From Hacks to Snitches (watchmen)

BY TOBIAS LENZ AND LUKAS MICHEL (GERMANY)

Let $L = \ell_1 + \cdots + \ell_K$. If u is a corner on a watchman route, let ℓ_u be the length of that route. Whenever we speak of reaching some corner or moving through the museum, we implicitely imply that you should not be noticed during these actions.

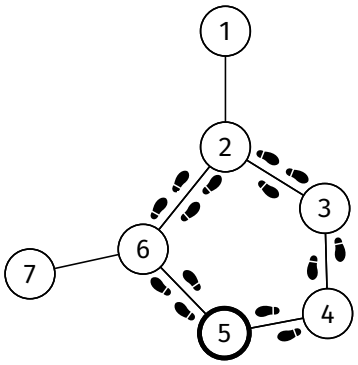
Subtask 1. $N, M \leq 100\,000$, $K = 1$, $\ell_1 \leq 125$

Create a directed graph whose vertices are pairs (u, v_i) where u is your current position and v_i is the current position of the watchman. Then, for a corridor from u to w, add an edge from (u, v_i) to (w, v_{i+1}). Also, add an edge from (u, v_i) to (u, v_{i+1}). Finally, delete all vertices where you and the watchman are at the same position as well as all edges where you and the watchman pass each other in a corridor.

This directed graph represents all possible movements of you and the watchman during which you are not noticed. Hence, we want to compute the shortest path in this graph from $(1, v_1)$ to a vertex of the form (N, v_i). This can be done by a BFS. The complexity of this approach is $O(M\ell_1)$ which suffices to solve Subtask 1. Note that the directed graph from this approach should not be constructed explicitely, instead, its edges should only be constructed on demand.

Subtask 2. $N, M \leq 100\,000$, $\ell_1 + \cdots + \ell_K \leq 125$ and no corridor connects the routes of two distinct watchmen.

Let t_u be the earliest time at which you can reach a corner u without getting noticed. Note that you might want to visit corner u also at a later point in time than t_u. This is because you can sometimes get a shorter path to your target by first letting the watchman pass corner u and then moving to u as soon as possible thereafter. See the following example for $u = 2$.



However, it turns out that if no corridor connects the routes of two distinct watchmen, you never need to visit corner u after time $t_u + L$ because of the following observation:

Lemma. *Under the above conditions, $t_N \leq N + \mathrm{dist}(1, N)$ where dist denotes the distance in the same graph, but without any watchmen.*

Proof. Consider any shortest path P from 1 to N in the graph without watchmen. We now iteratively build a safe route from P as follows: pick any watchman route and consider the first and last time

our current route P hits it. We then replace the actions inbetween as follows: should we run into the watchman the first time we enter his route, we first wait 1 min (note that this is possible because of our assumption that no corridor connect corners on distinct watchmen routes, and only because of it!), otherwise we immediately step onto the watchman route. In both cases, we then simply follow the watchman route (in the usual direction) until we reach the last node of P that is also on our watchman route. Note that this way we will never be noticed by this specific watchman.

We repeat this process for any other watchman route. As no two watchman routes intersect, it is guaranteed that after any iteration, none of the watchmen considered so far will ever notice us. Thus, this process yields a safe route in the end. Moreover, if the watchman route we consider in a given iteration has length ℓ, then the newly added segment on the route has length at most $\ell - 1$ and we wait at most 1 minute. As we consider each watchman route only once, we need at most L additional minutes in total as desired. □

Hence, we can make a BFS which visits each corner at every possible time step, but only as long as the time is still less than $t_u + L$ (this BFS will implicitely compute t_u as the first time it ever reaches corner u). The complexity of this approach is $O(ML)$ because we will consider each corner and therefore also each corridor at most L times, solving Subtask 2.

Subtask 3. $\ell_i \le 200$, $\ell_1 + \cdots + \ell_K \le 350$ and no corridor connects the routes of two distinct watchmen.

