Welcome Contest: European Selection

Moscow ACM ICPC Prefinals Workshop 2020

April 18, 2020

A. Automatic Banking

Keywords: scanline, RMQ data structure.

For each coin j we need to find $\min i$ such that $a_i \geqslant u_j$, $b_i \geqslant v_j$, and sum $\min i$ over all coins.
Treat a_i, u_j as x-coordinates, and b_i, v_j as y-coordinates. **Compress** all x's and y's.
Sort all coins and slots by decreasing of x, and **scanline with RMQ**. Initialize with all $RMQ[y] = \infty$.
For a slot i with $a_i = x$ update $RMQ[b_i] = \min(RMQ[b_i], i)$.
For a coin j with $u_j = x$ find the answer as $\min(RMQ[v_j \dots \infty])$.
For each x make sure that **updates are processed before queries**.
Total complexity: $O((n+m)\log(n+m))$.

B. Basketball Competitions

Keywords: dynamic programming.

Let $dp[i, j]$ be the smallest cost of team i winning j games. Initialize $dp[i, 0] = 0$.
For team i, let S be the set of teams that can play the team i at stage j. Then

$$dp[i, j] = dp[i, j-1] + \min_{i' \in S}(dp[i', j-1] + cost(i, i')).$$

Here $cost(i, i') = (s_i - s_{i'})^2$ if $s_i < s_{i'}$, and $cost(i, i') = 0$ otherwise.
The answer is $dp[1, r]$.
Observe that each pair of teams i, i' is considered at a unique stage, thus the **total complexity** is $O(n^2) = O(2^{2r})$.
To only find $dp[1, r]$ even less states are necessary.

C. Coins Distribution

Keywords: greedy, priority queue.

Let c_x be the multiplicity of coin type x.
Proposition: optimal answer is obtained by repeatedly taking k coins of types with largest c_x.
Proof sketch: let $T_1, \dots, T_{ans}$ be the optimal answer, and S consist of k most frequent types.
If $i \in T_1 \backslash S$, $j \in S \backslash T_1$, and $c_i < c_j$, try to find a T_2 containing j and not containing i, and swap $i \in T_1$ and $j \in T_2$.
If there is no such T_2, then i is contained in more T's than j, and we can swap all occurences of i and j.

Use **priority queue** Q to repeatedly find k most frequent types, decrease their c_x by one and reintroduce them back into Q.
At most $O(n)$ operations with Q with be made, thus **total complexity** is $O(n \log n)$.

D. Drawing of Electropolitan

Keywords: tree isomorphism, hashing, tree centroid.

Nodes v and u are indistinguishable if trees rooted at v and u are isomorphic.

Idea 1. We define a **hash** of a rooted tree as any function that depends on the multiset of hashes of root's children (but **not their order**).
Ex.: $h(T) = p(h(T_1), \dots, h(T_k))$, where $T_1, \dots, T_k$ are root's children subtrees sorted by $h(T_i)$, p is a random enough function.
Find hashes $h(v)$ for the tree rooted at each node v.
First root the tree at 1, and find hashes of all subtrees.
Next, find the hash of a subtree **to the top** of each node.
Do this top to bottom. To proceed to children of v, exclude children of v from $h(v)$ one at a time (e.g. with a data

structure or prefix sums).
Finally, count different $h(v)$.

Idea 2. Root the tree at its **centroid** — middle of the longest path.
Find hashes of all rooted subtrees. v and u are interchangeable if all their parents at equal heights have equal hashes.

In any case, **total complexity** is $O(n \log n)$ or $O(n)$.

E. Economics of Football

Keywords: assignment problem.

Define $a_{i,j}$ as:
- x if team i wants to buy goalie j for amount x;
- 0 if $i = j$;
- $-\infty$ otherwise.

Any set of deals is now an **assignment** with weights $a_{i,j}$. Find **optimal assignment** with weights $a_{i,j}$ with Hungarian algorithm or min-cost max-flow.
Total complexity: $O(n^3)$ or slightly worse.

F. Flawed Grid

Keywords: meet-in-the-middle, hashing.

Assign a vector of color multiplicities $(x_1, \ldots, x_c)$ to each row.
We need to select a largest subset of rows such that the sum of their vectors if $(x, \ldots, x)$ for some $x \leqslant n$.
Instead of matching the vectors we can match their **polynomial hashes** modulo P.
Use **meet-in-the-middle** to find a subset that sums up to $(x, \ldots, x)$ for all x.
Total complexity: $O(nm)$ (compute hashes for all rows) + $O(n2^{n/2})$ (compute hashes of all subsets of each half and sort them) + $O(n2^{n/2})$ (try all x and use two pointers to find matching sums). To avoid **collisions** make sure that $P >> 2^n$, or use several coprime P simultaneously.

G. Gregory's House

Keywords: greedy.

