NCPC 2009
Presentation of solutions

Heads of Jury:
Marcus Isaksson and Jon Marius Venstad

2009-10-03

NCPC Jury

Heads of Jury

- Marcus Isaksson (Chalmers)
- Jon Marius Venstad (NTNU)

Problem authors

- Andreas Björklund (ARM)
- Gunnar Kreitz (KTH)
- Jimmy Mårdell (Silobreaker)
- Jon Marius Venstad (NTNU)
- Marcus Isaksson (Chalmers)
- Mikael Goldmann (KTH)

A - Soda Surpler

Solution

- Maintain the current number of empty bottles,.
- and the number of bottles enjoyed.
- Keep on drinking while you can buy another:
  - Buy a soda.
  - Enjoy it.
  - Keep the bottle.

Translated to a program

- $b := e + f$
- $d := 0$
- while $b >= c$:
  - $b := b - c$
  - $d := d + 1$
  - $b := b + 1$

Alternative solution

- if $e + f = 0$:
  - 0
- else:
  - $(e + f - 1)/(c - 1)$

D - Rain Fall

Input

L: leak position (mm)
K: rate of leaks (mm/h)
T_1: duration of rainfall (h)
T_2: wait time after rainfall (h)
H: water level when observed (mm)

Output

F_1: minimal rainfall (mm)
F_2: maximal rainfall (mm)

Solution

- $H < L$: return $F_1 = F_2 = H$
- $H > L$: Let R be the rate of rain fall (mm/h). Writing the water level right after the rainfall in two ways gives:

$$L + (R - K)(T_1 - L/R) = H + KT_2$$

  Solve for R and return $F_1 = F_2 = RT_1$.
- $H = L$: return $F_1 = L$ and $F_2 = RT_1$.

I - Playfair Cipher

playfair example

P	L	A	Y	F
I	R	E	X	M
B	C	D	G	H
J	K	N	O	S
T	U	V	W	Z

hide the gold in the tree stump

`hi de th eg ol di nt he tr eX es tu mp`
`BM ND ZB XD KY BE JV DM UI XM MN UV IF`

Solution

Follow the specification.

- Fill in a table given the key phrase.
- Massage plain text (remove whitespace and insert extra X's).
- Encrypt pairs of characters as specified.

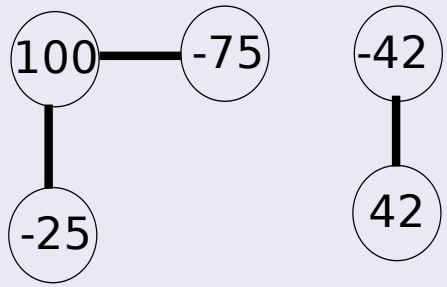
B - Money Matters

Problem

n persons with debts $o_1, \ldots, o_n$. Can the debts be settled by only transferring money between friends?

Solution

- View the persons as vertices in a graph with edges between friends.
- Note: any amount of money can move between two vertices that are connected.
- Return `POSSIBLE` if the debts in each connected component sum to zero.

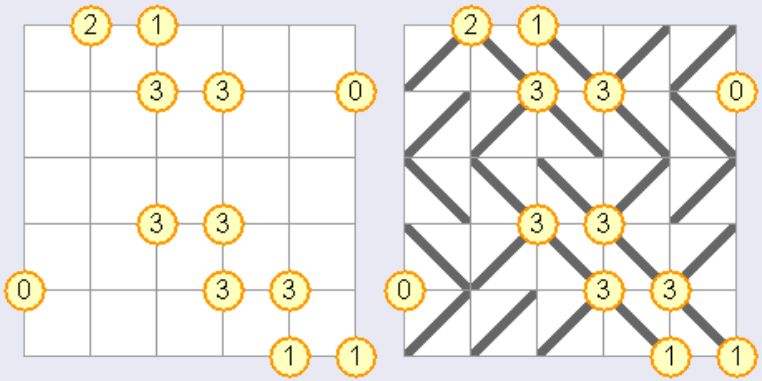


F - Gokigen

Problem

Draw one diagonal in each cell such that

- Number in circle = number of diagnoals coming in to circle
- Diagonals do not form an enclosed loop.



