Solutions

Preliminaries BAPC 2017

University of Amsterdam

September 2017

Abandoned Animal (1)

Benelux Algorithm Programming Contest 2017

- Given is an ordered list of items, and every item has a list of numbers (i.e. the stores where they can be bought).
- Can we choose a single number in this list for every item such that the list is non-decreasing. Also, can this be done uniquely?
- The first question is relatively easy: starting from the front, greedily adhere the lowest number to each item that is at least the previous number.
- To determine whether it is unique, there is an elegant solution: repeat the process, but start from the rear, and greedily choose the highest number. If you get the same path as in the previous step, the answer is `unique`, otherwise `ambiguous`.

Abandoned Animal (2)

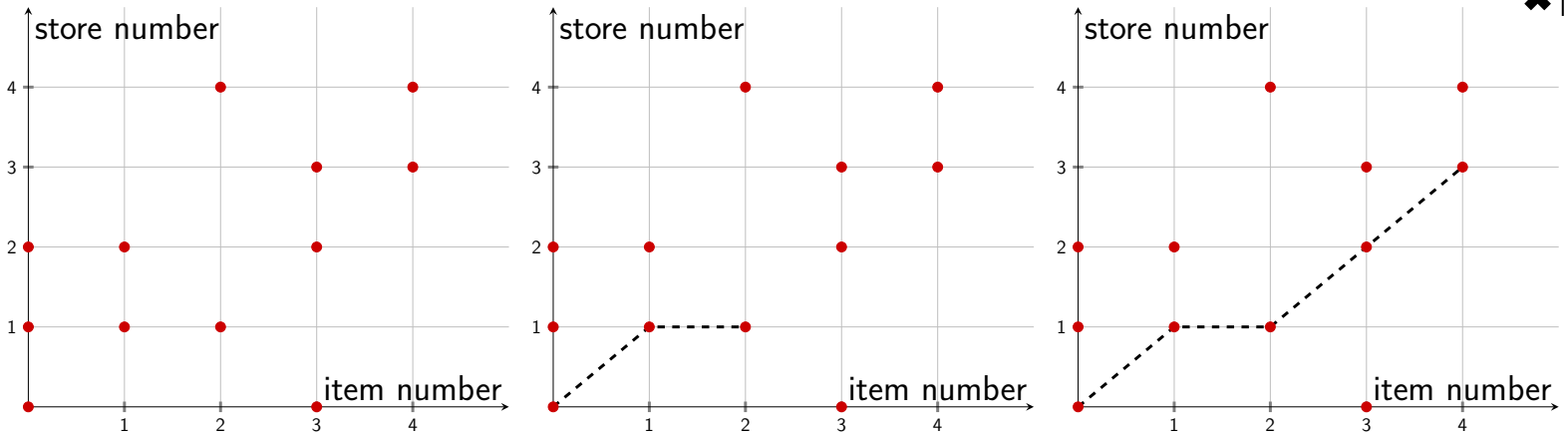
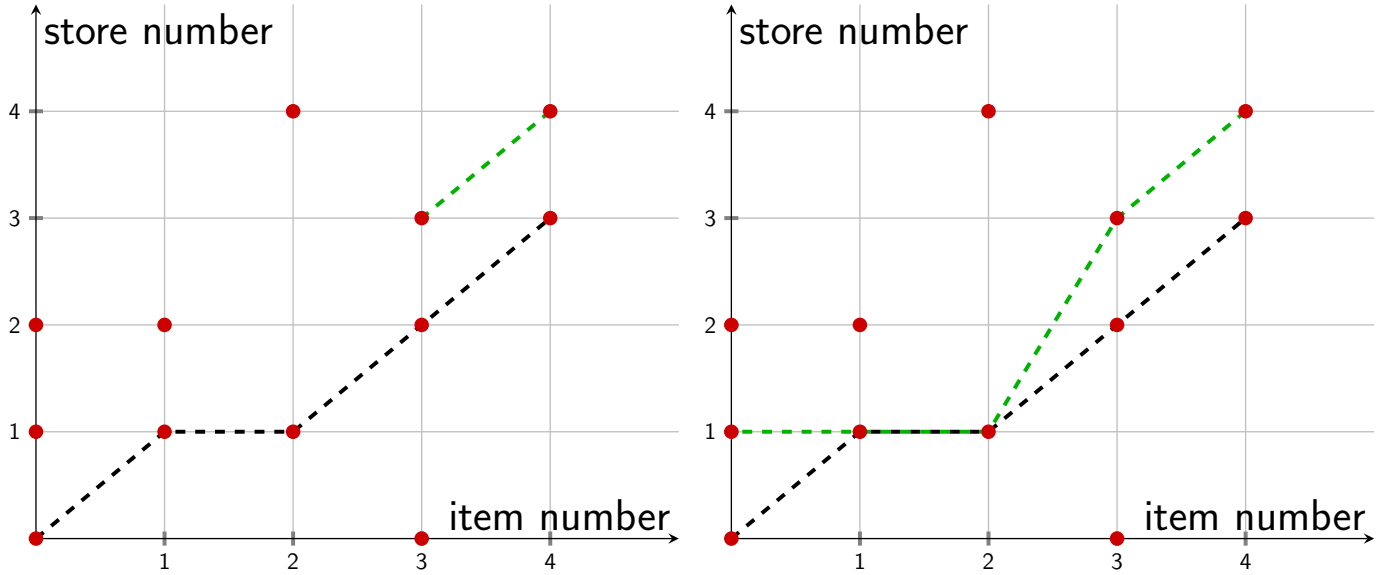
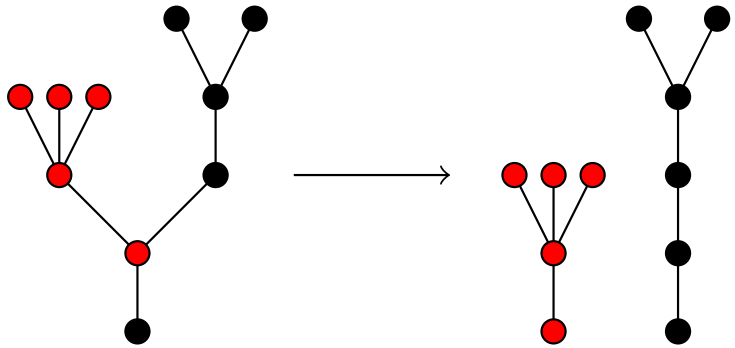


