Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Introduction to Competitive Programming

Jevgenijus Čistiakovas
For DUCSS @ TCD

2021-11-09

Agenda

- Whoami?
- What is competitive programming
- Why do competitive programming
- Main competitions overview
- How to be good at competitive programming + resources
- Sample problem
- Questions

Who am I?

- Jevgenijus or just Eugene.
- 5th year ICS student.
- DUCSS member since 1st year.
- Participated in many programming contests: UKIEPC, NWERC, IrlCPC, Google CodeJam / KickStart / HashCode, Codeforces, Bloomberg CodeCon, …
- Ask here, or by email, DUCSS discord, LinkedIn

What is competitive programming?

- It is a mind sport.
- Similar to academic olympiads.
- The task is to solve mathematical/algorithmic problems with code.
- Can be team vs individual, offline vs online, various formats.
- Often has prizes.

Why do competitive programming?

- Get good at problem solving and algorithms.
- Learn to work under time pressure.
- Get to work as a team.
- Meet other people, become part of big community.
- Win prizes, prestige, get internships, etc.
- Looks good on your CV.
- Good for programming interview, leetcode-style interviews.
- Get to travel … sometimes for free - paid by University or companies.
- **But remember competitive programming != software engineering.**

Main competitions and formats (university competitions)

- **ICPC: UKIEPC, NWERC, World finals**
  - 3 person team competition, only university students.
  - Offline, 1 PC, no internet access. Cheatsheet allowed (max 25 pages).
  - 5hrs for ~10 problems.
  - Pass all test cases to get a point. Penalties for time and incorrect submissions.
- **IrlCPC**
  - 3 person team competition in Cork in March, only Irish university students.
  - Similar to ICPC, but you get points **per each test case passed**.

Main competitions and formats 2 (university + general)

- **Google Code Jam**
  - Multi-round individual competition, usually 2.5hrs, starts in March.
- **Google Kickstart**
  - “Easier” 3hr online competition similar to Code Jam.
  - 8 independent rounds from Mar to Nov.
- **IEEExtreme**
  - 3 person student team competition.
  - 24hrs to solve the problems.
  - Online, but proctor (IEEE member) required to supervise.

Main competitions and formats 3 (general)

- **Codeforces Round**
  - Regular online 2hr competition, individual.
  - Shorter problems, great community.
- **Google HashCode**
  - 4 person team competition.
  - Optimisation task, 4hrs to solve. Scored based on specific score function.
  - Online QF in February (usually hosted by DUCSS), offline finals in April.
- **Kaggle**
  - ML competitions, mostly long-term.

Key takeaways

- There are many competitions and competition format.
  - Understand the rules and decide on an appropriate strategy. E.g. solve easy first to build confidence, try get extra points for being first to solve a problem..
- But there is always a time limit:
  - Hence, never preemptively optimise - it is all about doing the minimum to pass the test cases.
- Many competitions allow to use cheat sheets and templates. Find or compile a list of useful algorithms that you can copy and paste during the competition. Especially, I/O templates are a must have.

How to be good at competitive programming?

- Practice, practice, practice… + participate

- Learn about algorithms and data structures.
- Learn about algorithms complexity analysis. Get an intuition of what maximum complexity can work with given constraints.
- Get comfortable with coding. C++/Java/Python?
- How to practice:
  - Attempt a problem under time constraints.
  - If you cannot solve it, then read editorial AND implement the solution. Or find a solution on Github AND re-implement it.

Main topics in ICPC-style competitions

- Implementation
- Searching/sorting, classical algorithms
- Data structures - basic and advances
- String (e.g. KMP)
- Graphs
- Dynamic programming (DP)
- Ad-hoc
- Greedy
- Game Theory
- Geometry
- Network flow
- Randomised algorithms
- Maths
  - Probability
  - Combinatorics
  - Number theory
  - Numerical methods

Study material

- Your favourite algorithms book, e.g. Introduction to Algorithms by Cormen, Leiserson, Rivest, and Stein. (CLRS) or Algorithms by Segdewick.
- Competitive Programmer’s Handbook by Antti Laaksonen.
- Competition solutions and write-ups. See Google Code Jam/KickStart, Codeforces, NWERC, search Github etc.
- E-Maxx Algorithms in English -- cp-algorithms.com
- Watch Errichto et al. on Youtube.

