with m walls consisting of $n \times n$ bricks, each colored in one of c colors?

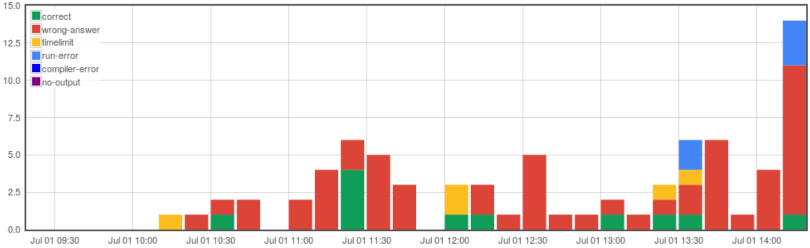
Solution

- Let $N = c^{(n \times n)}$ be the number of wall designs.
- There are exactly
$$\frac{1}{m} \sum_{0 \leq i < m} N^{gcd(i,m)}$$
possible house designs up to rotation.
- Compute this number modulo 1000000007 so it does not overflow `long`.

B: Buildings - Statistics

- Tried by 26 teams (20%), solved by 11 teams (9%)
- C++: Tried by 13 teams (10%), solved by 8 teams (6%), 33 submissions (24% correct)
- Java: Tried by 9 teams (7%), solved by 2 teams (2%), 31 submissions (6% correct)
- Python: Tried by 5 teams (4%), solved by 1 teams (1%), 16 submissions (6% correct)
- Fastest: 74 minutes, written by 💥🔥💥 PanzerFAUst 💥🔥💥
- Best runtime: 0% of the given time, written by 💥🔥💥 PanzerFAUst 💥🔥💥, Top University of Memes, vO)>, Lambda, oachkatzlschwoaf, ApfeLMUs, #define true !!! false
- Shortest: 201 characters, written by It compiles, submit it!

B: Buildings - Statistics



H: Ratatoeskr - Sample Solution

Problem

A tree of n nodes and starting positions (nodes) of a squirrel and two ravens are given. On a signal of Odin, one raven flies into the air and lands again. Meanwhile, the squirrel can travel in the tree, but may not pass over a node occupied by the other raven. Output the minimum number of signals after which the ravens are guaranteed to have captured the squirrel.

Possible Approaches

1. Dynamic programming on possible states
2. Force the squirrel into the shallowest subtree
3. Find the 'highest' node the squirrel can reach

H: Ratatoeskr - Sample Solution

Approach 1: Dynamic Programming

- Idea: Dynamic programming by backwards breadth-first-search on the states
- The final states are when the squirrel is at the same position as a raven
- To move between states, consider the possible ways for the ravens and the squirrel to travel (bearing in mind the squirrel cannot move past a raven)
- Already computed states must be skipped to avoid an infinite loop
- Runtime $O(n^4)$

H: Ratatoeskr - Sample Solution

Approach 2: DFSs on the tree

- Idea: Drive the squirrel into the shallowest subtree
- For every node, root the tree at this node and compute its depth
- The minimum depth is the number of signals required
- Runtime: $O(n^2)$

H: Ratatoeskr - Sample Solution

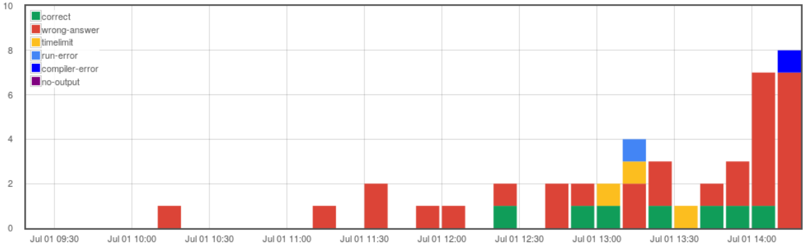
Approach 3: Assign heights to nodes

- Idea: Node heights correspond to the number of signals required (minus 1)
- Leaves are assigned a height of 0
- A node has a height of k if it becomes a leaf after all nodes of height $< k$ have been removed
- The squirrel should travel to the highest node (a 'center' of the tree) it can reach
- Compute the highest position the squirrel can reach from its starting position
- If there are two centers, add 1 for an extra signal
- Runtime: $O(n)$

