Problem Analysis Session

SWERC Judges

SWERC 2016

C - Candle Box

Problem

Consider two finite arithmetic progressions:

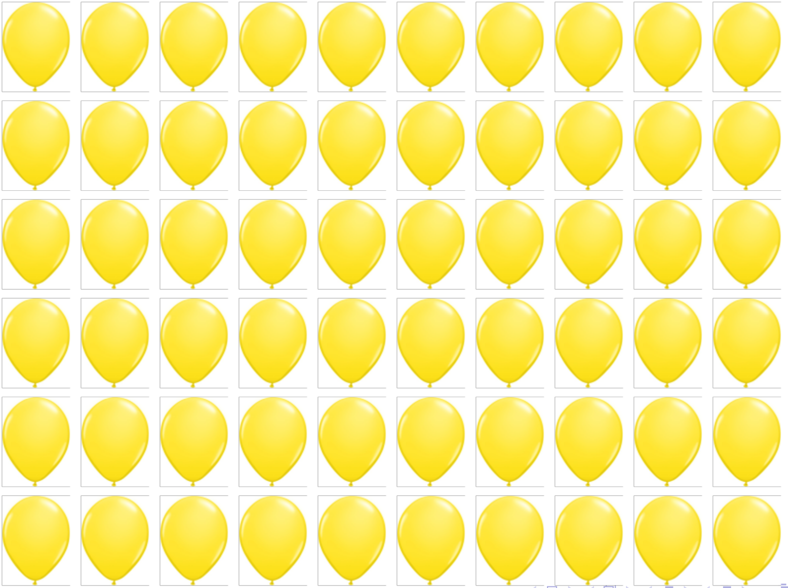
- $R = 4 + 5 + \cdots$ number of Rita's candles
- $T = 3 + 4 + \cdots$ number of Theo's candles

Theo placed X of its candles in Rita's box.
Given $R + X$, $T - X$, and D (difference between the ages of Rita and Theo), find out X.

Categories

- Math, arithmetic progression

C - Candle Box



Sample Solution

Given $(R + X)$, $(T - X)$, and D:

- $R = \sum_{k=0}^{n} (4 + k)$ and $T = \sum_{k=0}^{n-D+1} (3 + k)$
- To compute R, find out n such that $R + T = (R + X) + (T - X)$
  - either analytically
  - or iteratively computing the sum of arithmetic progressions R, T, until the condition holds
- The answer is $(R + X) - R$

K - Balls and Needles

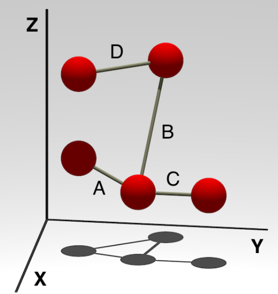
Problem

Given a set of needles / line segments $((x_1, y_1, z_1), (x_2, y_2, z_2))$ in 3D:

- Is there a true closed chain, i.e., a cycle in the (simple) graph whose edges are the line segments?
- Is there a floor closed chain, i.e., a cycle in the (simple) graph whose edges are the line segment projections $((x_1, y_1), (x_2, y_2))$?

Categories

Graphs – Cycle detection



K - Balls and Needles

Sample Solution

- Sculpture with K needles / line segments
- **Graph for 3D**:
  - $|E| = K, \ |V| \leq 2K$
  - Map each point (x, y, z) to an integer $0, 1, 2, \ldots$
- **Graph for 2D:**
  - $|E| \leq K, \ |V| \leq 2K$
  - Map each point (x, y) to an integer $0, 1, 2, \ldots$
  - Leave self-loops out
  - Do not add repeated edges
- **Cycle detection:** DFS or Union Find
- Time and space complexity: $\mathcal{O}(K)$

H - Hyper-Pyramids

Problem

Calculate the numbers at the base of Pascal's "triangle" generalized to higher-dimensions.