Online contests and problems with automatic scoring

- Kattis -- open.kattis.com -- has past ICPC-style problems.
- Codeforces -- codeforces.com
- Topcoder -- topcoder.com
- CodeChef -- codechef.com
- SPOJ -- spoj.com
- AtCoder -- atcoder.jp
- Google Code Jam / Kickstart archives -- codingcompetitions.withgoogle.com
- Leetcode -- leetcode.com
- Kaggle -- kaggle.com
- Project Euler -- projecteuler.net

Reminder - reality check

- Competitive programming is meant to be hard. But don’t be intimidated.
- Don’t be afraid to fail.
- Set realistic goals. E.g. don’t necessary aim for 1st place.
- Look for opportunities to learn, to have fun, make friends.
- Competitions often cater for very different levels of skill, e.g. there are usually a few problems everyone can solve.
- Top competitive programmers have done it since early secondary school, have dedicated coaches, etc. Don’t expect to beat them, at least not immediately ;)
- Remember it is a sport and it does not reflect your professional skills.

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Questions?

Sample ICPC-style problem format

- Problem statement
- Input
- Output
- Constraints
- Time limit
- Memory limit
- Examples

Sample problem - Elegant Showroom

From UKIEPC 2016 - see https://open.kattis.com/problems/showroom

Problem statement:

A car showroom is one of the few places where cars can be found indoors. Showrooms often have many cars, even above ground level! As cars are sold and new cars are bought in to sell, the cars must be moved carefully out of the showroom.

Clearly employees only wish to move cars if they have to. So, given a map of a showroom including its walls, doors and where the cars are, and the co-ordinates of the car to move, how many cars must be moved?

Cars can be rotated on the spot, but can only be moved through a completely empty space and not diagonally. Doors are always wide enough to move a car through.

Sample problem - Elegant Showroom

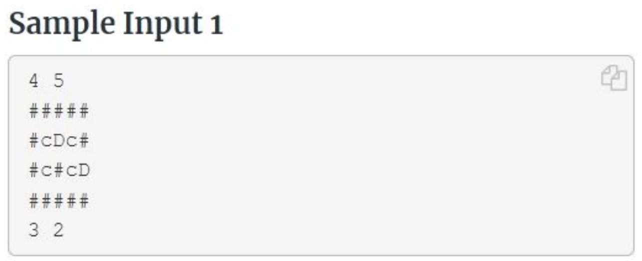
From UKIEPC 2016 - see https://open.kattis.com/problems/showroom

Input:

- One line containing two integers R,C (3≤R,C≤400), the size of the showroom in rows and columns.
- Another R lines, each containing a string of C characters with the following meaning:
  - ’#’: a wall;
  - ’c’: a car;
  - ’D’: a door in a wall.
- The first and last lines must be walls or doors. The first and last characters in a row must be walls or doors.
- The next line will contain two integers r (1<r<R), and c (1<c<C), the co-ordinates of the car to move. Where 1,1 is the top-left corner.

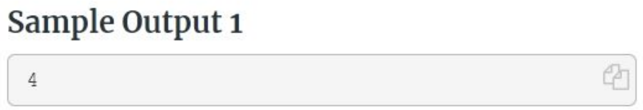
Output:

- One line containing one integer: the smallest number of cars that need to be moved (<u>including the car we are moving</u>) to allow our desired car to leave the building.