Let h_i be the hook that initially has the tie i.
Observe that the number of times t_i each h_i is visited **does not depend on the order** of h_i, and can be found with simple simulation.
Sort i by decreasing of h_i and find the answer.
Total complexity: $O(n \log n)$.

H. Health Insurance

Keywords: scheduling, Johnson's rule.

For each patient of type 3 (any order) try all options of two orders.
Let S_A and S_B be the sets of patients that want to visit A/B first.
Observation: in an optimal answer prof. A will see S_A in some order, then S_B in some order (same for B). Further, orders of S_A and S_B will be the same for both professors.

If S_A and S_B are now ordered, the answer is the largest of four options:
- $\sum_x A_x$, $\sum_y B_y$;
- for $x \in S_A$, the time to finish the two-machine job assignment with times (A_x, B_x);
- for $y \in S_B$, the time to finish the two-machine job assignment with times (B_y, A_y).

Proof sketch: if professors do not wait, the answer is one of the first two options. Otherwise, the only reason for prof. B to wait is when a patient x sees him immediately after finishing with prof. A, in which case the answer is the third option.
The two last options are independent, thus we can order S_A and S_B according to Johnson's rule (put x before y if $\min(A_x, B_y) < \min(A_y, B_x)$, with some care for equalities).

Total complexity: $O(2^n n \log n)$.

I. Ink Jet

Keywords: binary search, circle union area.

Binary search on the time T. We now need to find the area of circles union at time T and compare it with A.
Finding circle union area: for each circle C_i find the part of its circumference that is on the border of the union. For each other circle C_j, find the arc of C_i inside C_j (possibly the entire circumference).
Use scanline to find uncovered arcs. For each arc compute directed area under the arc and add it to the answer.
Take care of coinciding circles.
Total complexity: $O(n^2 \log n \log C)$, where C is relative precision of binary search.

J. Just Kidding

Keywords: greedy hacks.

Try to fill the **second lowest** row from left to right.
Make sure that all free cells in the row are on the right and are not beneath an occupied cell.
If not possible to continue the row, drop the piece on the left.

K. Knight or Liar?

Keywords: combinatorial probability, DP on partitions.

Suppose there no initial statements. Let us construct a directed graph with edges $i \to j$ meaning that i said something about j.
Suppose that "knight/liar" statements are now equiprobable. Each weakly connected component of the graph is a cycle with trees.
Observation: consistency depends only on the statements in the cycle, and the probability is $1/2$ if at least one statement on the cycle is random.
For each consistent set of statements, there are exactly two ways to recover the identites of everyone in the component.
Takeaway: the answer only depends on the number of components.

Draw edges from $1, \ldots, n$ successively. At any time each component is a rooted tree or a cycle with trees.
Use **unordered multiset of tree sizes** as parameter of DP. Transition = drawing an edge from the root of any tree to any other component, or to the same component (transforming the tree into a cycle with trees).
If there are initial edges, check for consistency and alter the initial state accordingly (the initial components are now trees/cycles with trees in the initial graph).
Total complexity: $O(p(n)n^c)$, where $p(n)$ is the partition function.

L. Lazy Managers

Keywords: DP, greedy, proofless handwaving.

Solution 1.

Consider the passes we purchase in chronological order. Keep track of the remaining time and number of uses on each one. **Assumption 1.** At any point we only need to keep track of at most two last passes.
This way we can use DP with $O(nC^4)$ states, where $C = \max(A, B)$.
However, the assumption is **wrong**. But
Assumption 2. At any point we only need to keep track of at most **three** last passes.
turns out to be correct. Can be proven with case analysis or brute-force on reachable states.
Keeping only reachable states and removing clearly unoptimal states helps with TL.

Solution 2.

Consider the first day we have to use a pass. For the next B days, the optimal usage turns out to be as follows: • use on the first day;
• use on earliest days where two usages are needed;
• use on earliest days where one usage in needed.
Use this order until all A usage are expired.
Update a_i, b_i accordingly, and do this until all $a_i = b_i = 0$.

M. Matches Arrangement

Keywords: brute-force, complex-state DP.

Try to remove any subset of matches, and check if it's possible to place them back to obtain a valid equality from the resulting expression F.
Place matches from left to right, and keep track of the formed partial expression E. We can do one of the following:

- create a new empty symbol (a gap) at the end of E (if the last symbol of E is valid);
- append the next symbol of F to E;
- place a match on top of the last symbol of E.

We can apply DP if we only store the following information:

- the number of remaining matches;
- the number of processed symbols of F;
- whether = was obtained in E;
- the balance in the equation (LHS - RHS);
- the last encountered sign (-/+);
- the entire sub-expression (part of a number) formed since the last sign/=/beginning of E.

This way, the number of states is small enough since the balance can be bounded by ~ 100, and the number of possible sub-numbers is small.