Categories

Math, dynamic programming

H - Hyper-Pyramids

General recurrence for dimension D:

$$
\begin{aligned}
C(x_1,\dots,x_D) &= C(x_1-1,x_2,\dots,x_D) \\
&\quad + C(x_1,x_2-1,\dots,x_D) \\
&\quad + \cdots \\
&\quad + C(x_1,x_2,\dots,x_D-1)\,, && \text{if all } x_i > 0 \\
C(x_1,\dots,x_D) &= 1 && \text{otherwise.}
\end{aligned}
$$

Base of the hyper-pyramid of height H:

$$\{C(x_1,\dots,x_D) \mid 0 \le x_i < H \ \wedge \ \textstyle\sum_i x_i = H-1\}$$

H - Hyper-Pyramids

Implementation:

- Use 64-bit integer arithmetic
- Recursively generate vectors of size D that sum $H-1$
- Calculate recurrence using memoization
- Collect results into a set
- Choice of memo and set data structures is not critical

However, the above is not enough, we need to exploit symmetry:

- $C(x_1, \ldots, x_D) = C(\pi \circ (x_1, \ldots, x_D))$ for any permutation π
- Normalize memo keys, e.g. ascending order
  (sort vectors before lookup/insert)
- Generate only ascending vectors $x_1 \leq \ldots \leq x_D$ that sum $H-1$

H - Hyper-Pyramids

Alternative approach (for the more math-savvy):

- Numbers in Pascal's hyper-pyramids are the multinomial coefficients
- Compute directly using the closed factorials formula:

$$C(x_1, x_2, \ldots, x_D) = \binom{H-1}{x_1, x_2, \ldots, x_D} = \frac{(H-1)!}{x_1!x_2! \cdots x_D!}$$

- No recurrence, no memoization
- Caveat: requires larger precision (BigIntegers) or careful implementation to avoid overflow with 64-bits

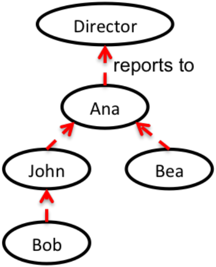
F - Performance Review

Problem

Given a rooted tree with N nodes, where each node has a (key, value) pair, find for each node x the sum of values in its children whose key is smaller than x's key.

Categories

- Graph Theory - Trees
- Data Structures



F - Performance Review

Naive approach

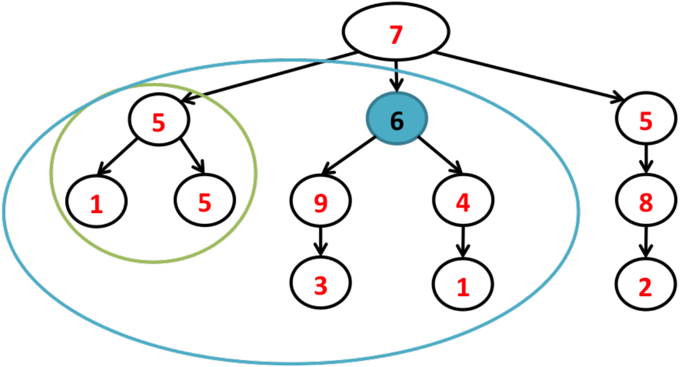
- Process each node's children and sum the values
- Complexity is $\mathcal{O}(N^2)$ — too slow

Several valid approaches

- Post-order traversal of the tree + Fenwick Tree
- Pre-order traversal + Segment Tree
- Process nodes from the deepest leaves to the root and merge (key,value) pairs

F - Performance Review

Sample Solution



Sample Solution

- Use a data structure $\mathcal{T}$ that supports:
  - Add(k, v): add the (key, value) pair to $\mathcal{T}$
  - Sum(key): return the sum of values in $\mathcal{T}$ with keys less than *key*
- Post-order traversal of the given tree

```
Dfs(node, T)
    prev = T.Sum(node.key)
    for each child of node
        Dfs(child, T)
    node.review_time = T.sum(node.key) - prev
    T.Add(node.key, node.value)
```

- Using an efficient data structure like a Fenwick Tree, this approach takes $\mathcal{O}(N \log N)$ time and $\mathcal{O}(N)$ space

B - Bribing Eve

Problem

Find the best and worst positions that a product m from a list can have when sorted by a score:

$$s = w_1 x_1 + w_2 x_2 \qquad \text{with } w_1, w_2 \in \mathbb{R}_0^+ \text{ and } w_1 + w_2 > 0.$$

