Benelux Algorithm Programming Contest 2016

Solutions

BAPC 2016

Delft University of Technology

22 October 2016

A: Airport Logistics [1/3]

Benelux Algorithm Programming Contest 2016

The solution for this problem has two parts:

1. Create a directed graph, with nodes representing points on the floor and cost-labeled edges representing the time to walk from one point to another,
2. find the shortest path in that graph.

Part 2 can be done using Dijkstra's algorithm.

Part 1 is the hard part.

A: Airport Logistics [2/3]

Doing some geometry, we find the following rules:

- The optimal path consists of straight line segments.
- When an optimal path joins a conveyor halfway (i.e. not at the begin of the conveyor), this conveyor is approached via a straight line intercepting the conveyor at a 60-degree angle.
- When an optimal path leaves a conveyor halfway (i.e. not at the end of the conveyor), the path leaves the conveyor via a straight line at a 60-degree angle with the conveyor.
- It is never necessary to leave one conveyor halfway and join the next conveyor halfway.

A: Airport Logistics [3/3]

Benelux Algorithm Programming Contest 2016

According to these rules we connect:

- the starting point with each belt,
- each belt to the end point,
- each pair of belts,
- (finally) the nodes within each belt - going from entrance nodes to exit nodes.

This graph has $O(N^2)$ nodes and $O(N^2)$ edges in the worst case. The shortest path in the graph is then found with Dijkstra's algorithm in time $O(E * log(E))$.

B: Battle Simulation

Benelux Algorithm Programming Contest 2016

Problem: replace characters in given string S. When subsequent combination occurs of all 3 characters, replace those 3 with one character instead.

B: Battle Simulation

Benelux Algorithm Programming Contest 2016

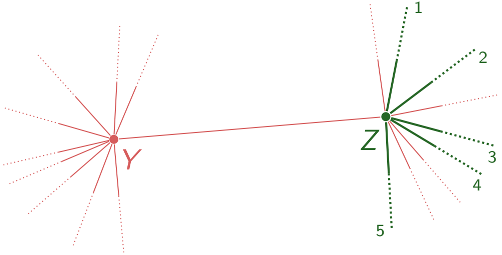
Problem: replace characters in given string S. When subsequent combination occurs of all 3 characters, replace those 3 with one character instead.

- Linearly replace all characters. Lookahead two characters to see if any combination occurs of three different characters.
- If so, ignore following two characters and continue replacing characters.
- Or... Use regular expressions instead! E.g. S.replaceAll("RBL—RLB—BRL—BLR—LRB—LBR", "C");
- String concatenation is too slow.

C: Brexit [1/2]

Benelux Algorithm Programming Contest 2016

- Simulation with some emphasis on efficiency.
- Look locally: when removing a country Y, see if this pushes one of its partners Z over the tipping point.
- Don't perform a recount every time we consider Z:

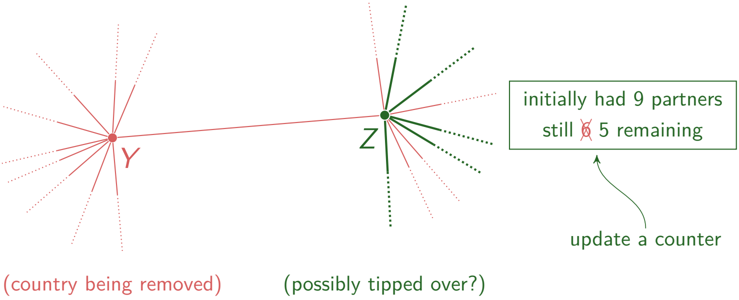


(country being removed) (possibly tipped over?)

Complexity can be up to $\Theta(P^2)$, which is too slow!

C: Brexit [2/2]

- Instead we keep count:



- Can be implemented breadth-first or depth-first.
- Time complexity: $\mathcal{O}(C + P)$.

D: Bridge Automation [1/2]

Task:

- No boat may wait more than 1800 seconds.
- Minimize amount of time where bridge is not fully closed.

Strategy:

- Keep bridge closed until oldest boat has waited (1800 - 60) seconds.
- Then open bridge, let the next k boats through, then close it.
- Repeat until all boats passed.

D: Bridge Automation [2/2]

Benelux Algorithm Programming Contest 2016

Algorithm: dynamic programming

table$[p]$ = minimum cost needed to let the first p boats pass

table$[0] = 0$

table$[p] =$

$$\min_{1 \le k \le p} \left(\text{table}[p-k] + \max\{ T_p - T_{p-k+1} - 1800 + 20, 20k \} + 120 \right)$$

(Assume first $(p-k)$ boats already passed;
let boat $(p-k+1)$ wait exactly 1800 seconds, then open bridge;
keep bridge open until boat (p) has passed, then close bridge.)

