# Problem Tutorial: "Goldberg Machine 2"

Let's construct an invariant that is preserved as a token moves through the cells. We want it to be purely additive, i.e. look like $P(B) = T(c) + \sum_c H_B(c)A(c)$, where $T(c)$, $A(c)$ are values for a token and a horizontal arrow at cell $c$, $H_B(c)$ is the indicator of a horizontal arrow in the board $B$.

For each cell $c$ just outside of the grid, assign a token in this cell a value $T(c) = x_c$ (formal variable). For any other cell, say, in row $x$ and column $y$ we need to have $T(x, y) + A(x, y) = T(x, y + 1)$, $T(x, y) = A(x, y) + T(x + 1, y)$ to preserve the sum for two possible movement directions. This implies $T(x, y) = (T(x, y + 1) + T(x + 1, y))/2$, $A(x, y) = (T(x, y + 1) - T(x + 1, y))/2$. If we take $T(x, y)$ and $A(x, y)$ to be rational linear combinations of all variables $x_c$, we can compute all $T(x, y)$ and $A(x, y)$ with simple DP; one can easily see that all coefficient denominators are powers of 2. Since tokens are removed once they reach terminal cells, to preserve the invariant we will consider all coefficients modulo 1.

Crucial **proposition**: $P(B_1) \equiv P(B_2) \mod 1$ implies $B_1 = B_2$. Indeed, locate the lexicographically smallest cell $c$ where $H_{B_1}(c) \neq H_{B_2}(c)$. Let $c'$ be the closest cell to $c$ to the right in the same row. One can see that $A(c)$ contains $x_{c'}$ with coefficient $2^{-d}$, where $d$ is the distance between $c$ and $c'$. Further, all cells lexicographically later have strictly smaller denominators, therefore $P(B_1) \equiv P(B_2) \mod 1$ is impossible.
Note that there are smaller collections of variables are also sufficient for this property (e.g. cells immediately to the right of each consecutive horizontal run of cells).

Let $B+k$ be the result of placing and resolving $k$ tokens in the top-left cell $c_0$ in a board $B$. By construction we have $P(B + k) \equiv P(B) + kT(c_0) \mod 1$. This, together with injectiveness of $P$, implies many useful things:

- The smallest number $k$ such that $B + k = B$ does not depend on $B$ and is equal to the largest denominator among coefficients of $T(c_0)$. We refer to this number as the period $2^p$.
- For two boards $B_1, B_2$, the solution to $B_1 + k = B_2$ is given by $k = (P(B_2) - P(B_1))/T(c_0) \mod 2^p$ when division is possible.

To perform updates, maintain $\Delta = P(B_2) - P(B_1)$. Each single arrow change in one of the boards changes $\Delta$ by an $A(c)$. Checking that $\Delta$ is divisible by $T(c_0)$ is straightforward.

To speed up we will get rid of handling lots of variables. Instead of maintaining linear combinations themselves, let's perform a random substitution $X$ for variables $x_c$. Now we can calculate $P(c)(X)$ and $A(c)(X)$ with the same DP. A few modifications will be needed:

- We need to know $2^p$. Note that $P(c_0)(X)$ has denominator $2^p$ with probability $1/2$. Make several substitutions $X_1, \ldots, X_r$ and take $2^p$ as the largest denominator of $P(c_0)(X_i)$.
- We compute the answer as $k(X) = \Delta(X)/T(c_0)(X)$, which is defined modulo $2^p$ only when $T(c_0)(X)$ has denominator $2^p$. As above, retrying enough times helps to find suitable $X$ (or any collection $X_1, \ldots, X_r$ of them).
- To check for consistency, we have to verify $k(X_1) = k(X_2) = \ldots = k(X_r)$. The probability of a false positive is bounded by $2^{-r}$, with the worst case being when $\Delta \bmod T(c_0) \neq 0$ has all coefficient denominators equal to 2. Taking, say, $r = 30$ ensures that the answer is correct with high probability.

# Problem Tutorial: "Nein"

Denote the number $\underbrace{999\ldots9}_{k}$ by $M$. Note that $10^k \equiv 1 \pmod{M}$, which means that if we split the decimal representation of a number $x$ into blocks of $k$ digits (starting from the least significant digit) then $x$ has the same remainder modulo $M$ as the sum of these blocks (treated as numbers).

