NWERC 2020 presentation of practice solutions

A: Another Eruption

Problem Author: Jeroen Bransen, Bjarki Ágúst Guðmundsson

Problem

We want to put a barrier tape around the border of a circular gas cloud. The area of the gas cloud in metres^2 is already known. Tell us its perimeter.

A: Another Eruption

Problem Author: Jeroen Bransen, Bjarki Ágúst Guðmundsson

Solution

- The area a of a circle with radius r is given by πr^2.
- The perimeter p of such a circle is $2\pi r$.
- Because $a = \pi r^2$, we know $r = \sqrt{\frac{a}{\pi}}$.
- Hence $p = 2\pi\sqrt{\frac{a}{\pi}} = \sqrt{4\pi a}$.

Gotchas

- Remember to print with high-precision:
  - C++: `cout.precision(12)` or `printf("%.9f\n", p)`
  - Python: `"{:.9f}".format(p)`
  - Java: `System.out.printfln("%.9f\n", p)`
- Use `long` or `double` to read the input, $10^{18} > 2^{31}$

Statistics: 123 submissions, 106 accepted

B: Broken Gearbox

Problem Author: Jim Grimmett

Problem

Put a set of gears back on spindles so that they all mesh

Solution

- Choose an arbitrary "root" node n, and let x be the value of the gear placed there.
- If nodes n and m have an edge weight of w_{nm}, then the gear on node m must be $(w_{nm} - x)$. Gears on nodes adjacent to m are $(w_{mo} - w_{nm} + x)$, and so on. Fixing the value of x, we can express all other gears as formulae of the form $c_i \pm x$.
- We can use this to generate a solution in $O(n)$ time for any given value of x. There are $O(n)$ different possible values for x so this is too slow. We need to pre-filter.
- The largest gear will be on the node with the largest value of $c_i \pm x$.
  - This could be a node with $c_i + x$ or with $c_i - x$. We can't be sure which is higher, so we'll check both.
- Plug in the two possible values of x, generate the other values, and check **all** of the constraints.
  - That we use the right quantity of every gear.
  - That every edge constraint w_{ab} is satisfied.

B: Broken Gearbox

Problem Author: Jim Grimmett

Gotchas

- Odd cycles (or general inconsistencies in the graph) mean that just using the formulae to check solutions for correctness won't always work.
- You need to traverse the whole graph checking for flaws in the solution before outputting it.

Statistics: 80 submissions, 26 accepted

C: Cheating

Problem Author: Jeroen Bransen

Problem

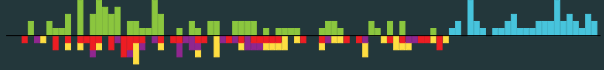
In an interactive problem, try to find out the 4-letter password in at most 50 guesses. For each guess, the system reports the number of letters in the correct position (a), and the number of letters that are in the password but in the wrong position.

Solution

- Do 26 guesses: `aaaa`, `bbbb`, ..., `zzzz`
- Now we know which 4 letters to use
- There are $4! = 24$ possible orders for these letters
- The right password will be guessed within $26 + 24 = 50$ tries

C: Cheating

Problem Author: Jeroen Bransen



```
guess :: String -> IO (Int,Int)
guess g = do
  putStrLn g
  hFlush stdout
  s <- getLine
  case words s of
    ["correct"] -> exitSuccess
    [a,b] -> return (read a, read b)

main :: IO ()
main = do
  cs <- forM ['a'..'z'] $ \c -> do
    (a,b) <- guess $ replicate 4 c
    return $ replicate a c
  mapM_ guess $ permutations $ concat cs
```

C: Cheating

Problem Author: Jeroen Bransen

Gotchas

- Flush the output:
  - C++: `cout << s << endl` (no explicit flush needed)
  - Python: `print(s, flush=True)`
  - Java: `System.out.flush()`
- Stop once you got the right answer (it could be aaaa)
- Testing tool may crash on certain errors / race conditions, but for correct solutions it will always work correctly.

C: Cheating

Problem Author: Jeroen Bransen

Alternative solution

- Keep a list of all $26^4 = 456976$ possible passwords
- Guess a random password from that list
- Filter the list with the information we got
- Repeat until we guess correctly

Statistics: 220 submissions, 82 accepted

Language stats