Figure: Finding the lower path.



Booming Business (1)

- Want to count Bonsai trees with w edges and height h.
- Let $T_{n,k}$ be the number with n edges and height at most k. Then we want $T_{w,h} - T_{w,h-1}$.
- We can split each tree in the left subtree and the rest of the tree.



- So we write $T_{n,k}$ as $T_{i,k-1} \times T_{n-i,k}$. This gives

$$T_{n,k} = \sum_{i=1}^{n-1} T_{i,k-1} \times T_{n-i,k}.$$

- Now set up the base cases and use dynamic programming.

Booming Business (2)

- This solution is a variant of counting rooted trees without height restriction using Catalan numbers.
- It runs in $\mathcal{O}(hw^2)$, which is fast enough.
- Other solutions include:
  - Counting by removing the rightmost edge from the tree.
  - Counting by considering the outline of a tree (up, up, down, up, ...).
- These can even be done in $\mathcal{O}(hw)$.

Crowd Control

- Find path with max capacity and remove adjacent edges.
- Multiple ways to solve, e.g.
  - Find max spanning tree and then perform a breadth-first on the tree to filter the edges;
  - Perform a variant of Dijkstra where the optimal path is defined by the capacity.
- Be careful that the output does not mention an edge to remove multiple times!

Disastrous Doubling

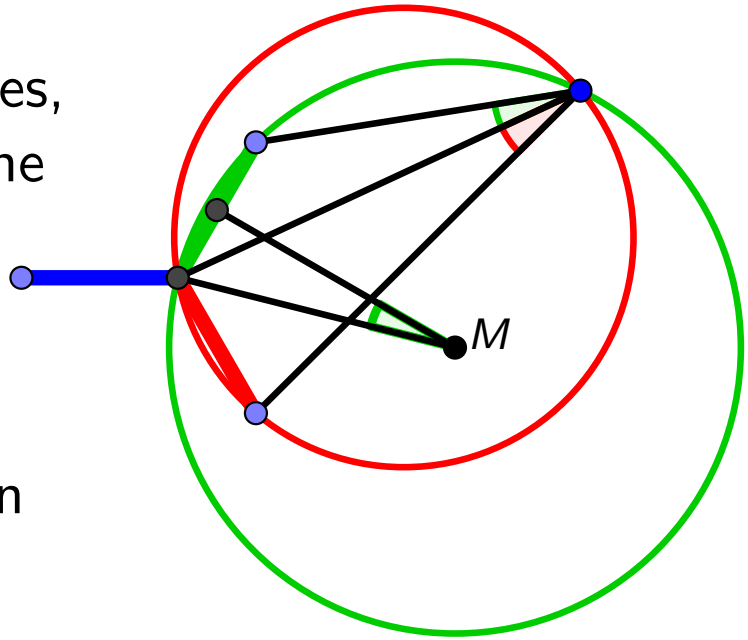
- Double number of bacteria for each hour.
- Start modulo when number of bacteria $\geq$ maximum used in one experiment.
- Make sure that you don't get negative results.