Sample Input 1

```
4 5
#####
#cDc#
#c#cD
#####
3 2
```



Sample Output 1

```
4
```

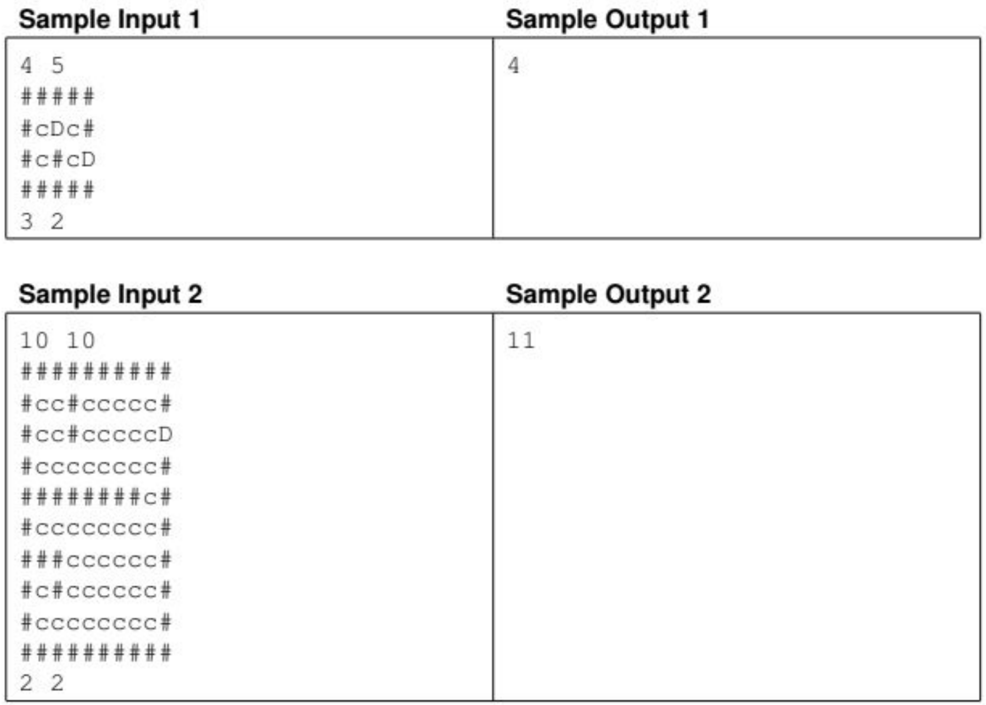
Sample problem - Elegant Showroom

From UKIEPC 2016 - see https://open.kattis.com/problems/showroom

Examples: ------------------>

Time limit: 1 second

Memory limit: 1024 MB



Sample Input 1

```
4 5
#####
#cDc#
#c#cD
#####
3 2
```

Sample Output 1

```
4
```

Sample Input 2

```
10 10
##########
#cc#ccccc#
#cc#cccccD
#cccccccc#
########c#
#cccccccc#
###cccccc#
#c#cccccc#
#cccccccc#
##########
2 2
```

Sample Output 2

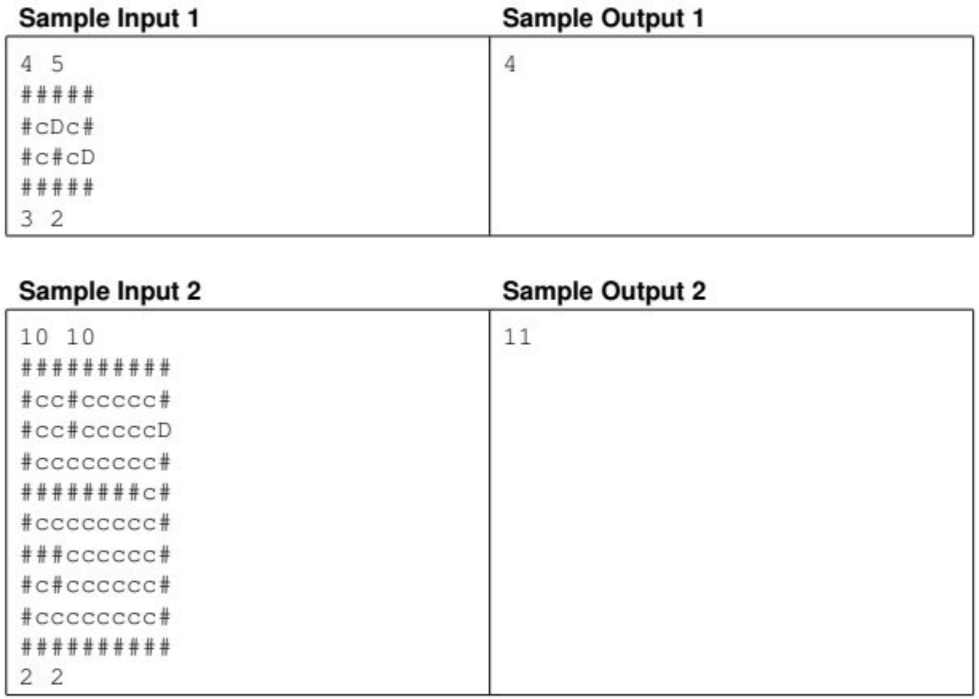
```
11
```

Sample problem - Elegant Showroom

From UKIEPC 2016 - see https://open.kattis.com/problems/showroom

Overview:

- A map of a car showroom with doors(D), cars(c) and walls(#).
- There can be many doors in the outer wall leading to the target.
- Given the coordinates of a car in the showroom, how many cars must be moved in total.