Categories

- Sweep
- Geometry

B - Bribing Eve

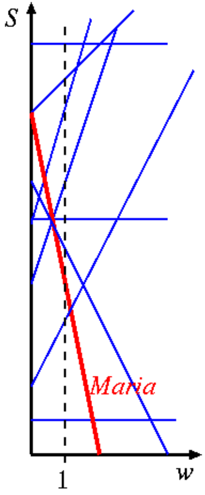
Naive approach

- For every product i find the weights $s_m = s_i$.
- Use every valid weight found to sort the products and find the best and worst ranking position.
- Complexity is $\mathcal{O}(N^2 \log N)$ — too slow

Sweep

- Find a way to sweep through all valid weights while keeping a set of best/worst products.
- Complexity is $\mathcal{O}(N \log N)$

B - Bribing Eve



$$
\begin{aligned}
S &= w_1 x_1 + w_2 x_2 \\
\frac{1}{w_1 + w_2} S &= \frac{w_1}{w_1 + w_2} x_1 + \frac{w_2}{w_1 + w_2} x_2
\end{aligned}
$$

Therefore, we can reduce the analysis to:

$$S = w_1 x_1 + w_2 x_2, \text{ with } w_1 + w_2 = 1$$

That is, $S = x_2 + (x_1 - x_2)w$, with $0 \leq w \leq 1$.

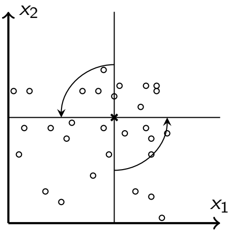
Solution approach:

Find the intersection points in $O(n)$. Sort them.

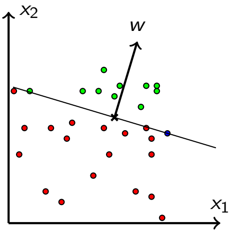
Linear Sweep to update counters.

B - Bribing Eve

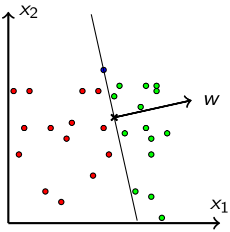
Radial sweep



Best ranking



Worst ranking



Caveats

- Watch for boundary conditions - degenerate cases.
- Be careful of products with same test scores.
- Test with epsilon if solving with floats.

D - Dinner Bet

Problem

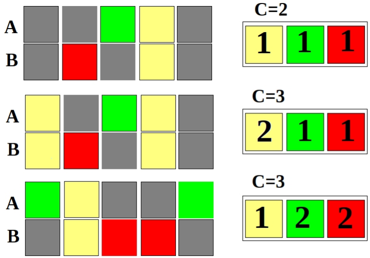
Given the number N of balls, the number D of balls drawn each round and the C values chosen by each of the **two players**, find the expected number of rounds their game will last.

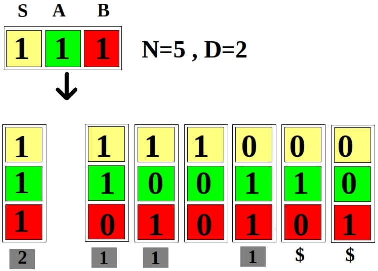
Categories

- Dynamic Programming
- Probabilities, Combinatorics
- (Absorbing Markov Chains)

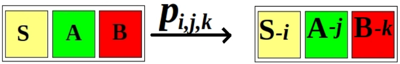
D - Dinner Bet

State abstraction **Shared, Player A, Player B**





State Transition (with probability $p_{i,j,k}$)



$$D-(N-S-A-B) \leq i+j+k \leq D$$

$$p_{i,j,k} = \frac{\binom{S}{i}\binom{A}{j}\binom{B}{k}\binom{N-A-B-S}{D-i-j-k}}{\binom{N}{D}}$$

Sample Solution

- The game ends when $S = 0 \;\wedge\; (A = 0 \vee B = 0)$
- The expected number of rounds satisfies the recurrence

$$\mathcal{E}_{R,S,A,B} = \sum_{i+j+k \leq D} p_{i,j,k}\, \mathcal{E}_{R-1,S-i,A-j,B-k}$$

$$\mathcal{E}_{R,0,A,0} = \mathcal{E}_{R,0,0,B} = R$$

- Compute $\mathcal{E}$ using Dynamic Programming: $\mathcal{O}(C^3 D^3)$ time

Another formula. . .