Envious Exponents

- Given N, k, find the smallest $M > N$ such that the binary representation of M contains exactly k 1s.
- Many valid approaches. For example: let $A = N + 1$, and let b be the number of 1s in the binary representation of A:
  - If $b \leq k$, we simply flip the $k - b$ least significant 0s of A.
  - If $b > k$, find the kth most significant 1 at some position i. We flip all less significant 1s of A, and add 2^i to A. Then go to the above step.
- This solution may seem arcane, but you can arrive at this or a similar solution pretty easily by reasoning about what happens when you just keep incrementing N by 1 until it has exactly k 1s.

Fidget Spinner

- Exactly two colours will be fully visible.
- We can calculate the angle both these colours span.
- By the inscribed-angle theorem for circles, we know that the camera must lie on the intersection of two circles.
- We can easily find the centres of these circles.
- Find a linear formula for the intersection point, using that they intersect in the origin (or do a circle-circle intersection).
- Edge case when the circles coincide.

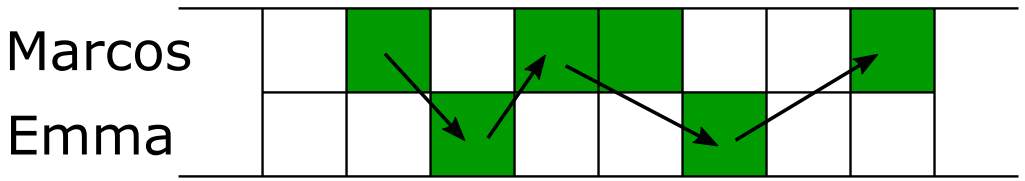


Ghostbusters

- The keyboard is a bipartite graph (V, E) where rows and columns are vertices V and signals define edges E.
- Probability that a key is pressed is < 0.5 so the most likely set of pressed keys will have minimal size.
- Solution is one of the most likely subsets of edges E' such that (V, E') has the same connected components as (V, E).
- Compute a maximum spanning tree of the graph and output the keys in lexicographical order.

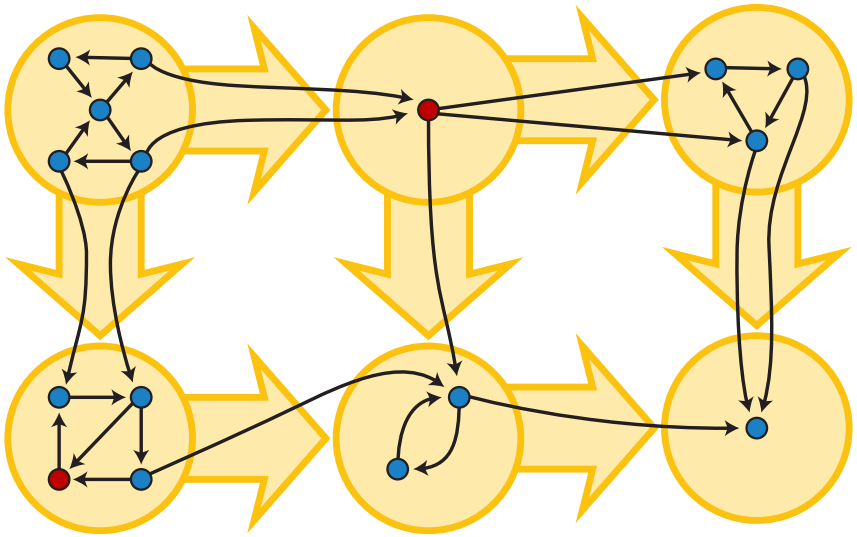
Horror Film Night

- Find the maximum number of films Emma and Marcos can watch together.
- A greedy solution works.
- Go through the films liked by at least one person in sorted order.
- Watch the current film whenever possible.
- Do not forget to sort the films first!