Solution

- Backtracking - keep trying to fill in diagonals and backtrack when not possible to fill in more (without having too many or too few diagonals at a circle, or a loop)
- Optimization: always fill in diagonals that have no choice.

G - Flight Planning

Solution

- Define the *width* $\mathrm{w}(T)$ of a tree T as the longest possible path in the tree.
- The flight graph is a tree T, and removing one edge yields two new trees, T_1, T_2.
- The minimal width we can get when joining T_1 and T_2 with a single edge is $\max\{\mathrm{w}(T_1), \mathrm{w}(T_2), \lceil \frac{\mathrm{w}(T_1)}{2} \rceil + \lceil \frac{\mathrm{w}(T_2)}{2} \rceil + 1\}$
- The width of a tree can easily be found by iteratively removing leaf nodes.
- For each edge e in T, find width of the trees obtained by removing e and calculate the minimal width we can obtain when combining these.

C - Allergic Test

Solution

- Each allergen must have at least one day in which it is the only active allergen.
- When adding a new allergen to the end of another scheme, the only property that matters is the number of days the last allergen is the only one active (and the length of the scheme, obviously).
- DP on scheme length (`M[i][stub]`), over the subset of allergens tested (`i`, bitmask) and the number of days the last allergen is the only one active (`stub`).
- for $\texttt{i} = 1 \ldots 2^{\texttt{k}} - 1, \texttt{j} \in \texttt{i}, \texttt{old} = 1 \ldots 7$

  $\texttt{stub} \leftarrow \max(\texttt{D[j]} - \texttt{old} - 1, 0)$

  $\texttt{M[i][stub]} \leftarrow \min(\texttt{M[i][stub]}, \texttt{M}[\texttt{i} - 2^{\texttt{j}}][\texttt{old}] + \texttt{stub} + 1)$

E - Speedy Escape

Problem

Given city map - a graph - find the minimal speed with which the Brothers' car can reach an exit without any possibility of getting caught by the police car (driving at 160km/h) before.

Solution

- Use Dijkstra to find the earliest time the police can be at each intersection.
- For a fixed speed S of the brothers car, we can use a variation of Dijkstra to find the earliest time (or possibly Infinity) that the Brothers can be at each intersection - without risking getting caught.
- Use binary search to find the minimal S such the Brothers can exit without risk.

J - Code Permutations

Foo

- The order of a permutation is the least common multiple of the lengths of its cycles.
- Let M_n^k be the permutations of n elements of orders m_i where $k|m_i$ and $m_i|K$ for all i. Define $M_0^1 = 1$ and $M_n^k = 0$ for $n \leq 0, k > 1$ otherwise. We want to compute $|M_N^K|$.
- Adding a cycle of length j to a permutation in M_n^k such that the last element is in this cycle can be done in $\binom{n-1}{j-1}$ ways, each yielding a unique permutation in $M_{n+j}^{\mathrm{LCM}(j,k)}$. Let $\rho_K(j,k) = \frac{k}{j}$ if j is a maximal order prime power of K and $\rho_K(j,k) = k$ otherwise. Then:

$$|M_n^k| = \sum_{j|K} \binom{n-1}{j-1} |M_{n-j}^{\rho_K(j,k)}|$$

H - Beacons

Solution

- The naïve DFS algorithm is $O(n^3)$, and will time out. Some options:
- Develop a good heuristic for selecting beacons and mountains to check, e.g. when checking for mountains that block the view to a beacon, choose mountains with angle close to the angle of the beacon, then beacons close to this mountain etc.
- From each beacon, keep track of which angles are blocked (in an efficient way). Check beacons and mountains by increasing distance and add blocked angles as mountains are encountered. Or check mountains by increasing distance, and then check if beacons within the angle interval of a mountain are behind it or not.
- Possibly other smart tricks and optimizations.