Note that for any corner u which is not part of any watchman route, it suffices to know t_u to know all the times during which you can be at corner u. This is because you can simply stay at corner u forever after time t_u since no watchman ever passes this corner. Therefore, our BFS should visit such a corner u at most once, and therefore it can also consider any corridor from some other corner w to u in that direction at most once.

For a corridor in the opposite direction from u to some other corner w, we know that if can reach u at time t_u, we will be able to reach w at all the times $t_u + 1, t_u + 2, \ldots$ except for those times at which w is occupied by a watchman. There are multiple ways to handle this efficiently so that we also need to consider the corridor in the direction from u to w at most once. For example, during our BFS, we can keep a separate list of all those corners on watchman routes which can be reached at all times from the current time step on. Once the BFS visits corner u, we can then add w to this list. During any BFS step, we look whether there are any corners on this list, and if so, add them to the BFS queue for the current time step. Finally, we can remove any corner w from the list after time $t_w + L$.

So, all corridors incident to some corner not part of any watchman route and such corners themselves need to be considered at most once by this modified BFS. Consequently, there remain at most $O(L^2)$ corridors between corners which both lie on watchman routes that have to be considered multiple times, and those corridors will be considered at most L times since any corner is visited at most L times. In total, the complexity of this approach is therefore $O(M + L^3)$, solving Subtask 3.

Subtask 4. No corridor connects the routes of two distinct watchmen.

Let's consider those segments of a path which use corridors belonging to some watchman route separately from those that do not. For this purpose, we will compute (additional to t_u) the earliest time s_u at which you can reach a corner u without being noticed via a path whose last corridor does not belong to any watchman route. Then, we will run Dijkstra on t_u and s_u simultaneously and

update segments of corridors belonging to watchman routes at once while we will update segments of corridors not belonging to watchman routes only corridor-by-corridor.

Assume the Dijkstra has just computed t_u, so we want to update the neighbours of u based on this value. Consider a single corridor not belonging to any watchman route from u to some other corner w which we want to update. Then, if there is no watchman at time $t_u + 1$ at w, you can reach w at time $t_u + 1$ and your last used corridor will not belong to any watchman route. Hence, in that case we have to update $t_w = \min(t_w, t_u + 1)$ and $s_w = \min(s_w, t_u + 1)$. On the other hand, if there is a watchman at time $t_u + 1$ at w, we can simply wait one time step at u and then move to w afterwards. We can do this because the constraints of this subtask guarantee that there are never two watchman at neighbouring corners, and so there cannot be a watchman at corner u at time $t_u + 1$. Hence, we reach w at time $t_u + 2$, and can update t_w and s_w accordingly.

Otherwise, if the Dijkstra has computed s_u, we want to use s_u update the values t_w at once for all corners w reachable from u via segments of corridors belonging to watchman routes. First, note that this only applies if u lies on a watchman route because otherwise there do not exist any such segments starting from u, and for the same reason corner w should be on the same watchman route as u. Consequently, we only have to consider the case $u = v_i$ and $w = v_j$ for the vertices $v_1, \dots, v_{\ell_u}$ of the watchman route of u, and we want to find the shortest possible path from v_i to v_j using only corridors from that route. This is just a matter of case distinctions:

- Either we directly move to w along the route of the watchman in his direction.
- Or we directly move to w in the opposite direction along the watchman route.
- And finally, we can first let the watchman pass corner u and then move to w in that opposite direction. Note that it is actually possible to let the watchman pass corner u because your last used corridor before arriving at u at time s_u did not belong to any watchman route. Hence, while the watchman is at corner u, we can avoid him by temporarily moving back along that corridor.

For all of these three cases, we can compute in constant time whether the corresponding movement is possible without getting noticed and then update t_w accordingly. We will not update s_w because after these movements the last used corridor will belong to a watchman route.

Note that the collection of all updates from t_u will run in time $O(M)$. On the other hand, at most L corners u lie on watchman routes, and for each of them we have to update t_w from s_u for at most L further corners w on the same watchman route of u with all of these updates taking constant time. The total runtime is therefore $O((M + L^2)\log(M + L^2))$ with the logarithm coming from the Dijkstra.