Instead of looking for the $n$-th positive integer $x$ so that $x \cdot M$ doesn't contain nines, we'll look for the $n$-th positive integer $y$ so that $y$ contains no nines and is divisible by $M$, and then output $y/M$. Assume that we know for sure that the number of blocks doesn't exceed $B$ (that is, $y$ has no more than $B \cdot k$ digits). Then we find its digits one after another, starting from the most significant.

Now we want to solve the following subtask: assume that the current prefix is $p$ (of length $l$), find out how many numbers there of type $10^{Bk-l} \cdot p + x$ with no nines in its decimal representation and divisible by $M$ (and $x < 10^{Bk-l}$, of course). It is equivalent to solving the following task: calculate the number of 0-8 sequences of length $l' = Bk - l$ which, when treated as integers, have remainder $r$ modulo $M$. If we denote this sequence by $\{d_i\}_{i=0}^{l'-1}$ then one can rewrite the condition above like this:

$$\sum_{i=0}^{l'-1} d_i \cdot 10^i \equiv r \pmod{M}.$$

As we already know, this remainder is the same if we sum blocks of $k$ digits, making the condition as follows:

$$\sum_{i=0}^{l'-1} d_i \cdot 10^i \mod M \equiv r \pmod{M} \Leftrightarrow \sum_{i=0}^{l'-1} d_i \cdot 10^i \mod M \in \{r, r+M, \ldots, r+(B-1)M\}.$$

So if we learn how to count such sequences with a fixed sum $t$ of blocks, we are done. To count them, one can precalculate `digit_sum_ways[k][s]` which is the number of ways to represent $s$ as the sum of $k$ digits 0-8 (this is fast and can be done once, because $k \le B$, $s \le 8k+1$). After this, let's note that for a given $l'$ and for all $i$ such that $0 \le i < k$ there are $\left\lfloor \frac{l'+k-1-i}{k} \right\rfloor$ digits which are multiplied by $10^i$. After this one can compute `dp[i][c]` which is the number of ways to pick all digits which are multiplied by something under $10^i$ so that their sum $c \cdot 10^i + (t \mod 10^i)$, where $0 \le i \le k$ and $0 \le c < B$. It can be recalculated by the following recurrence relation:

$$dp[i+1][c] = \sum_{c'=0}^{B-1} dp[i][c'] \cdot digit\_sum\_ways\left[\left\lfloor \frac{l'+1-i}{k} \right\rfloor\right]\left[10c' - c + \left\lfloor \frac{t}{10^i} \right\rfloor \mod 10\right].$$

The answer is then stored at `dp[k][0]`.

Total time complexity is about $O(Bk \cdot 9 \cdot Bk \cdot B)$. It can be shown that $B = 20$ is enough, so this easily fits into the time limit.

# Problem Tutorial: "MIPT: Connecting People"

The resulting configuration can be turned into a tree, with vertices being individual floors, and edges being vertical and horizontal transitions. We can compute the answer for any such tree as $\sum_e w(e) S_1(e) S_2(e)$, where summation is over all edges $e$, $w(e)$ is the cost of the edge, and $S_1(e), S_2(e)$ are sizes of the two halves of the tree with respect to the edge $e$.

Let's root the tree, say, at the top floor of skyscraper 1. By analyzing possible rooted subtrees, we find three cases:

1. an entire segment of skyscrapers $[l, r]$, rooted at a floor $x$ of a skyscraper $i \in [l, r]$;
2. several bottom floors $1, \ldots, x$ of a skyscraper $i$, together with zero or more entire skyscrapers $[l, i-1] \cup [i+1, r]$, with floor $x$ of skyscraper $i$ being the root;
3. several top floors $x, \ldots, h_i$ of a skyscraper $i$, together with zero or more entire skyscrapers $[l, r]$ such that $i \notin [l, r]$, with floor $x$ of skyscraper $i$ being the root.

For each of those cases we will compute DP — the smallest value of $\sum_e w(e)S_1(e)S_2(e)$ inside a subtree of this sort. There are $O(n^2R)$ states, where $R = \sum h_i$. What are possible transitions to smaller cases?

