Solution Outlines

Jury

GCPC 2014

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

acm International Collegiate Programming Contest

Algebraic Teamwork

Algebraic problem

Determine the number of non-idempotent permutations on some finite set.

Algebraic Teamwork

Algebraic problem

Determine the number of non-idempotent permutations on some finite set.

Intuition: Identity is the only idempotent permutation.

Algebraic Teamwork

Algebraic problem

Determine the number of non-idempotent permutations on some finite set.

Intuition: Identity is the only idempotent permutation.
Proof. Let $f : A \to A$ be idempotent & bijective:

$$f(a) \overset{\text{bijective}}{=} f^{-1} \circ f \circ f(a) \overset{\text{idempotent}}{=} f^{-1} \circ f(a) \overset{\text{bijective}}{=} \mathrm{id}(a)$$

Algebraic Teamwork

Algebraic problem

Determine the number of non-idempotent permutations on some finite set.

Intuition: Identity is the only idempotent permutation.
Proof. Let $f : A \to A$ be idempotent & bijective:

$$f(a) \overset{\text{bijective}}{=} f^{-1} \circ f \circ f(a) \overset{\text{idempotent}}{=} f^{-1} \circ f(a) \overset{\text{bijective}}{=} \mathrm{id}(a)$$

Solution

For input n, print $(n! - 1) \mod 10^9 + 9$.

(Substracting 1 without mod worked because of lucky constraints).

Beam me out!

- Evidently, the maze is a directed graph.
- If a cycle is reachable from room 1, you can get stuck if you are unlucky.
- Else the number of beaming operations is limited.
- $\Rightarrow$ Detect cycles with DFS.

Beam me out!

- Evidently, the maze is a directed graph.
- If a cycle is reachable from room 1, you can get stuck if you are unlucky.
- Else the number of beaming operations is limited.
- $\Rightarrow$ Detect cycles with DFS.
- If the graph contains (reachable) dead ends, we don't have a warranty that we reach the goal room.
- Else the probability is 100%.
- $\Rightarrow$ Use DFS on reversed graph.

Beam me out!

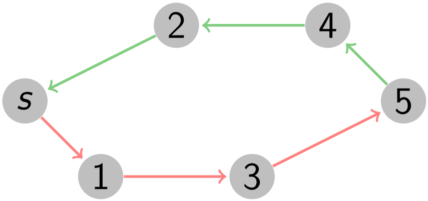
- Evidently, the maze is a directed graph.
- If a cycle is reachable from room 1, you can get stuck if you are unlucky.
- Else the number of beaming operations is limited.
- $\Rightarrow$ Detect cycles with DFS.
- If the graph contains (reachable) dead ends, we don't have a warranty that we reach the goal room.
- Else the probability is 100%.
- $\Rightarrow$ Use DFS on reversed graph.
- Combine information from both DFS runs.

Bounty Hunter

Problem: Traveling Salesman Problem

- in 2D Euclidean space
- additional bitonic restriction

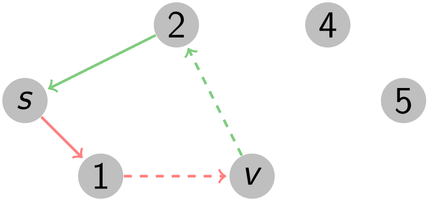
Bounty Hunter



Possible Solution

- partition trip into a left-to-right (LR) and a right-to-left (RL) path
- iterate over all places v from left to right
  - put v in the LR path or in the RL path
- use DP

Bounty Hunter



Possible Solution

- DP with state (v, last in LR, first in RL) is in $O(n^3)$.
- However v can be computed as $max(LR, RL) + 1$.
- $\Rightarrow O(n^2)$ solution

Connected Caves

- Input is a directed acyclic graph (DAG).

Connected Caves

- Input is a directed acyclic graph (DAG).
- Trying all paths would be exponential.

Connected Caves

- Input is a directed acyclic graph (DAG).
- Trying all paths would be exponential.
- However, the sub-result for trying all paths downwards a specific cave will not change.

