Delft University Of Technology

NWERC 2012 Problem Set

Discussion & Solutions

2012.nwerc.eu

Delft University of Technology

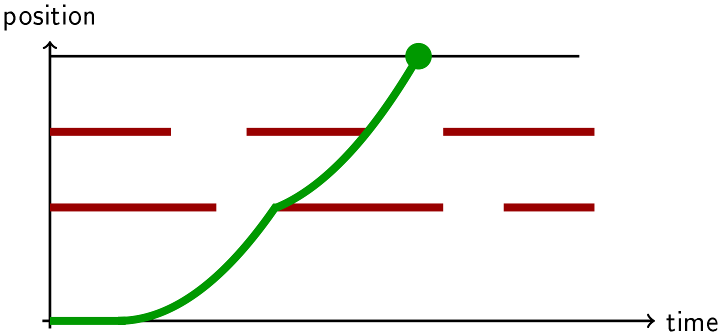
A – Admiral

- Two vertex-distinct paths, minimum cost.
  1. Suurballe, or
  2. network flow.
- Expand vertices: $v_k \Rightarrow v_k^{in} \rightarrow v_k^{out}$.
- $\mathcal{O}(|E|)$ time for each (augmenting) path.

B – Beer Pressure

- Random walk over n-dimensional lattice.
- Iterate over all possible end states after voting.
- Find corresponding probabilities:
  (Dynamic Programming or Memoization)
- Accumulate probabilities for all the pubs
- If w is the number of non-dominate students, this algorithm is $\mathcal{O}\left(\frac{w^n}{n!}\right)$.
- Smarter algorithms exist (e.g., closed form of probability of end states) but dynamic programming is sufficient.

C – Cycling



- Only brake/wait immediately after a traffic light; otherwise max acceleration.
- For each traffic light, for each time it changes color, find the highest velocity that you can have there.
- Worst case: $(10 \text{ lights} \times 580 \text{ changes})^2 = 33\,640\,000$

D – Digital Clock

- Solution: **Just try all possible answers!**
- For *start time* from 00:00 to 23:59 :

  $t \leftarrow$ *start time*

  For each input pattern p :

  For all 28 segments s :

  If segment s active in p : mark s as *working*

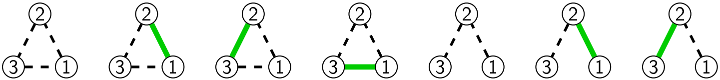
  If p differs from t wrt s : mark s as *broken*

  $t \leftarrow t + 1$

  If no segment is both *working* and *broken* :

  Print answer *start time*
- Worst case $= 24 \times 60 \times 50 \times 28 = 2\,016\,000$ steps

E – Edge Case



- Count the number of sets of edges not containing any pair of adjacent edges.
- First, solve the problem for a linear graph of n nodes
  - F_2=2 ($\emptyset$, $\{(1\ 2)\}$)
  - F_3=3 ($\emptyset$, $\{(1\ 2)\}$, $\{(2\ 3)\}$)
  - $F_n = F_{n-1} + F_{n-2}$
- Now extend this to a circular graph of n nodes
  - Either do not use edge (n,1): F_n sets
  - Or do use edge (n,1): another F_{n-2} sets
- $L_n = F_n + F_{n-2} = L_{n-1} + L_{n-2}$
- L_n = Lucas numbers

F – Foul Play

Problem

- Given $n = 2^k$ teams, and a team 1 such that
  - for every team t that 1 cannot beat, 1 can beat a team t' that can beat t, and
  - 1 can beat at least half of the teams.
- Give a tournament where 1 wins.

Solution

- For every round (k in total) of the tournament:
  1. Match as many teams t that beat 1 to teams t' that can beat them, i.e., t
  2. Let 1 play against a team from which it can win.
  3. Let any remaining teams play against each other.
  4. Remove the losers.

G – Guards (1/2)

Problem

- Given graph, place minimum number of guards k at vertices (halls) to monitor edges (corridors).
- Instance of the NP-complete (minimum) vertex cover problem.