Sample Input 1

```
4 5
#####
#cDc#
#c#cD
#####
3 2
```

Sample Output 1

```
4
```

Sample Input 2

```
10 10
##########
#cc#ccccc#
#cc#cccccD
#cccccccc#
########c#
#cccccccc#
###cccccc#
#c#cccccc#
#cccccccc#
##########
2 2
```

Sample Output 2

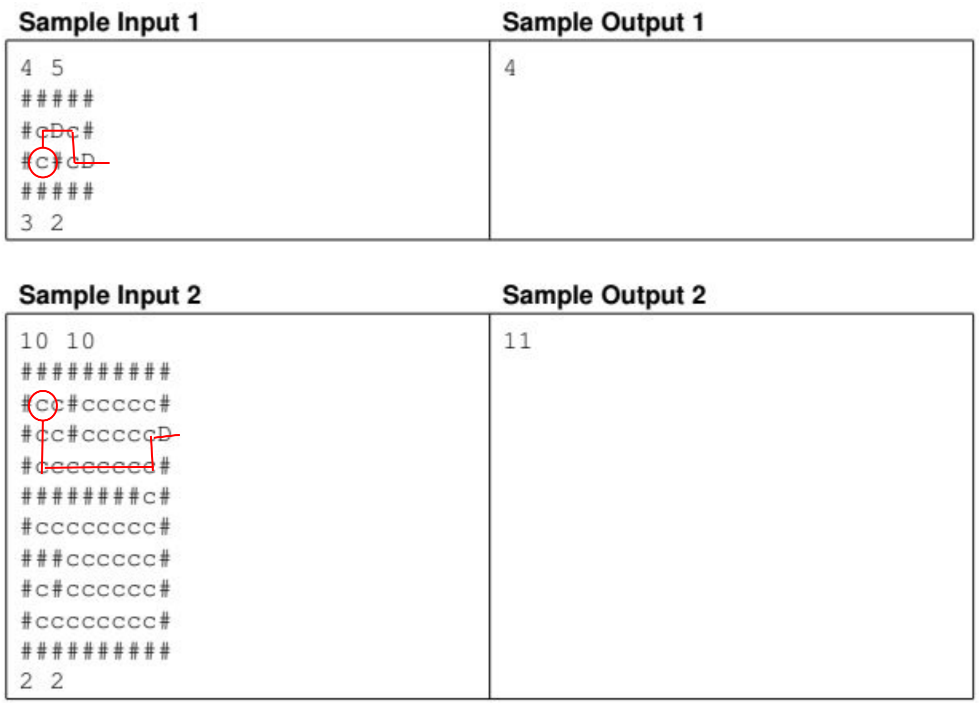
```
11
```

Sample problem - Elegant Showroom

From UKIEPC 2016 - see https://open.kattis.com/problems/showroom

Overview:

- A map of a car showroom with doors(D), cars(c) and walls(#).
- There can be many doors in the outer wall leading to the target.
- Given the coordinates of a car in the showroom, how many cars must be moved in total.



Sample Input 1

```
4 5
#####
#cDc#
#c#cD
#####
3 2
```

Sample Output 1

```
4
```

Sample Input 2

```
10 10
##########
#cc#ccccc#
#cc#cccccD
#cccccccc#
########c#
#cccccccc#
###cccccc#
#c#cccccc#
#cccccccc#
##########
2 2
```

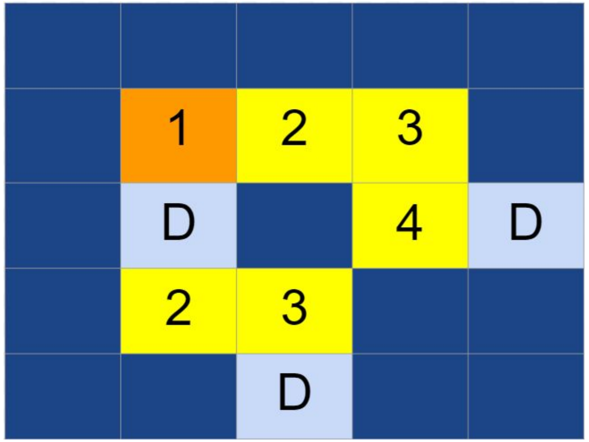
Sample Output 2

```
11
```

Elegant Showroom - Solution

Techniques

- Dijkstra's algorithm
- Breadth-first search



Algorithm

- Read in the 'map' of the showroom and build a graph. Make a note of the doors on the edges.