$$\begin{aligned} \mathbb{E}_{S,A,B} &= \frac{1}{1 - p_{0,0,0}} \left(1 + \sum_{i+j+k \neq 0} p_{i,j,k}\, \mathbb{E}_{S-i,A-j,B-k} \right) \\ \mathbb{E}_{0,A,0} &= \mathbb{E}_{0,0,B} = 0 \end{aligned}$$

D - Dinner Bet

Sample Solution

- The game lasts the most when $N = 50$ and $D = 1$ and the players have the same key
- With 1 number left, the of probability continuing playing after $\mathcal{R}$ rounds is $(\frac{49}{50})^{\mathcal{R}} = 0.98^{\mathcal{R}}$
- The contribution to the result is $\mathcal{R} \times 0.98^{\mathcal{R}}$, so an estimate for the minimum number of iterations needed is finding $\mathcal{R} \times 0.98^{\mathcal{R}} < EPS$.
- Considering 1000 rounds was more than enough for a 10^{-3} epsilon

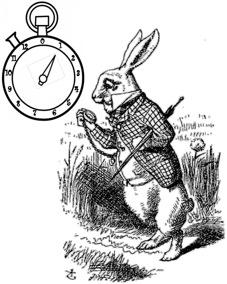
I - The White Rabbit Pocket Watch

Problem

Given a set of tracks over an undirected weighted graph with unknown edge weights, along with the sum modulo 13 of its edge weights, determine the weight of the shortest path between two given nodes.

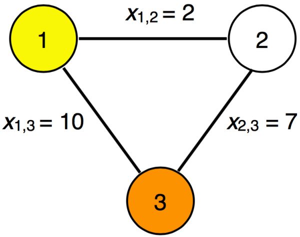
Categories

- Graph theory / shortest path
- System of linear equations in a prime field



I - The White Rabbit Pocket Watch

Tracks	Duration in $\mathbb{Z}_{13}$
$1 \to 2 \to 3 \to 2$	3
$1 \to 2 \to 1$	4
$1 \to 2 \to 1 \to 3$	1



$$\begin{cases} x_{1,2} + 2\,x_{2,3} \equiv 3 \mod 13 \\ 2\,x_{1,2} \equiv 4 \mod 13 \\ 2\,x_{1,2} + x_{1,3} \equiv 1 \mod 13 \end{cases}$$

The system has a unique solution.

I - The White Rabbit Pocket Watch

Solution Overview:

- Build a system of linear equations from the track information
- Solve the system by Gaussian elimination in $\mathbb{Z}_{13}$ to find the edge weights (the problem statement guarantees a unique solution)
- Compute the shortest path between the target nodes (e.g. Dijkstra)
- Time complexity is $O(|E|^3)$

G - Cairo Corridor

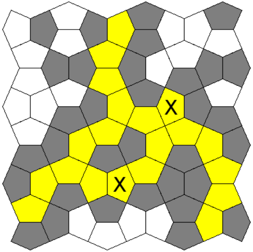
Problem

Given a pentagonal tiling:

- Is there a corridor, i.e., a maximal set of clear adjacent pentagons that connect the 4 borders?
- (If yes) Is the corridor minimal, i.e., does it have no superfluous pentagon?
- (If yes) How many pentagons has the minimal corridor?

Categories

Graphs – Connectivity

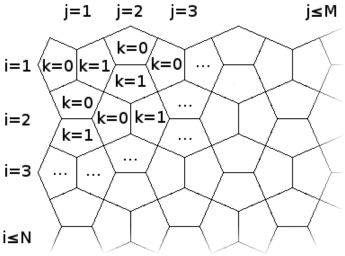


G - Cairo Corridor

Sample Solution

- Model empty tiles and their adjacencies as a graph
  - Each tile identified by (i, j, k)
  - Adjacency and border checks by case analysis, e.g. left border:

    $(i + j \text{ is even} \wedge j = 1 \wedge k = 0) \vee$
    $(i + j \text{ is odd } \wedge j = 1)$
  - $|V| \leq 2 \times N \times M \approx 125\,000$,
    $|E| \leq 5 \times 125\,000$



- Search for a corridor (connected component that touches all borders): DFS, BFS or Union Find
- Check the corridor for minimality

G - Cairo Corridor

How to check minimality?