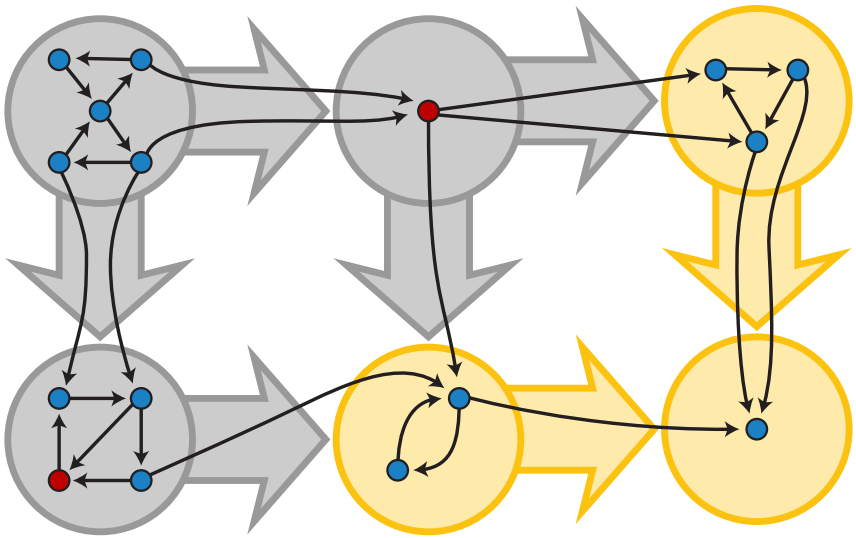
Intelligence Infection (1)

- Divide the graph of spies into groups of spies that can all reach each other.
- This can be done by finding all the strongly connected components (SCCs)
  - Naive algorithm: $\mathcal{O}(n^2)$ - too slow
  - Kosaraju's or Tarjan's algorithm: $\mathcal{O}(n)$ - fast enough
- Form the condensation graph of the SCCs.



Intelligence Infection (2)

- An SCC is *dirty* if:
  - One of the spies in the SCC is an enemy spy
  - Another *dirty* SCC can reach this SCC - use DFS
- We need to privately message all non-enemy spies in a *dirty* SCC
- Next, to find out who we need to message publicly:
  - Remove all the *dirty* SCCs from the condensation graph
  - Publicly message exactly one spy in an SCC that has indegree zero in the resulting condensation graph.



Journal Editing

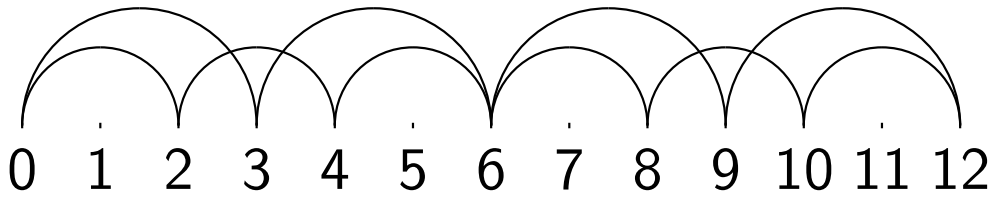
- Find a correct proof of the main theorem.
- Dynamic programming over all subsets of theorems:
- For each subset S, sorted by size, if we can prove the theorems in this set:
  - Loop over all unproven theorems T (not in S)
  - If we can prove the new theorem T using theorems in S, update the minimal length of proving $S \cup \{T\}$.
- Find the set S with $T_0 \in S$ with the shortest proof.
- Beware that cyclic dependencies are not allowed!
- Runtime is $\mathcal{O}(\text{poly}(n)2^n)$.

Knight Marathon

- Distance between the knight and the capital can be very large for a search algorithm.
- Most of the moves will consist of just two types of moves, e.g. $(+1, +2)$ and $(+2, +1)$.
- We can check in constant time if two squares are connected using only moves of these two types. In this case we can also compute the number of steps needed to reach one from the other in constant time.
- There are always squares close to the capital such that the knight's starting location is connected to them in this way. Use a search algorithm for squares near the capital and compute the minimum total number of moves.

Leapfrog (1)

- We want to figure out the largest group of frogs that will get together at the smallest position.
- Since every jump distance d is prime, all frogs with different jump distances can reach each other.
- We can group frogs by their jump distance d and start position modulo jump distance $[x \mod d]$, since every frog in group $[x \mod d]$ can reach each other.
- When two groups $[x_1 \mod d]$ and $[x_2 \mod d]$ both have the largest size for their jump distance d, both groups are valid for the largest tower.
- Every combination of groups $[x_1 \mod d_1, x_2 \mod d_2, \ldots, x_n \mod d_n]$ will have their own unique smallest position they can reach.



Leapfrog (2)

- Check every possible combination of viable groups $[x_1 \mod d_1, x_2 \mod d_2, \ldots, x_n \mod d_n]$. A reasonably efficient enumeration is needed.
- Determine for the combination the smallest position where the frogs get together.
  - By using the Chinese Remainder Theorem
  - With a very efficient merging strategy, where you combine groups together one by one.
  - You can not do this by simply checking all possible positions.
- You can prune combinations:
  - By using starting positions as a lower bound for the smallest position, comparing with your best known answer.
  - By using intermediate results from the Chinese Remainder Theorem.
- Overflow can be avoided when carefull!