For subtrees of sort 1, we consider all ways to split $[l, r]$ into two subsegments such that one of them is reachable by going up from the root, and the rest.

For subtrees of sort 3, a portion of $[l, r]$ may be reachable by taking a horizontal transition from the root floor $x$ of $i$. This portion has to be either a prefix or a suffix of $[l, r]$, depending on mutual arrangement of $i$ and $[l, r]$. Try all ways to split $[l, r]$ and move the root to the floor $x + 1$. In all cases, we know sizes of resulting subtrees, therefore, can figure out contributions of traversed edges to the answer.

For subtrees of sort 2, the root floor $x$ may be connected to some prefix and suffix of $[l, r]$. Instead of trying $O(n^2)$ cases, we can introduce an extra parameter $f$: $f = 0$ means "try to separate prefix, don't move the root down", $f = 1$ means "try to separate suffix, move the root down".

Note that whenever we consider a horizontal transition from a root floor, the other endpoint of this transition is defined unambiguously (closest floor on the same height in the chosen direction). The skyscraper containing this endpoint has to be contained in a suitable part of the subdivision of $[l, r]$.

We have to consider $O(n)$ transitions from any state, therefore total complexity is $O(n^3R)$, with low constant factor since order of $i, l, r$ is constrained.

# Problem Tutorial: "Matryoshka Dolls"

Let's do something like divide&conquer. Assume that we fixed segment $[L, R]$ with median $M = \frac{L+R}{2}$ and want to calculate the answer for all queries $[l, r]$ inside this segment such that $l < M \le r$. We will find the left half of the sum and the right half separately from each other. To find the first value, let's iterate elements in the left part one by one starting from $M - 1$.

Assume that we fixed $L \le I < M$. Let maintain the set of values $p_I, p_{I+1}, \ldots, p_M$ and sum $S_I$ of distances between neighbouring elements inside this set. Let call elements $p_x$ and $p_y$ neighbours in current set if there is no $z$ in current set such that $p_x < p_z < p_y$. By adding element $p_I$, some pair $(p_x, p_y)$ (if exists) becomes not neighbours, but there appear neighbour pairs $(p_x, p_I)$ and $(p_I, p_y)$. So $S_I$ is simply recalculated in respect to appeared/disappeared pairs.

For each neigbour pair $(p_x, p_y)$ in current set, let find out the leftmost index $z$ in right part such that $p_x < p_z < p_y$. It is clear that if $z \le r$ than the nearest greater element for $p_x$ on segment $[l, r]$ have index $k \ge z$, same for the lower element for $p_y$. Let maintain some Fenwick $D^I$ on right part, for every index $z$ described above let add $(M - x) + (M - y) - |x - y|$ to $D^I_z$. It means that starting from $z$ these elements are no more neighbours, and instead of adding distance between them we want to add sum of distances to median $M$.

To calculate the answer for the query $[l, r]$ we want to sum up value $S_l + \sum_{i=M}^{r} D^l_I$ with the same value for the right part. For each neighbour pair $(p_x, p_y)$ on that segment such that $x < y$, if it contains in same part their distance will be counted in $S_l$. Otherwise it will be counted like sum $(M - x) + (y - M) = y - x$ in $D^L_i$ and $D^R_j$ for some $i >= M$ such that $p_i > p_x$, and some $j < M$ such that $p_i < p_y$.

The complexity is a $O(nlog^2n)$ for preprocessing because we can maintain sets, nearest elements, and Fenwicks in $O(logn)$, and $O(logn)$ for one query.

# Problem Tutorial: "No Rest for the Wicked"

Denote by $f(x, v)$ the maximal possible spectacularness one can achieve starting in $v$ with COVID level $x$. How can we find it?

Assume we have a COVID level $x$. Some countries (with $t_i < x$) are banned for us forever, consider all other countries. Call an edge $i \leftrightarrow j$ *free* if both $c_i$ and $c_j$ do not exceed $x$. If $c_i \le x < c_j$ then we say that this edge is *one-way*. We call a *free component* any connectivity component of the graph with only free edges, and we denote the free component of any vertex $v$ by $FC(v)$.