- Use Dijkstra's algorithm to find the distance to the target car.
  - Weight each node. 1 for a car, 0 for a door.
  - Push all of the edge doors onto a priority queue at once, distance 0
  - Starting a new search from each door is slow.
    - About 1,500 times slower, in fact.

- See also: Sokoban for a harder challenge with the same idea

Sample problem - Elegant Showroom

From UKIEPC 2016 - see https://open.kattis.com/problems/showroom

Learnings:

- Problem comprehension.
- Input representation.
- Be careful with 0-indexing vs 1-indexing.
- Reusing well-known existing algorithms.

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Questions?

Sample problem - Clock Pictures

From NCPC 2014 - see https://open.kattis.com/problems/clockpictures

Problem statement:

You have two pictures of an unusual kind of clock. The clock has n hands, each having the same length and no kind of marking whatsoever. Also, the numbers on the clock are so faded that you can’t even tell anymore what direction is up in the picture. So the only thing that you see on the pictures, are n shades of the n hands, and nothing else.

You’d like to know if both images might have been taken at exactly the same time of the day, possibly with the camera rotated at different angles.

Task:

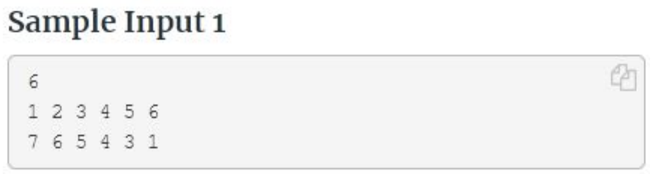
Given the description of the two images, determine whether it is possible that these two pictures could be showing the same clock displaying the same time.

Sample problem - Clock Pictures

From NCPC 2014 - see https://open.kattis.com/problems/clockpictures

Input:

– The first line contains a single integer n (2≤n≤200000), the number of hands on the clock.
– Each of the next two lines contains n integers ai (0≤ai<360000), representing the angles of the hands of the clock on one of the images, in thousandths of a degree. The first line represents the position of the hands on the first image, whereas the second line corresponds to the second image. The number ai denotes the angle between the recorded position of some hand and the upward direction in the image, measured clockwise. Angles of the same clock are distinct and are not given in any specific order.

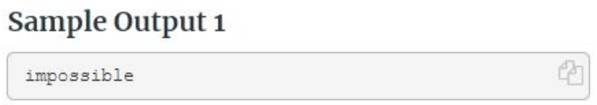


Sample Input 1

```
6
1 2 3 4 5 6
7 6 5 4 3 1
```

Output:

– Output one line containing one word: possible if the clocks could be showing the same time, impossible otherwise.



Sample Output 1

```
impossible
```

Sample problem - Clock Pictures

From NCPC 2014 - see https://open.kattis.com/problems/clockpictures

Examples:

Sample Input 1

```
6
1 2 3 4 5 6
7 6 5 4 3 1
```

Sample Output 1

```
impossible
```

Sample Input 2

```
2
0 270000
180000 270000
```

Sample Output 2

```
possible
```

Sample Input 3

```
7
140 130 110 120 125 100 105
235 205 215 220 225 200 240
```