Final answer is table$[n]$

E: Charles in Charge [1/2]

Problem: given a graph G, find the lowest value such that the shortest path using only edges of length at most this lowest value is at most $X\%$ longer than the shortest path without any limitations.

E: Charles in Charge [1/2]

Benelux Algorithm Programming Contest 2016

Problem: given a graph G, find the lowest value such that the shortest path using only edges of length at most this lowest value is at most $X\%$ longer than the shortest path without any limitations.

Some notation:

- let $G = (V, E)$ be the given graph and let D be the maximum distance Charles is allowed to travel;
- for a value K, let $G_k = (V, E_K)$ be the subgraph of G using only edges of length at most K;
- let D_K be the shortest distance from 1 to N in G_K.

E: Charles in Charge [2/2]

The problem is now formulated as follows: what is the smallest K such that the shortest path from 1 to N in G_K is at most D?

E: Charles in Charge [2/2]

Benelux Algorithm Programming Contest 2016

The problem is now formulated as follows: what is the smallest K such that the shortest path from 1 to N in G_K is at most D?

We make a pair of observations:

1. Given a value K, we can calculate the shortest path from 1 to N in G_K using Dijkstra.
2. For any value $L \geq K$ we have $D_L \leq D_K$.

E: Charles in Charge [2/2]

The problem is now formulated as follows: what is the smallest K such that the shortest path from 1 to N in G_K is at most D?

We make a pair of observations:

1. Given a value K, we can calculate the shortest path from 1 to N in G_K using Dijkstra.
2. For any value $L \geq K$ we have $D_L \leq D_K$.

Hence we can use *binary search* to find the correct value of K and solve the problem.
Runtime: $O(|E| \log(|V|) \log(|E|))$.

F: Endless Turning

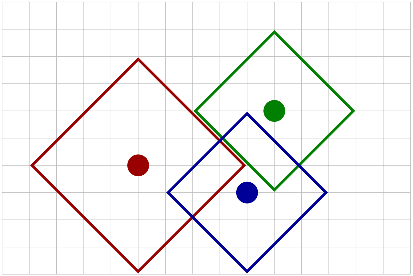
Benelux Algorithm Programming Contest 2016

- For each pair of streets calculate their intersection.
- For each street find the order in which the intersection points lie in that street, using a sort algorithm.
- Find the street on which the starting point is located.
- Now simulate the driving, keeping track of the direction in which you are traversing the streets.
- If you arrive at the first intersection for the second time, take N modulo the number of turns taken so far.
- Finish the simulation.
- *Funny fact:* as you walk around a polygon, in each street you will visit only two intersections: one where you enter each time and one where you leave.

G: Manhattan Positioning System [1/2]

Benelux Algorithm Programming Contest 2016

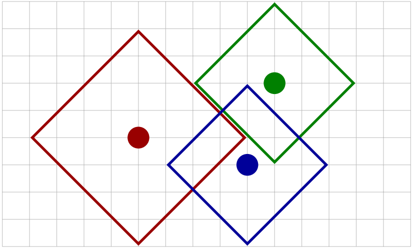
- Task: Find a unique point at specific Manhattan distance to each beacon.
- The set of points at specific distance to one beacon is a “circle”. Under Manhattan distance metric, a “circle” looks like a diamond shape.
- Task: Find the intersection of the diamond shapes of all beacons.



G: Manhattan Positioning System [2/2]

Benelux Algorithm Programming Contest 2016

- Choose one beacon.
- Create abstract representation of its diamond shape:
  Set of line segments, $\{(x1, y1, x2, y2), \ldots\}$.
- Visit all other beacons, and intersect the remaining set of line segments with the other beacon's diamond shape.
- After processing all beacons; the set of line segments is empty (*impossible*) or contains exactly one point (unique solution), or contains multiple points/segments (*uncertain*).



H: Multiplying Digits [1/3]

Benelux Algorithm Programming Contest 2016

Given a number n and a base b, find the least x such that the product of the digits of x (x written in base b) equals n.
Possible **digits** of x are divisors of n that are less than b.
Find an *ascending* sequence of digits, such that

- Their product equals n,
- the sequence is as short as possible
- the sequence is lexicographically minimal.

It is tempting to put the largest possible digit at the end. But that is wrong (Sample 3):

- $b = 9$; $n = 216 = 2 * 2 * 2 * 3 * 3 * 3$.
- Choosing 8 as last digit gives 3 3 3 8 $\Rightarrow$ 1115.
- However 6 6 6 $\Rightarrow$ 546 is a better (the best) solution.