If we start in $v$ with COVID level $x$ then there are two ways: either we will visit the most spectacular

country with the same level $x$ or we increase it at least once. In the first case this country is among the free component of vertex $v$, in the other case we need to go along any one-way edge at least once. This gives us the following relation:

$$f(x, v) = \max\left(\{s_u : u \in FC(v)\} \cup \{f(c_w, w) : u \in FC(v),\ u \to w \text{ is a one-way edge}\}\right).$$

One can see that every edge $(i, j)$ is a free edge for $\max(c_i, c_j) \le x \le \min(t_i, t_j)$ and is a one-way edge for $c_i \le x \le \min(t_i, c_j - 1)$, that is, it can come in these states on some segments of $x$. This gives us the following divide-and-conquer solution:

```
procedure solve_for_xs(l, r, free, oneway, possible_free, possible_oneway):
  // relaxes all answers with f(x, v) for x from [l, r)
  // free is a set of edges which are free on the whole [l, r)
  // similar with oneway
  // possible_free and possible_oneway are sets of edges which are
  //   free and one-way for some x from [l, r), but not for all of them
  update the components with all free edges
  update the answers for components with all oneway edges outgoing from them
  if l + 1 == r:
    for all countries with c_v == x:
      update ans[v] with f(x, v)
  else:
    m = (l + r) / 2
    let free_left be the set of edges from possible_free which are free on [l, m)
    let free_right be the same for [m, r)
    let oneway_left, oneway_right be the same for one-way edges
    let possible_(free/oneway)_(left/right) be the same for possibles
    solve_for_xs(l, m, free_left, oneway_left, possible_free_left, possible_oneway_left)
    solve_for_xs(m, r, free_right, oneway_right, possible_free_right, possible_oneway_right)
  rollback all updates from free and oneway
```

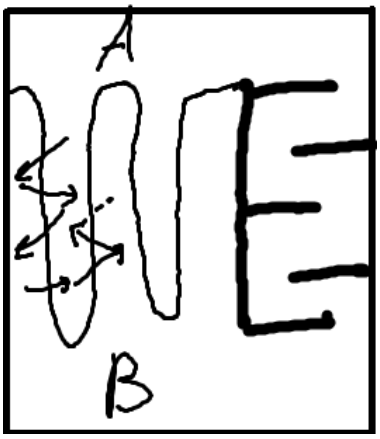
One can use DSU with rollbacks to obtain a solution working in $O(n \log^2 n)$.

# Problem Tutorial: "The Last Samurai"

We will not try to exploit the greediness, and construct an arrangement with a large best-case answer. If the board is $n \times n$, the answer is clearly bounded by $O(n^4)$: each of $O(n^2)$ pieces will be captured in $O(n^2)$ moves. How do we achieve this bound?

One natural way is to construct a chain of $\Omega(n^2)$ pieces, one protecting the next, so that each capture takes $\Omega(n^2)$ moves. We'd like to have two safe regions $A, B$ connected by a very long ($\Omega(n^2)$) safe path, so that the chain contains pieces in $A$ and $B$ alternately.

At the same time, $A$ and $B$ should be next to each other so that pieces can attack across the (protected) border easily. However, we can find that, assuming no pieces form an unattackable cycle, the number of pieces attacking across a border are bounded by the length of this border. Therefore, we must make a border long as well, by making another "snake", this time out of protected cells. So the board should look something like this (black meaning attacked cells, arrows meaning the chain, one protecting the next):




The obstructions should not be vulnerable until the chain is destroyed, thus the last piece of the chain should protect them. Obstructions themselves may be constructed as similar chains.

Actually building a construction like that is pretty hard. It helps to construct several "gadgets" that can function independently of the rest of the board, such as:

- an "impervious ray" of attacked cells (e.g. a rook blocked on three sides with knights, and protected by another knight for easier chaining),
- a "king-permeable ray" (a bishop protected on three diagonal sides with knights, useful for protecting rooks that shape the snake path on the right),
- an "alternating chain piece" that sits on the border between $A$ and $B$, e.g.

```
. . .
n.Xrn
.nXn.
..X..
..X..
nrX.n
.nXn.
. . .
```

You **will** need to implement a checker of your solution, and carefully construct a program that produces parts of your construction, checking for conflicts at every step.