H: Ratatoeskr - Statistics

- Tried by 22 teams (17%), solved by 7 teams (5%)
- C++: Tried by 19 teams (15%), solved by 7 teams (5%), 38 submissions (18% correct)
- Java: Tried by 3 teams (2%), solved by 0 teams (0%), 7 submissions (0% correct)
- Python: Tried by 0 teams (0%), solved by 0 teams (0%), 0 submissions (0% correct)
- Fastest: 184 minutes, written by 💥🔥💥 PanzerFAUst 💥🔥💥
- Best runtime: 0% of the given time, written by 💥🔥💥 PanzerFAUst 💥🔥💥, <(■v■)>, <(Ov, Hexaflexagons, We came for the Pizzabrötchen, BonnBOS
- Shortest: 1408 characters, written by <(■v■)>

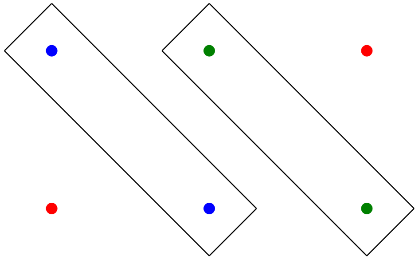
H: Ratatoeskr - Statistics



A: Borders - Sample Solution

Problem

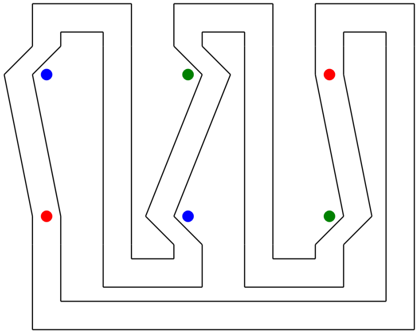
Given a set of blue, red and green points in the plane, draw two polygons so that each color has its own region.



A: Borders - Sample Solution

Solution 1

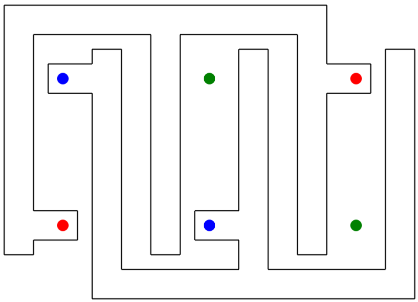
- Sort all points lexicographically, then go column by column.
- Draw the polygons in parallel, shifting left or right depending on color.



A: Borders - Sample Solution

Solution 2

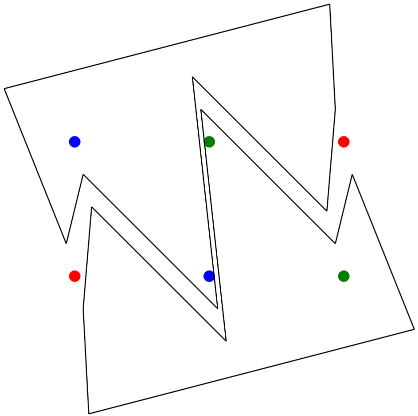
- Sort all points lexicographically, then go column by column.
- Draw two comb-shaped polygons from top and bottom, adding protrusions to "capture" the appropriate color.



A: Borders - Sample Solution

Solution 3

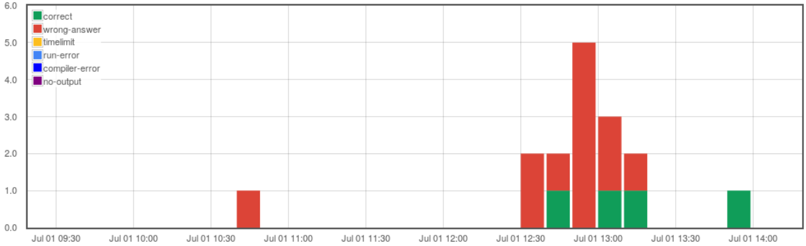
- Pick a direction at random and do a zigzag in that direction.



A: Borders - Statistics

- Tried by 6 teams (5%), solved by 4 teams (3%)
- C++: Tried by 5 teams (4%), solved by 4 teams (3%), 13 submissions (31% correct)
- Java: Tried by 1 teams (1%), solved by 0 teams (0%), 3 submissions (0% correct)
- Python: Tried by 0 teams (0%), solved by 0 teams (0%), 0 submissions (0% correct)
- Fastest: 203 minutes, written by Spezialexpertenkomitee
- Best runtime: 0% of the given time, written by Spezialexpertenkomitee, Lambda, Hello KITty 😼, #define true !!! false
- Shortest: 1984 characters, written by Hello KITty 😼

A: Borders - Statistics



J: Word Clock - Sample Solution

Problem

Given a set S of at most 18 words and a rectangular grid, fill the grid with letters such that every word occurs in the grid.