Connected Caves

- Input is a directed acyclic graph (DAG).
- Trying all paths would be exponential.
- However, the sub-result for trying all paths downwards a specific cave will not change.
- $\Rightarrow$ DFS with Memoization

Connected Caves

- Input is a directed acyclic graph (DAG).
- Trying all paths would be exponential.
- However, the sub-result for trying all paths downwards a specific cave will not change.
- $\Rightarrow$ DFS with Memoization
- $\Rightarrow \mathcal{O}(N + E)$

Connected Caves

- Input is a directed acyclic graph (DAG).
- Trying all paths would be exponential.
- However, the sub-result for trying all paths downwards a specific cave will not change.
- $\Rightarrow$ DFS with Memoization
- $\Rightarrow \mathcal{O}(N + E)$
- Alternative solution: topologically sort caves + DP

Connected Caves

- Input is a directed acyclic graph (DAG).
- Trying all paths would be exponential.
- However, the sub-result for trying all paths downwards a specific cave will not change.
- $\Rightarrow$ DFS with Memoization
- $\Rightarrow \mathcal{O}(N + E)$
- Alternative solution: topologically sort caves + DP
- $\Rightarrow \mathcal{O}(N + E)$

Equator

Linear version i.e. without wrapping $\rightsquigarrow$ DP

$$p(i) = \begin{cases} 0 & \text{if } i < 0 \\ c_i + \max(0, p(i-1)) & \text{else} \end{cases} \Rightarrow \max_{-1 \leq i < n} p(i)$$

Equator

Linear version i.e. without wrapping $\rightsquigarrow$ DP

$$p(i) = \begin{cases} 0 & \text{if } i < 0 \\ c_i + \max(0, p(i-1)) & \text{else} \end{cases} \Rightarrow \max_{-1 \le i < n} p(i)$$

With wrapping

- Optimal solution does not wrap?
  $\rightsquigarrow$ Can be found by linear version.
- Optimal solution wraps?
  $\rightsquigarrow$ Find **minimal** interval of cities not to rob (identical DP) and subtract it from total c_i sum.

Gold Rush

- Math problem of the set

Gold Rush

- Math problem of the set
- Consider $a = \sum_{i=0}^{n} a_i 2^i$ in binary notation

Gold Rush

- Math problem of the set
- Consider $a = \sum_{i=0}^{n} a_i 2^i$ in binary notation
- As $a + b = 2^n$, the smallest non-zero index i' must be identical

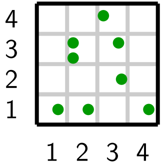
Gold Rush

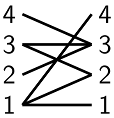
- Math problem of the set
- Consider $a = \sum_{i=0}^{n} a_i 2^i$ in binary notation
- As $a + b = 2^n$, the smallest non-zero index i' must be identical
- Counting downwards from n to i', the answer is $c := n - i'$

Gold Rush

- Math problem of the set
- Consider $a = \sum_{i=0}^{n} a_i 2^i$ in binary notation
- As $a + b = 2^n$, the smallest non-zero index i' must be identical
- Counting downwards from n to i', the answer is $c := n - i'$
- Running time $O(n)$ with step-by-step bisections of a until zero.

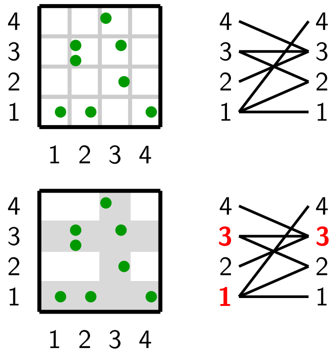
Jewelry Exhibition





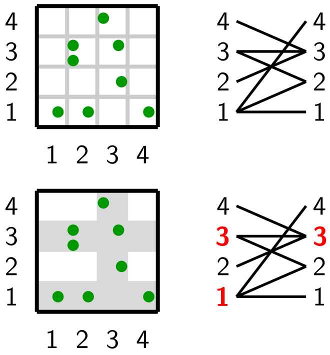
- Rows and columns form a bipartite graph.
- Exhibit at (x, y) represent an edge $\{a_{\lceil x \rceil}, b_{\lceil y \rceil}\}$.

Jewelry Exhibition



- Rows and columns form a bipartite graph.
- Exhibit at (x, y) represent an edge $\{a_{\lceil x \rceil}, b_{\lceil y \rceil}\}$.
- $\Rightarrow$ Find size of the minimum vertex cover.

Jewelry Exhibition



- Rows and columns form a bipartite graph.
- Exhibit at (x, y) represent an edge $\{a_{\lceil x \rceil}, b_{\lceil y \rceil}\}$.
- $\Rightarrow$ Find size of the minimum vertex cover.

- König-Egerváry Theorem: sizes of minimum vertex cover and maximum matching are equal in bipartite graphs.
- Solution: find maximum matching in $\mathcal{O}(N^2 M)$.