# Problem Tutorial: "Nikanor Loves Games"

Let $X$ be the set of all coin values of Nikanor's friends, and also $1 \in X$. Assume that he is going to buy a coin $(a, b)$, where (say) $a \notin X$. Then there is a number $x \in X$ which is less than $a$. Then it's more profitable to buy a coin $(x, b)$ instead, since it is cheaper and provides the same probabilities of winning against whoever. Hence, it's always best to pick coin values from $X$.

Assume that we have fixed some $c \in X$ and we want to choose $y \in X$ to buy a coin — that is, to maximize the value of

$$-cy + \sum_{i=1}^{n} x_i \cdot \mathsf{P}\left((c, y) \text{ wins against } (a_i, b_i)\right) = \sum_{i=1}^{n} \frac{x_i}{4} \cdot \left([c \geq a_i] + [c \geq b_i] + [y \geq a_i] + [y \geq b_i]\right) - cy.$$

Denote the whole sum without the $-cy$ by $f_c(y)$. Then, if we consider the set $S$ of points $(y, f_c(y))$, we want to find the highest line of a fixed direction containing a point of this set, which can be done by binary search over the upper convex hull of this set.

Let's iterate $c$ over $X$ in ascending order. What changes in the set $S$ is that some points increase their $f_c(y)$, and after this we have to perform a binary search with the new direction. One can add points to this convex hull in amortized $O(\log n)$ (because there is no need to remove any of the old points). It's also easy to see that there will be at most $2n$ increases, as each of indicators of type $[c \geq \text{smth}]$ increases at most once.

We also need to build the initial convex hull for $c = 1$, but this can be done by a simple Fenwick tree, segment tree or even prefix sums: first we add $x_i$ to points $a_i$ and $b_i$ for all $i$, then for each $y$ we find the value on the prefix. The total complexity is $O(n \log n)$.

# Problem Tutorial: "Roads of the Empire"

Let's clarify the structure of the graph. We can see that for every $y$ there is at most one $x$ with an edge from $x$ to $y$. Moreover, this $x$ exists if and only if $n$ is not divisible by $y$, and equals $y - (n \bmod y)$.

To get the answer, we can decrease $max(x, y)$ until they are equal or $n$ is divisible by this maximum (in this case, the answer is $-1$). The random structure of tests guarantees that the number of steps is small.

# Problem Tutorial: "Drunkards"

Let's calculate $dp_{i,j}$ — the probability to visit the location at a distance $j$ from the current location at least once if the last $i$ moves remain. Note that $j$ has range $[-i; i]$

$dp_{0,0} = 1$ is a base of our dp.

$dp_{i,j} = dp_{i,j} \cdot p + dp_{i,j-1} \cdot (1 - p)$ if $a_{n-i} = 1$. The similar transition takes place in the opposite case. Finally, we have to assign $dp_{i,0} = 1$, as it is always possible to reach the current cell.

The complexity of this solution is $O(n^2)$

# Problem Tutorial: "Rational Dimasik"

Denote by $\text{ord}_p(x)$ the maximal integer $k$ such that $x$ is divisible by $p^k$. Let $\frac{a}{b}$ and $\frac{c}{d}$ be two irreducible fractions with difference $\frac{x}{y}$, which is also irreducible. Then we claim that for any prime $p$ if $\text{ord}_p(b) \neq \text{ord}_p(d)$ then $\text{ord}_p(y) = \max(\text{ord}_p(b), \text{ord}_p(d))$. Indeed,

$$\text{ord}_p(y) = \text{ord}_p(bd) - \text{ord}_p(ad - bc) = \text{ord}_p(b) + \text{ord}_p(d) - \min(\text{ord}_p(b), \text{ord}_p(d)) = \max(\text{ord}_p(b), \text{ord}_p(d)).$$