Sample Output 3

```
impossible
```

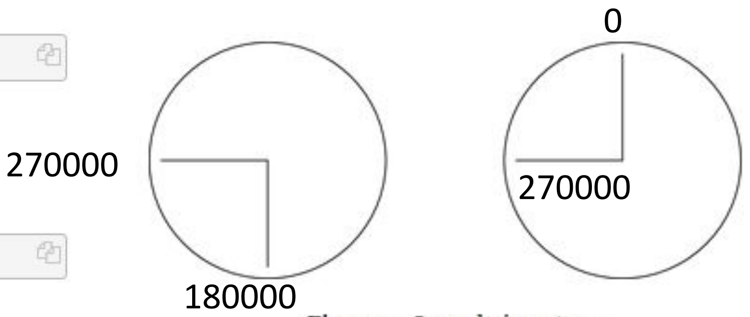
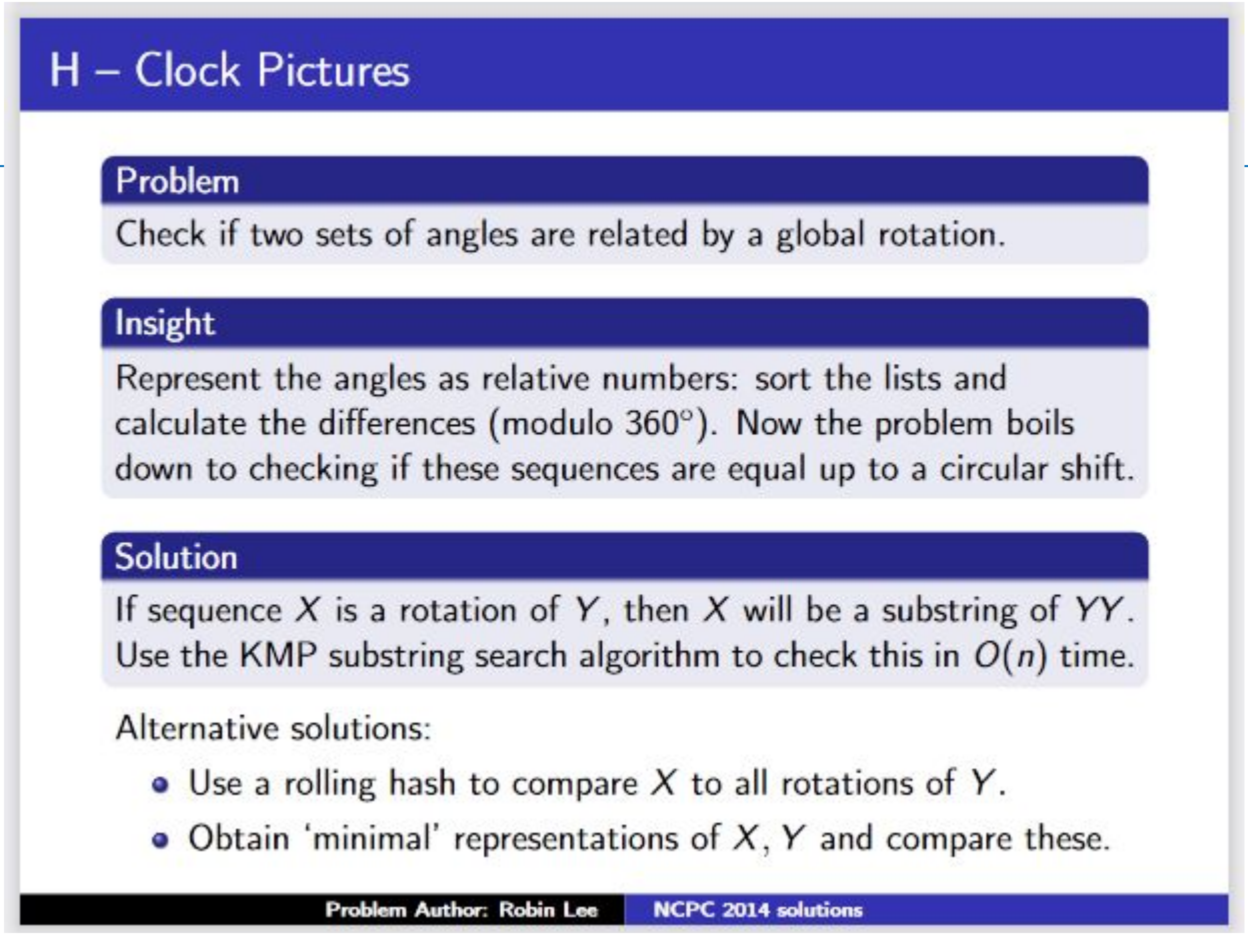


Figure 1: Sample input 2



H – Clock Pictures

Problem

Check if two sets of angles are related by a global rotation.

Insight

Represent the angles as relative numbers: sort the lists and calculate the differences (modulo 360°). Now the problem boils down to checking if these sequences are equal up to a circular shift.

Solution

If sequence X is a rotation of Y, then X will be a substring of YY. Use the KMP substring search algorithm to check this in $O(n)$ time.

Alternative solutions:

- Use a rolling hash to compare X to all rotations of Y.
- Obtain 'minimal' representations of X, Y and compare these.

Problem Author: Robin Lee NCPC 2014 solutions

Sample problem - Clock Pictures

From NCPC 2014 - see https://open.kattis.com/problems/clockpictures

Learnings:

- Carefully check the limits.
- Transform the problem into a string search.
- Re-use existing algorithms like KMP

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Questions?

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

That’s all Folks!
Thank you!