Problems

NWERC 13

Delft University Of Technology

Absurdistan Roads (1/2)

NWERC 13
Delft University Of Technology

Problem

Given shortest distances in a connected graph with N vertices and edges, reconstruct the original graph.

Solution

- Let H be the graph given by the shortest distances.
- Minimum spanning tree T of H is part of the solution.
  - Take a set S and its complement $\overline{S}$.
  - The shortest distance between two vertices in S and $\overline{S}$ must be realized over one edge.
  - Start with $S = \{1\}$, at each step add the shortest edge between S and $\overline{S}$ – this is Prim's algorithm.
- ...

Absurdistan Roads (2/2)

Solution

- ...
- Spanning tree T has $N - 1$ edges, so we need one more edge.
- Find the shortest distance in H that is not possible using T, add it as an edge.
- Total time complexity is $\mathcal{O}(N^2)$.

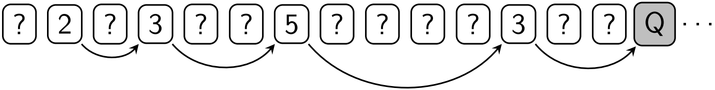
Battle for Silver

Find all cliques in given graph: Clique Problem (NP-Complete). Note that graph is *planar*! Kuratowski"s theorem: *Maximum clique size is 4*. Therefore, naïve approach suffices:

Naïve approach

- Find all cliques of size 2 (given by all edges);
- Find all cliques of size 3;
- Find All cliques of size 4;
- Output loot of clique that provides highest loot (*not from the largest clique*)!

Card Trick



Naïve approach

- Try X random card configurations and calculate final card
- Depending on X: either too slow or incorrect, probably both

Solution

- Insight: for every known card, the probability is 1
- For every unknown card, try every possibility
- Speed up with dynamic programming

NWERC 13 Delft University Of Technology

Diagrams & Tableaux (1/2)

Problem

- Given a diagram, count the corresponding tableaux
- Given $\begin{array}{|c|c|}\hline \ & \ \\ \hline \ & \multicolumn{1}{c}{} \\ \cline{1-1} \end{array}$, count
  $\begin{array}{|c|c|}\hline 1 & 1 \\ \hline 2 & \multicolumn{1}{c}{} \\ \cline{1-1} \end{array}$ $\begin{array}{|c|c|}\hline 1 & 1 \\ \hline 3 & \multicolumn{1}{c}{} \\ \cline{1-1} \end{array}$ $\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 2 & \multicolumn{1}{c}{} \\ \cline{1-1} \end{array}$ $\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & \multicolumn{1}{c}{} \\ \cline{1-1} \end{array}$ $\begin{array}{|c|c|}\hline 1 & 3 \\ \hline 2 & \multicolumn{1}{c}{} \\ \cline{1-1} \end{array}$ $\begin{array}{|c|c|}\hline 1 & 3 \\ \hline 3 & \multicolumn{1}{c}{} \\ \cline{1-1} \end{array}$ $\begin{array}{|c|c|}\hline 2 & 2 \\ \hline 3 & \multicolumn{1}{c}{} \\ \cline{1-1} \end{array}$ $\begin{array}{|c|c|}\hline 2 & 3 \\ \hline 3 & \multicolumn{1}{c}{} \\ \cline{1-1} \end{array}$

Diagrams & Tableaux (2/2)

NWERC 13
Delft University Of Technology

Solution

- Maximum output: 27 million
- Backtracking: top to bottom, bottom to top, left to right, right to left
- Don't forget to prune the search tree: e.g. never put a number $< k$ in the kth row from the top when going from bottom to top
- Dynamic programming: e.g. over the number of different columns; a columns of height H admits $\binom{N}{H}$ different labelings
- There even exists a closed formula takes $\mathcal{O}(N^2)$ steps to evaluate

Exponential Towers (1/4)

Some preliminaries

- Let $n = \prod p_i^{n_i}$, how many ways to write n as a power?
- let $g = gcd(n_i)$, and let $t = \prod p_i^{n_i/g}$.
- Then $n = t^g$.
- Every decomposition $g = u * v$ gives rise to a representation of n as a power (and vice versa):

$$n = t^{u*v} = (t^u)^v$$

Exponential Towers (2/4)

And now for the problem

- Let $n = a^{b^c}$, let g be the gcd of the exponents of a, and let $a = x^g$,
- then $n = x^{g*b^c}$, let $B = g * b^c$, so $n = x^B$. Forget about x.
- B can be huge, but its prime decomposition is easily obtained: $B = \prod p_k^{B_k}$.