Let's deal with the case when $\text{ord}_p(b) = \text{ord}_p(d) = k$. Let $b' = b/p^k$ and $d' = d/p^k$. One can see that

$$\text{ord}_p(y) = \max(0, k - \text{ord}_p(ad' - cb')) = k - \sum_{i=1}^{k} [p^i \mid ad' - cb'] = k - \sum_{i=1}^{k} [a/b' \equiv c/d' \pmod{p^i}],$$

where the last division is performed modulo $p^i$. That means that to calculate the total power of $p$ in the answer it takes the following:

1. Group all fractions with $p$ in its reduced denominator by the order of $p$ in it;
2. For each order $k$ add $k$ times the number of all fractions with strictly less order to the total power;
3. For each order $k$ and all $1 \leq i \leq k$ consider all fractions with order $k$, group them by their $a \times b'^{-1}$ mod $p^i$ and add $\binom{size}{2}$ for the size of each group to the total power.

Since each number will be considered for at most, say, 10 pairs $(p, k)$, each inverse can be taken in $O(\log(p^i)) = O(i \log p)$, the total complexity is about $n \log n + 10n \log^2(10^6)$, which is actually faster, because for most of the primes their order in any denominator is much less than $\log 10^6$.

# Problem Tutorial: "Mission Impossible: Grand Theft Auto"

The bound on the number of paths in the answer is almost tight. Indeed, every leaf has to be a path endpoint for at least one operation, therefore we only have at most 2 endpoints to spare.

One can analyze how can a case when $m$ is even be solved in exactly $m/2$ steps. For simplicity we can ignore vertices of degree 2. It is not hard to see that, assuming each leaf is used as an endpoint exactly once, each operation effectively erases (makes unreachable) two leaves, together with their degree-2 chains of reachable vertices. Further, once a vertex was covered by an operation, it has to be covered by next consecutive operations as well, until it becomes unreachable.

One can show that for any such sequence of operations it is possible to root the tree and order children of each vertex, then number the leaves in the DFS pre-order, so that the paths have endpoints $(0, m-1)$, $(1, m-2)$, $\ldots$, $(m/2-1, m/2)$ in this order.

There is a possible obstruction to this approach, namely, an edge $e$ that is not covered by any path. Indeed, operations $1, \ldots, z$ and operations $z+1, \ldots, m/2$ (for some $z$) will be happening on opposite sides of $e$. Then, after $z$ operations $e$ will become a "leaf edge" in the subgraph of reachable vertices, thus we will have to waste at least one more operation.

Let's try to carry this intuition to our problem. Order leaves $0, \ldots, m-1$ in an arbitrary DFS pre-order, choose an index $x$, and try performing operations connecting leaves $(x-1, x)$, $(x-2, x+1)$, $\ldots$ (indices taken modulo $m$). For each $x$, how many bad edges (as described above) are there? Before we count them, let's compress all degree-2 chains.

Halves of the tree with respect to any edge $e$ contain leaves with indices in **cyclic segments** $[l, r)$ and $[r, l)$. If, say, $l < r$ and $r - l$ is even, then $e$ is bad for $x = (l + r)/2$ (similar for the midpoint, if any, of the other segment).

Now consider two cases of parity of $m$:

- $m$ is even. After compressing the chains there are at most $2m-3$ edges in the tree. An edge adjacent to a leaf can not be bad for any $x$, thus $m-3$ candidates remain. Each of them is bad for at most two values for $x$, thus there exists an $x$ with at most one bad edge. Start matching leaves $(x-1, x)$, $(x-2, x+1)$, $\ldots$, until we (potentially) reach the only bad edge. At this point, perform any operation covering this edge, removing the extra "leaf". After that, finish matching leaves as prescribed. This makes at most $m/2+1$ operations.
- $m$ is odd. Observe that every $x$ has at least one bad edge, namely, the leaf edge $x - \frac{1}{2} \bmod m$, "opposite" of $x$ in the tree. Further, every edge has exactly one subtree with an even number of leaves. Since there are at most $2m-3$ edges, we conclude that there is such $x$ that the only bad edge is opposite of $x$. We can then perform $(m-1)/2$ operations $(x-1, x)$, $\ldots$, until only the opposite edge remains; cover it with the last operation.

**Question**: for which trees with an even number of leaves $m$ the problem can be solved in exactly $m/2$ operations?