JuQueen

- Naive solution requires $C \cdot O$: $\sim 10^{11}$ operations

JuQueen

- Naive solution requires $C \cdot O$: $\sim 10^{11}$ operations
- Coord compression reduces C to $4 \cdot O$: $\sim 10^{10}$ ops

JuQueen

- Naive solution requires $C \cdot O$: $\sim 10^{11}$ operations
- Coord compression reduces C to $4 \cdot O$: $\sim 10^{10}$ ops
- $\Rightarrow$ Advanced data structure necessary
- Segment tree with lazy propagation
- Inner nodes represent intervals
- Store minimum, maximum and value in each node
- Propagate values only when necessary
- Answer queries in $\mathcal{O}(\log n)$

JuQueen

- Naive solution requires $C \cdot O$: $\sim 10^{11}$ operations
- Coord compression reduces C to $4 \cdot O$: $\sim 10^{10}$ ops
- $\Rightarrow$ Advanced data structure necessary
- Segment tree with lazy propagation
- Inner nodes represent intervals
- Store minimum, maximum and value in each node
- Propagate values only when necessary
- Answer queries in $\mathcal{O}(\log n)$
- (Even more efficient: combine both ideas)

JuQueen

- Naive solution requires $C \cdot O$: $\sim 10^{11}$ operations
- Coord compression reduces C to $4 \cdot O$: $\sim 10^{10}$ ops
- $\Rightarrow$ Advanced data structure necessary
- Segment tree with lazy propagation
- Inner nodes represent intervals
- Store minimum, maximum and value in each node
- Propagate values only when necessary
- Answer queries in $\mathcal{O}(\log n)$
- (Even more efficient: combine both ideas)
- Don't use billions of objects in Java – your Garbage Collector will go crazy!

Laser Cutting

- Basic geometry problem
- Required algorithms:
  - Intersection of two lines
  - Point in polygon
- First condition: Every line of a polyline may only intersect with the next and the previous line of that polyline.
- Second condition: For two different polylines, any line from the first may not intersect with any line from the second.

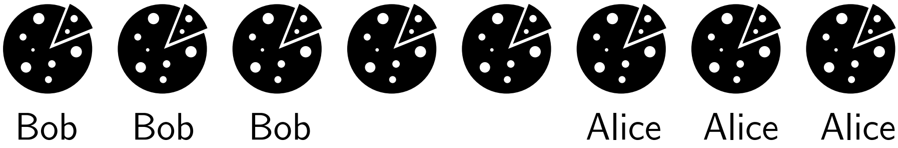
Laser Cutting

- Third condition:
  - For any two polygons, check whether one contains the other.
  - To do so, check whether an arbitray boundary point of the one polygon is in the other polygon.
  - As polygons do not touch, this is sufficient.
  - If any polygon is inside two other polygons, the condition is failed.
- To use (in Java):
  - `java.awt.geom.Line2D.linesIntersect`
  - `java.awt.geom.Path2D.contains`

Not a subsequence

- Focus on length of shortest non-subsequence (counting them is done similarly).
- Start with DP in $\mathcal{O}(nk)$ [$n =$ string len, $k =$ alphabet size]
  - For every suffix $s[i..n]$ compute length of shortest non-subsequence $T[i]$.
  - Define $f_a(i+1)$ as leftmost occurrence of a in $s[i+1..n]$
  - $\Rightarrow$ Minimize $1 + T[f_a(i+1)]$ over all possible chars c
- Improve to $\mathcal{O}(n)$:
  - Notice that $T[f_a(i+1)]$ is either x or $x+1$.
  - Of course we prefer x (when all are $x+1$, increment x).
  - Keep track of where x's are, and how many of them are still there.
  - This can be all done in $\mathcal{O}(1)$ time per i.

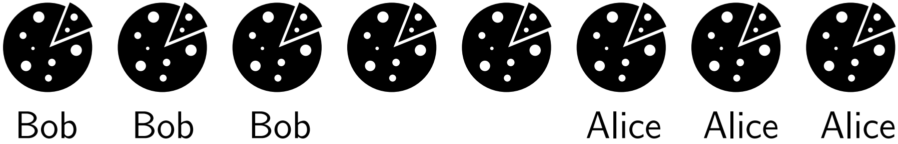
Pizza Voting



- Alice will veto last third
- Bob will veto first third
- You can choose any in the middle third
- Rounding at the border of thirds

pizza icon by `http://www.danilodemarco.com/`

Pizza Voting



- Alice will veto last third
- Bob will veto first third
- You can choose any in the middle third
- Rounding at the border of thirds
- `puts((i > n/3 && i <= n-(n+1)/3) ? "YES" : "NO");`

pizza icon by `http://www.danilodemarco.com/`