Subtask 5. $\ell_1 + \cdots + \ell_K \le 125$

Assume that u is a corner on a watchmen route of length ℓ_u. From this subtask on, the crucial observation is the following: If you can reach corner u at time t, you can also reach corner u at time $t + \ell_u$. This is because if you are at corner u at time t, you can simply run away from the watchman (frantically screaming)* on his very own route. This will bring you back to corner u in ℓ_u steps without ever crossing the steps of any watchman.

For such corners, we therefore only need to compute, for any $0 \le s < \ell_u$, the earliest time t_u^s at which you can reach corner u such that $t_u^s \equiv s \mod \ell_u$, i.e. the earliest time which has remainder s after division by ℓ_u, as this suffices to know all the times at which you can be at u. For any other corner u

* Depending on your perspective, you might also be *following* the watchman (still frantically screaming).

not on watchman routes, we only compute the earliest time t_u to reach that corner since we will then be able to stay there forever.

To compute these values, we use Dijkstra. If we know the value t_u for a corner u not on watchman routes, we update all adjacent corners w as follows:

- If w is not on watchman routes, update w with time $t_u + 1$.
- If w is on a watchman route, we also update w with time $t_u + 1$, but only if there no watchman occupies w at that time. Moreover, if T is the next time at which a watchman occupies w, we will also update w with the time $T + 1$. We need to do this because we can only stay at w until time $T - 1$, and so if we would only update w with time $t_u + 1$, we would miss the residue classes t_w^s for those s coming directly after time T.

Conversely, if we know t_u^s for a corner u on some watchman route, we update the adjacent corners w as follows:

- If t_u^s is the first time we ever visit corner u, we update all adjacent corner w not on watchman routes with time $t_u^s + 1$.
- We will always update all adjacent corners w belonging to a watchman route. We want to update them with the times $t_u^s + 1, t_u^s + 1 + \ell_u, t_u^s + 1 + 2\ell_u, t_u^s + 1 + 3\ell_u$, and so on. However, note that after at most ℓ_w number of steps, these times will fall again into the same residue class as one of the earlier times. So, we have to update w only with the times $t_u^s + 1, t_u^s + 1 + \ell_u, \dots, t_u^s + 1 + (\ell_w - 1)\ell_u$.
- Finally, we will update u with time $t_u^s + 1$ because we can stay at corner u.

Note any corridor incident to some corner not part of any watchman route and such corners themselves will be considered at most once by this algorithm. Moreover, the algorithm will consider each of the $O(L^2)$ corridors between corners which both lie on watchman routes at most ℓ_u times, and every time it will update w with at most ℓ_w values. Therefore, the complexity of this approach is $O((M+L^4)\log(M+L^4))$, solving Subtask 5.

Subtask 6. $\ell_i \le 200, \ell_1 + \cdots + \ell_K \le 350$

Note that the solution from the previous subtask is somewhat inefficient for a corridor from u to w where u and w are both corners on some watchman route. Namely, for such a corridor we will update w in total with $\ell_u\ell_w$ values even though there are only ℓ_w many residue classes at corner w which need to be updated.

Instead, for the corner u, we can compute for all watchmen routes i the earliest time $t_u^{i,s}$ at which we can reach u such that $t_u^{i,s} \equiv s \mod \ell_i$. Then, once we have computed $t_u^{i,s}$ for the i satisfying $\ell_i = \ell_w$, we can update w with the value $t_u^{i,s} + 1$. So, we need to consider the corridor from u to w only ℓ_w times, reducing total runtime for corridors between corners which both lie on watchman routes to $O(L^3)$.

However, it still remains to compute the times $t_u^{i,s}$. For that, we can actually use the values t_u^s which are computed by our Dijkstra anyway. Once we know t_u^s, we can iterate through all i and update the values $t_u^{i,s'}$ with the times $t_u^s, t_u^s + \ell_u, \dots, t_u^s + (\ell_i - 1)\ell_u$. In total, this amounts to $\ell_1 + \cdots + \ell_K = L$ update steps for a single value t_u^s. Because there are at most L corners on watchman routes, each of which has at most L residue classes, all these updates together take time $O(L^3)$.