H: Multiplying Digits [2/3]

Benelux Algorithm Programming Contest 2016

Dynamic Programming, memoize the function Best:

- If n has a prime divisor $\geq b$, there is no solution.
- function Best(long k) gives the best solution.
- base case: if (k $<$ b) Best $=$ k
- recursion:

```
for (d < BASE, d divides k)
   find solutions ending with digit d, as follows:
      k1 = k/d
      b1 = Best(k1)
      sol1 = b * b1 + d
and return the best (least) of the sol1
```

- The *least* of the sol1 will be less than 2^{63}, but not necessarily *all* sol1,
- so beware of overflow!

H: Multiplying Digits [3/3]

Unfortunately, this is not fast enough. We need some of the following optimizations:

- Store the possible digits beforehand (the divisors of n below b)
- If $d * d < b$ then d will not occur in an optimal solution, except as the *first* digit. The left neighbour of d, say d_1, is at most d so the two can be replaced by $d_1 * d < b$, making a smaller number.
- If a multiple of a digit d can be chosen as the last digit in the solution for some k, then d will not be the last digit in the optimal solution for that k.

I: Older Brother

Is q a prime power? Use a simplified factorization algorithm:

```
bool isPrimePower(int q) {
    if (q == 1) // Corner case.
        return false;
    for (int p = 2; p * p <= q; p++) {
        if (q % p == 0) {
            // Least divisor will be prime.
            // Check if q is a power of p.
            while (q % p == 0)
                q /= p;
            return q == 1;
        }
    }
    // Apparently, q is prime.
    return true;
}
```

J: Programming Tutors

We are looking for a matching which minimizes the maximal distance between pairs. Some ways to solve this efficiently enough include:

- Use a binary search over the maximal distance. Given a candidate maximal distance, use your favourite matching algorithm.
- Use a standard minimal matching algorithm, but look for augmenting paths with minimal highest distance, instead of minimal total distance.
- Even fast enough: start with an empty partial matching, allow new edges one by one starting with the shortest, look for a new augmenting path each time.

K: Safe Racing [1/2]

Benelux Algorithm Programming Contest 2016

- General remark: reduce modulo 123456789 in all intermediate calculations to avoid overflow.
- Calculate

$$D[i] = \begin{cases} \text{number of ways to allocate marshalls to} \\ \text{booths 0 up to and including } i \text{ given that} \\ \text{there is a marshall in booths 0 and } i \end{cases}$$

for $i = 0, \ldots, L-1$, using dynamic programming in runtime $\mathcal{O}(L)$ using:

$$D[i] = \sum_{j=\max(0,i-S)}^{i-1} D[j].$$

- During the process, keep track of the partial sums of the last S values. Do not recalculate them to avoid getting runtime $\mathcal{O}(S \cdot L)$, which is too big.

K: Safe Racing [2/2]

- If the first marshall is at position f and the last one at position $L - g$ (satisfying $f \geq 0$, $g \geq 1$ and $f + g \leq S$), then the number of ways to put marshalls in between these positions is $D[L - f - g]$.
- Hence, the answer is

$$\sum_{f=0}^{S} \sum_{g=1}^{S-f} D[L - f - g],$$

  but naively it would take $\mathcal{O}(S^2)$ time to calculate this.
- Notice that each value of $h := f + g$ occurs h times in the sum. Hence, we can also write the answer as

$$\sum_{h=1}^{S} D[L - h] \cdot h,$$

  which can be calculated in $\mathcal{O}(S)$.

L: Sticks [1/2]

- Among a sequence of numbers, are there three that form the side of a triangle?
- That is, are there $a < b < c$ with $a + b > c$?

L: Sticks [1/2]

- Among a sequence of numbers, are there three that form the side of a triangle?
- That is, are there $a < b < c$ with $a + b > c$?
- There are too many to check all triples.
- If any triple works, then a triple of consecutive lengths does.
- Solution: sort the list of stick lengths. Check if sticks $i, i + 1, i + 2$ form a triangle.

L: Sticks [2/2]

Benelux Algorithm Programming Contest 2016

- The biggest set of sticks for which no solution exists are Fibonacci numbers:

$$1, 1, 2, 3, 5, 8, 11, \ldots$$

- The largest Fibonacci number allowed ($< 2^{60}$) is F_{88}.

L: Sticks [2/2]

- The biggest set of sticks for which no solution exists are Fibonacci numbers:

$$1, 1, 2, 3, 5, 8, 11, \ldots$$

- The largest Fibonacci number allowed ($< 2^{60}$) is F_{88}.
- Silly solution: if $n > 90$, it is always possible.
- If $n \leq 90$, check all possible triples.