Solution

Use dynamic programming: for $S' \subseteq S$ and $s \in S'$ let

$$f(S', s) = \text{the earliest possible end position (row,column) of a text containing all the words from } S' \text{ and ending in } s.$$

- A solution exists if and only if one of the positions $f(S, s)$ still lies in the grid.
- To reconstruct the solution one needs to keep track of the best predecessor states while building the DP table (see next slide).

J: Word Clock - Sample Solution

Problem

Given a set S of at most 18 words and a rectangular grid, fill the grid with letters such that every word occurs in the grid.

Solution

Use dynamic programming: for $S' \subseteq S$ and $s \in S'$ let

$$f(S', s) = \text{the earliest possible end position (row,column) of a text containing all the words from } S' \text{ and ending in } s.$$

- A solution exists if and only if one of the positions $f(S, s)$ still lies in the grid.
- To reconstruct the solution one needs to keep track of the best predecessor states while building the DP table (see next slide).

J: Word Clock - Sample Solution

Solution

$f(S', s) =$ the earliest possible end position (row,column) of a text containing all the words from S' and ending in s.

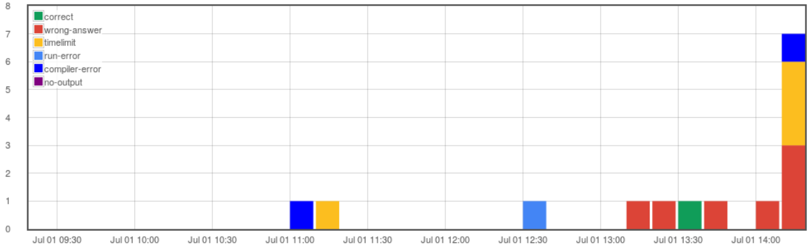
The DP table can be constructed as follows:

- From the current state (S', s) try to append every $t \notin S'$ to the text, maximizing overlap between s and t.
  If t doesn't fit in the current line, start a new line.
  The new state is $(S' \cup \{t\}, t)$.
- The maximal overlaps between words can be precalculated.
  Possible pitfall: some words may be substrings of other words.
- Time complexity: $\mathcal{O}(n^2 2^n)$.

J: Word Clock - Statistics

- Tried by 5 teams (4%), solved by 1 teams (1%)
- C++: Tried by 3 teams (2%), solved by 1 teams (1%), 12 submissions (8% correct)
- Java: Tried by 2 teams (2%), solved by 0 teams (0%), 3 submissions (0% correct)
- Python: Tried by 0 teams (0%), solved by 0 teams (0%), 0 submissions (0% correct)
- Fastest: 255 minutes, written by Hello KITty 😼
- Best runtime: 6% of the given time, written by Hello KITty 😼
- Shortest: 2945 characters, written by Hello KITty 😼

J: Word Clock - Statistics



Thanks

We thank all organizers, problem setters, jury members, contest site organizers and other helpers for their work! Thank you for making the GCPC 2017 possible!

Contest Director

Gregor Schwarz, Technische Universität München

Main Organization

Christian Müller, Technische Universität München
Stefan Jaax, Technische Universität München
Stefan Toman, Technische Universität München

Thanks

Head of Jury

Christian Müller, Technische Universität München

Jury

Gregor Behnke, Universität Ulm
Markus Blumenstock, Johannes Gutenberg-Universität Mainz
Moritz Fuchs, Freelancer
Stefan Jaax, Technische Universität München
Gregor Schwarz, Technische Universität München
Martin Tillmann, Karlsruher Institut für Technologie
Paul Wild, Friedrich-Alexander-Universität Erlangen-Nürnberg

Thanks

Contest Site Organizers

Matthias Bentert (Berlin), Matthias Bentert (Berlin), Dr. André Nichterlein (Berlin), Michael Etscheid (Bonn), Maksym Planeta (Dresden), Michael Baer (Erlangen), Daniela Novac(Erlangen), Tobias Polzer (Erlangen), Paul Wild (Erlangen), Julian Pätzold (Göttingen), Bakhodir Ashirmatov (Göttingen), Dr. Annette Bieniusa (Kaiserslautern), Martin Tillmann (Karlsruhe), Tim Kunold (Lübeck), Prof. Dr. Maciej Liskiewicz (Lübeck), Markus Blumenstock (Mainz), Domenico Mosca (Mainz), Julian Baldus (Saarland), Gregor Behnke (Ulm), Dr. Marianus Ifland (Würzburg), Fabian Lipp (Würzburg), Thomas van Dijk (Würzburg)

Thanks