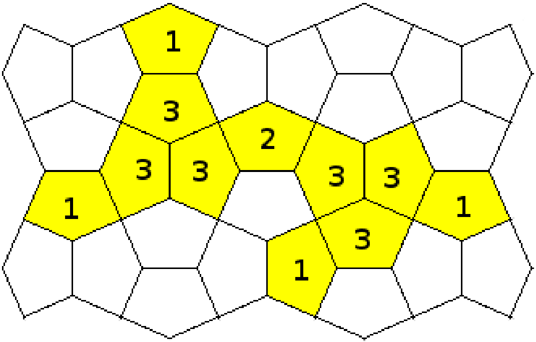
Naive check

- Delete each node and check if the remaining nodes are a corridor
- Time complexity: $\mathcal{O}((N \times M)^2)$ — too high

Better approach

- Check only critical nodes, i.e.
  - with degree 1
  - with degree ≥ 3 and their adjacents
- Bounded number:
  - at most 4 nodes with degree 1
  - at most 6 nodes with degree ≥ 3
- Time and space complexity: $\mathcal{O}(N \times M)$

G - Cairo Corridor



Sample corridor with 6 nodes of degree 3

E - Passwords

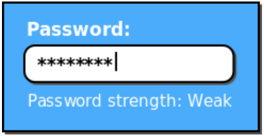
Problem

Given a blacklist with N words and two integers A and B, the task was to compute the number of different valid passwords obeying:

- length is between A and B (inclusive)
- at least one lowercase letter, one uppercase letter, and one digit
- no blacklisted substring.

Categories

- Dynamic Programming
- Strings, Aho-Corasick



E - Passwords

Overview

- Generating all passwords (obviously) would exceed time limit
- We can count... using dynamic programming (DP)
- Our DP state needs to have into account all restrictions
  - Size of the string *(easy)* - *integer*
  - Having > 1 upper/lower/digit *(easy)* - *3 booleans*
  - Avoiding blacklisted substrings *(not so easy...)*

Main Difficulty

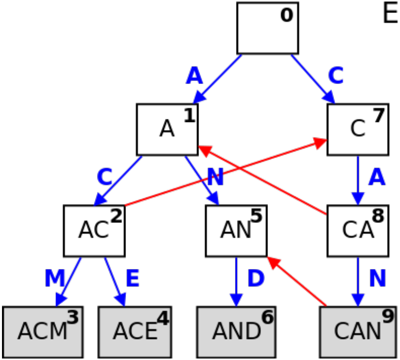
- How can we incorporate in the DP state how to avoid all blacklisted words at the same time?

E - Passwords

Idea - Use an automaton

Capture at which "position" we are at all blacklisted words

- We could use something like Aho-Corasick
- A "position" is a node in the automaton



Example for **{ACM, ACE, AND, CAN}**

→ Normal trie edge

→ "Fail" edge (largest proper suffix)

Example state change for: **ACAND**

$$0 \xrightarrow{A} 1 \xrightarrow{C} 2 \rightarrow 7 \xrightarrow{A} 8 \xrightarrow{N} 9 \rightarrow 5 \xrightarrow{D} 6$$

E - Passwords

Sample Solution

- Use a five dimensional DP state: (S, L, U, D, N)
  - S - Size of current password
  - L, U, D - Yes/No we have at least one upper/lower/digit
  - N - Position/Node in the automaton
- Let $\delta(N, c)$ be the transition from node N using character c
- $count(S, L, U, D, N) = 0$ if N is a word; otherwise:

$$\sum_{c \in \{a..z\}} count(S+1, true, U, D, \delta(N, c)) +$$
$$\sum_{c \in \{A..Z\}} count(S+1, L, true, D, \delta(N, lower(c))) +$$
$$\sum_{c \in \{0..9\}} count(S+1, L, U, true, \delta(N, leet(c)))$$

- Solution in count(0,false,false,false," ")
- Construction of automaton could be "inefficient"/not optimal

Caveat

Care should be taken with words that contain other smaller words

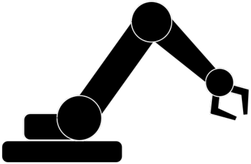
A - Within Arm's Reach

Problem

Given a robotic arm description and target coordinates, calculate a configuration that places the arm's tip as close as possible to the target.