Exponential Towers (3/4)

Algorithm

- $n = x^B$, then every decomposition $B = u * v^w (v, w > 1)$ gives a representation $n = (x^u)^{v^w}$, or more representations, if w can be written as a power, or even a tower of powers.
- For every $w > 1, w \leq max(B_k)$
- for every prime p_k count the decompositions $p_k^{B_k} = p_k^{u_k} * p_k^{w*v_k}$, so $B_k = u_k + w * v_k$. So we have to count the number of multiples of w up to B_k (including 0), and this equals $[B_k/w] + 1$.
- ...

Exponential Towers (4/4)

Algorithm

- ...
- The number of representations for this w equals $\prod([B_k/w]+1)$, $v=1$ is not allowed, however, so the actual count for this w is: $\prod([B_k/w]+1)-1$.
- multiply with the number of ways to write w as a tower of powers (of height ≥ 1). The algorithm was given above, but for each representation: $w = r^s$ we have to count (recursively) the number of representations of s.
- Sum over all $w > 1$, $w \leq max(B_k)$.

First Date (1/2)

The Hard Way: $\mathcal{O}(1)$, but complicated

- Implement `JulianDateToDayNr()` and `DayNrToGregorianDate()`.
- Calculate `DayNrToGregorianDate(JulianDateToDayNr(Y, M, D) + 1)`.
- However, `DayNrToGregorianDate()` is quite complex!
- You can avoid much of the complexity by using the standard library: `GregorianDate` class in Java, or `gmtime()` function in C/C++.

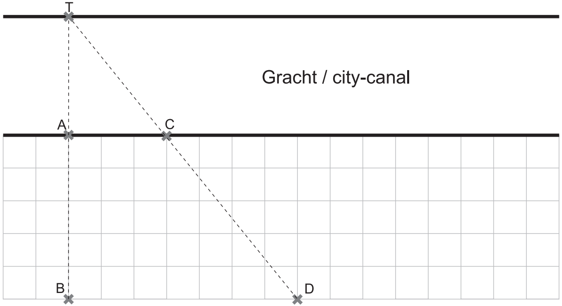
First Date (2/2)

The Easy Way: $\mathcal{O}(1)$ by building a LUT

- Walk all dates in the interval (approx. 3 million).
- Keep track of the Julian date and the Gregorian date of the next day; build a lookup table.
- You only need to implement a 'proceed-to-next-day' for the Julian and Gregorian calendars, which is easy.

Other algorithms are also possible, e.g. careful bookkeeping of the number of skipped dates.

Grachten



Use intercept theorem

- $AC : AT = BD : (TA + AB)$
- Solve for $TA \Rightarrow TA = \frac{AB \cdot AC}{BD - AC}$
- Compute greatest common divisor to reduce the fraction

Highway of the Future (1/2)

Problem

Given a collection of line segments, find the number of line segments going through the point in which the maximum number of line segments intersect.

Considerations

- Car speed (integer, $1 \leq v \leq 100$), integer start times, and length of highway (100) severely limit the amount of possible collisions over a large collection of cars.
- Collisions are not always on integer coordinates.

Highway of the Future (2/2)

Solution

- For each car: consider all cars which arrive earlier than this car but depart later. These are precisely the cars that it passes on the highway.
- One list ordered by arrival, and one list ordered by departure.
- There are other ways to consider only the necessary cars.
- Some pitfalls for example:
  - Two identical cars always require at least two lanes.
  - Intersections that are not on the highway ($x < 0$ or $x > 100$).

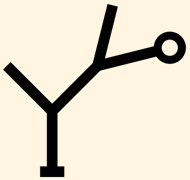
Infix to Prefix

- Create a lookup table.
- For each substring of the input calculate the maximal and minimal possible value (if this substring represents a subexpression). Short strings first.
- The values for the substring that starts at position x and has length l are stored in an array, at position (x, l).
- These values then are used to calculate the values for longer strings, using these rules:
  - max(x+y) = max(x) + max(y)
  - max(x - y) = max(x) - min(y)
  - max(-x) = - min(x)
  - and equivalent rules for min.
- If n is the length of the string, the array to be filled has about $n * n/2$ elements.

Jingle Balls (1/2)

Extended problem

- Given a decorated tree T and integer K, determine whether it is possible to end up with exactly K balls in T such that the balls are balanced.
- Determine how many balls must be moved within T or brought into T.



Recursive solution

- If K is even, put $K/2$ balls in left subtree and $K/2$ balls right.
- If K is odd, try both ways: $(K+1)/2$ left and $(K-1)/2$ right, or the other way around.

Jingle Balls (2/2)

Time complexity

- $B =$ total number of balls
- Depth of the tree is bound by $\log_2(B)$
- Time complexity for recursive search $\mathcal{O}(B^{1.69})$

Linear time algorithm

- Any subtree needs to consider at most two different values K.
- Subtrees at depth d of the tree:
  consider $K = \lfloor B/2^d \rfloor$ and $K = \lceil B/2^d \rceil$
- Or use memoization.