Consequently, the complexity of the entire algorithm is then $O((M + L^3)\log(M + L^3))$, solving Subtask 6.

Subtask 7. No further constraints.

The crucial observation for the last subtask is the following: If we visit a corner u on a watchman route at time t_u^s and consider a corridor from u to another corner w which is also on a watchman route, we can actually perform all updates for w in constant time. This works as follows. First, let T be the next time at which w is occupied by a watchman. Then, if there is no watchman at u at time T, we know the optimal times to visit w from u for any residue class of w at once: namely, we can just go from u to w now, wait there until $T - 1$, go back to u (which is safe by assumption), and then to w again, where we wait until time $t_u^s + \ell_w$. You can easily check that even if you visit u again at some later point in time, you will never be able to get to any residue class of w earlier than via this path. Thus, using a similar update as from a node without watchman to w, we can handle all these updates in one step. Even better, we can then also remove this corridor from the graph completely.

Otherwise, if there is a watchman at time T at corner u, let t be the earliest time after T at wich we can again be at corner u with residue class s. Then, we update w with the times $t_u^s + 1$ and $t + 1$, and consider the next time T' at which w is occupied by a watchman after time t. If there is no watchman at u at time T', we already know how to update w in constant time. If there is watchman at u at time T', then it turns out that we have already updated all necessary times for w from t_u^s. This is because if we would again visit u after T' with residue class s, the same situation would arise again, so we would not be able to get to any new residue classes and only get to the already visited residue classes with a higher time.

It remains to estimate the work necessary to handle all updates by this algorithm. First, note that a corridor from u to w is removed from the graph after at most ℓ_w updates. This is because if it takes more than ℓ_w minutes for the watchman to visit u from the current residue class s, then the watchman at the other route will pass corner w before that happens, and so we remove the corridor from u to w right away. Hence, any corridor will therefore be removed after at most ℓ_w updates. Furthermore, we can make the following case distinction:

- Every corridor from the current watchman route to itself will be updated only once and removed immediately thereafter.
- If $\ell_u \geq \ell_w$, consider the first time t_u that you visit corner u. After that time, there will remain at most one edge from u to the watchman route of w. Indeed, assume that the corridor from u to w remains, i.e. there a watchman occupies u at the next time T where a watchman occupies w. For another corner $w' \neq w$ on the watchman route of w, the next time T' that w' is occupied by a watchman will be different and differ by less than ℓ_w from T. So, T and T' cannot be congruent modulo ℓ_u, but we know that the watchman of the current route will be at u at time T. Hence, there cannot be a watchman at u at time T', meaning that the corridor from u to w' will be removed. Letting u vary, at most ℓ_u corridors from the current watchman route to the watchman route of w will remain, and each of them produces at most ℓ_w updates as argued above. This yields a total of $\ell_u \ell_w$ updates.
- If $\ell_u < \ell_w$, then the above argument does not work. However, we analogously see that there at most $\lceil \ell_w/\ell_u \rceil \leq \ell_w/\ell_u + 1$ corridors remaining from u to the watchman route of w, yielding at most $(\ell_w/\ell_u + 1)\ell_u \leq \ell_w + \ell_u \leq 2\ell_w$ edges from the current watchman route to that of w. Since each of them produces at most ℓ_u updates since there are only ℓ_u residue classes at corner u, this yields $2\ell_u \ell_w$ updates in total.

Summing over all ordered pairs i, j of watchmen routes, we get at most $\sum_{i,j} 2\ell_i \ell_j \leq 2 \sum_i \ell_i \sum_j \ell_j = 2L^2$ updates in total for edges between watchmen routes, yielding a total runtime of $O((M + L^2) \log(M + L^2))$.

Some final remarks: The log-factors in the above solutions could be removed by an efficient implementation of Dijkstra which takes advantage of the fact that all times considered by the algorithm will be of the order $O(N + L)$.