Categories

- Geometry



A - Within Arm's Reach

Dificulties

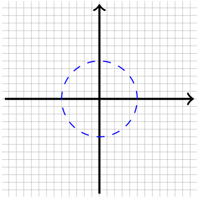
- Geometric
- Multiple output - need to construct any valid solution. Too much freedom. How to handle it?

Possible approach: understanding what are the reachable areas and how to step between them allows to construct a solution from target to origin.

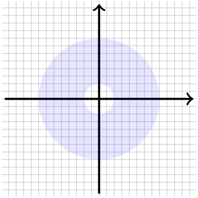
Useful geometric knowledge

- A polygon has every side smaller or equal to the sum of all other sides.
- Re-ordering segments does not affects reachable positions.
- Problem can be rotated to place target on an axis.

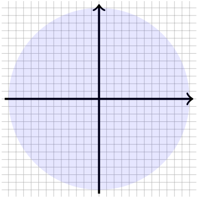
A - Within Arm's Reach



$L = \{5\}$
$r \in [5, 5]$



$L = \{5, 3\}$
$r \in [2, 8]$



$L = \{5, 3, 4\}$
$r \in [0, 12]$

Reachable area

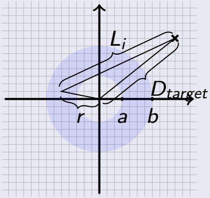
- The set of reachable points given a set L of segments is a single disc.
- One can also see r as a segment that can be added to L such that it is possible to construct a polygon.

A - Within Arm's Reach

Step function

To find a point in a disc $r \in [a, b]$ that is at distance L_i of a target:

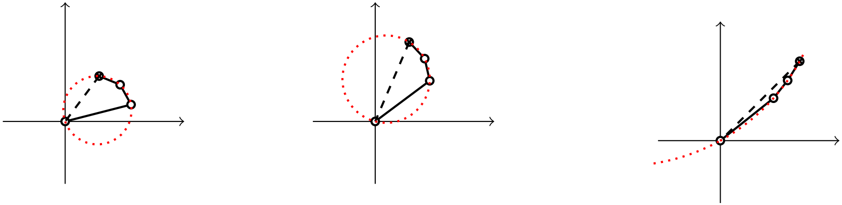
- Pick any r such that $\{r, L_i, D_{target}\}$ are the sides of a valid triangle
- From the triangle sides, calculate the angles and then the point.



Solution overview

- Calculate reachable disc when using every prefix of n segments.
- Construct solution from tip to origin.
- Print from origin to tip.

A - Within Arm's Reach



Other possible solution

- Interpret the problem as building a polygon with $N+2$ sides.
- The extra sides are the distance from the origin to target and the min side to make it possible to construct a polygon.
- Limit the solution space by trying to inscribe the polygon sides as cords on a circumference.
- Binary search on the circumference radius.

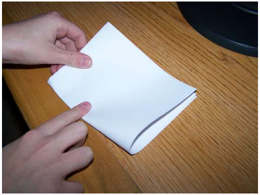
J - Risky Lottery

Problem

Approximate numerically the symmetric equilibrium strategy of the lowest unique bet game with N players choosing numbers from 1 to M.

Categories

Game Theory, Numerical Methods



J - Risky Lottery

Key Insight

In an optimal strategy, the probability of winning is the same regardless of the number we select.
(Otherwise, we should be picking numbers with higher probability of winning more frequently than we currently are.)

Example (3 players, 3 numbers):

$$\begin{cases} PWin_1(P) = \lambda \\ PWin_2(P) = \lambda \\ PWin_3(P) = \lambda \\ P_1 + P_2 + P_3 = 1 \end{cases} \Leftrightarrow \begin{cases} (P_2 + P_3)^2 = \lambda \\ P_1^2 + P_3^2 = \lambda \\ P_1^2 + P_2^2 = \lambda \\ P_1 + P_2 + P_3 = 1 \end{cases}$$

(Note: λ is not trivial because of ties.)

J - Risky Lottery

Solution sketch

Any iterative method that finds a solution should be accepted.

Even something *basic*:

- we only need 10^{-3} precision, so let's pick $EPS = 10^{-5}$;
- calculate $avg = \sum_i PWin_i(P)/M$;
- if $(PWin_i(P) > avg)$ then $P_i += EPS$ else $P_i -= EPS$;
- iterate until converged.

Questions?