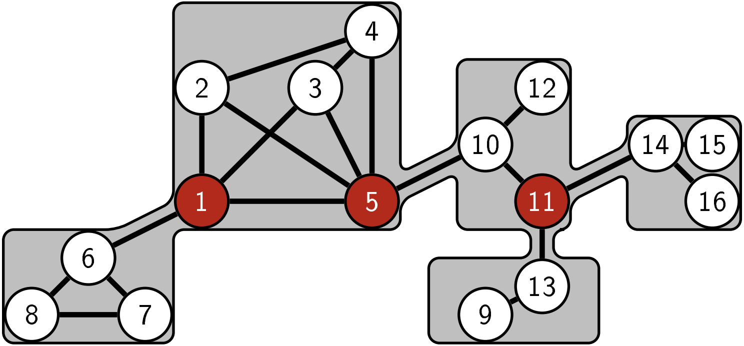
Solution for general graphs

- Search tree: branch on edges (u, v): select (and remove) u or v: $\mathcal{O}\left(2^k \cdot (n + m)\right)$.
- However k may be close to $n \leq 10000$.

G – Guards (2/2)

Use recursive structure: $\mathcal{O}\left(2^{10} \cdot (n+m)\right)$

- Compute recursively two solutions for each peripheral building: with and without a guard in the hall leading to it.
- Use search tree to compute optimal solution for main building using these peripheral solutions.



H – Hip to be Square (1/2)

- Given an interval, find the subset with the least square product. Terrible complexity: $[|S| \leq 4899, 2^{4899}$ subsets$]$
- If the interval S contains a square, return the least square. $[|S| \leq 138, 2^{138}$ subsets, but size $\leq 12]$
- Reduce: Split each number in a square and primes:
  $n = a^2 * p_1 * p_2 \ldots$
  Remove numbers containing a unique prime. Cascading.
  If a prime occurs twice: combine.
  $[|S| \leq 30, 2^{30}$ subsets, $\binom{30}{12} \cong 10^7$ $]$.

H – Hip to be Square (2/2)

- Make a priority queue Q, of subsets (products) ordered by value (and by size!). Invariant: If s is a square product of S, then Q contains a divisor d of s.
- Put the elements of S as singletons in Q.
- While $m =$ Q.dequeue is not a square:
  choose a prime p that divides m.
  multiply m with each n in S such that:
  p divides n with odd exponent
  n is not a member of the product m.
  (keep track of the factors of the elements of Q.)
  add all these products to Q. This keeps the invariant.
- If m is a square, it is the least square product.

I – Idol

- 2SAT:
  - variable x_i: contestant i advances
  - judges are clauses:
  - "-1 4" corresponds to $\neg x_1 \vee x_4$.
- Choosing $x_1 = 1$ (in ex.), forces $x_4 = 1$.
- Propagation leads to:
  - contradiction, or
  - finding an independently satisfiable subset of clauses:
    - $l_1 \vee l_2$: untouched
    - $(l_1 = 1) \vee l_2$: satisfied
    - $l_1 \vee (l_2 = 1)$: satisfied
    - $(l_1 = 0) \vee l_2$: can propagate
    - $l_1 \vee (l_2 = 0)$: can propagate
    - $(l_1 = 0) \vee (l_2 = 0)$: "no"
- Very limited backtracking: $\mathcal{O}(n \cdot m)$.

J – Joint Venture

- Naive $\Theta(n^2)$ algorithm is too slow.
- First, sort Lego pieces.
- Then, either
  - keep two indices at resp. end of sorted array. Update indices depending on if their sum is too large or too small when compared to x.
  - or, for each Lego piece, binary search for the difference between x and length of current piece.
- Gives $O(n \log n)$ behaviour.

K – Key Insight

- Find out to which positions a letter may have moved.
- Which of n^2 transpositions is consistent with all blocks.
- Take factorial per letter.
- Careful consistency checking, e.g.
  - Watch out for missing letters.
  - Take into account all blocks.
- Accepted solutions: $\mathcal{O}\left(n^3\right)$

NWERC 12
Delft University Of Technology