CSCI 270 Homework 10

Qilin Ye

December 7, 2022

1. Mountain Lions and *Territorialities*

Solution. We first show that the provided problem, assuming it is phrased as a decision problem (given these lions, does there exist an assignment whose resulting total *territoriality* trouble $\leqslant C$?) is NP. Given a proposed placement of the mountain lions, we can simply brute force iterate through every single node $v \in V$, and if a lion lives at v, iterate through all the neighboring nodes, and see if a lion of the same sex lives at u. If so, we increment the total *territoriality* trouble by t_i corresponding to the lion living at v. We finally compare the total *territoriality* trouble of this assignment to a given threshold, and return TRUE iff it does not exceed the threshold. It is easy to see that we iterate over $|V|$ edges and in each iteration we check at most $|E|$ edges while doing constant time operations for each edge. Therefore the total runtime for verifying such a certificate is polynomial in $|V|$ and $|E|$, showing the decision problem is NP.

To show NP-completeness we reduce the problem from INDEPENDENT SET. Namely, we want to solve INDEPENDENT SET using this lion decision problem as a blackbox. Let $\tilde{G} = (\tilde{V}, \tilde{E})$ and $k \in \mathbb{N}$ be the parameters of an INDEPENDENT SET instance. We hypothetically assume there are k male lions, each with *territoriality* ∞, and ask, "does there exist an enclosure assignment of these lions on $\tilde{G}$ with total *territoriality* trouble $\leqslant$ some constant, say 100?" If the answer to this question is "yes", then the k nodes corresponding to the k assigned enclosures are independent — no edge between any two nodes; conversely, if there exists an independent set of k nodes, we can simply assign each lion to a node while completely avoiding any *territoriality* trouble. Finally, the transformation of parameters is obviously done in polynomial time. QED.

2. Thanksgiving Hugs!

Solution. We first show THANKSGIVING HUGS is NP. Let a certificate, i.e., a sequence of hugs be given. To verify whether this sequence of hugs meet our requirement we maintain an array of size n, where the j^{th} entry, initialized to 0, records the length of hug they receive. We simply traverse through the sequence of hugs and update the array as needed. In the end, for each j, we compare the j^{th} entry in the array with the desired amount h_j, and return TRUE if every guest is satisfied with their hug. Since by assumption hugs cannot be interrupted or resumed there can be at most n hugs in the sequence, and so the verification of the certificate clearly terminates in polynomial time.

To show NP-completeness we reduce THANKSGIVING HUGS from SUBSET SUM. Given a SUBSET SUM instance (numbers $\{x_i\}_{i=1}^n \subset \mathbb{R}_{\geqslant 0}$ and a target sum $\xi \in \mathbb{R}_{\geqslant 0}$), we define $C = \sum_{i=1}^n x_i$ and define n normal guests: for $1 \leqslant j \leqslant n$, let normal guest j have $h_i = x_i$, $s_i = 0$, and $f_i = C + 1$. Finally, we define a special guest with hug requirement 1 such that the hug must take place precisely on $[\xi, \xi + 1]$.

If there exists a subset $S \subset \{x_i\}_{i=1}^n$ with sum ξ, then the THANKSGIVING HUGS is also satisfiable. We hug all the normal guests corresponding to values in S first, without any rest in-between any hugs. When we are done, the total time elapsed is ξ, and we immediately hug the special guest. Starting from time $\xi + 1$, we hug all the remaining guests in any order. Conversely, if THANKSGIVING HUGS is satisfiable, we must have hugged the special guest from time ξ to $\xi + 1$. All n normal guests request a total of $\geqslant C$ time for hugs. Since each of them needed to be hugged before time $C + 1$ and we have spent one unit time on the special guest, there is no leeway. That is, we must be constantly hugging guests from time 0 to ξ and also from time ξ to $C + 1$. By assumption hugs are not interrupted or resumed, so the guests we hugged before the special one must correspond to a subset T of $\{x_i\}_{i=1}^{(n)}$ whose sum is precisely ξ. Finally, the transformation of parameters is done in polynomial time. This completes the proof.

3. City Transportation System

Solution. The decision problem simply asks, "given a graph $G = (V, E)$ representing the city's current transportation system, does there exist a set E' of additional edges such that (i) $|E'| \leqslant$ some prescribed constant K and (ii) all nodes $v \in V$ are reachable from s using at most two edges in $E \cup E'$?"

It is quite trivial to show this problem has an efficient certifier: given any proposed set of additional edges E', run BFS on all nodes in V with root s, and return FALSE if and only if some $v \in V$ is at least 3 away from s. As BFS runs in polynomial time, so does this certifier.

To show this decision problem is NP-complete we reduce from 3SAT. Given an instance of 3SAT, say variables $\{x_i\}_{i=1}^n$ and 3-CNF clauses $C_1, ..., C_m$, we set the cap on number of additional edges to be n and construct a graph $G = (V, E)$ as follows:

(1) Add $3n + m + 1$ nodes to V:

- one node s representing the central station,
- n "literal nodes" $v_1, ..., v_n$ corresponding to $x_1, ..., x_n$, and n nodes $\overline{v}_1, ..., \overline{v}_n$ to the negations $\overline{x}_1, ..., \overline{x}_n$,
- n "dummy nodes" $u_1, ..., u_n$, and
- m "clause nodes" $c_1, ..., c_m$ corresponding to the m clauses $C_1, ..., C_m$.

(2) Add $3m + 3n$ edges to E (note s is isolated):

- for each $i \in \{1, ..., n\}$, add edges $(v_i, \overline{v}_i), (u_i, v_i)$, and $(u_i, \overline{v}_i)$, and
- for each c_i, add an edge connecting it to each literal in clause C_i; for example if $C_1 = (x_2 \vee \overline{x}_3 \vee x_5)$, add three edges $(c_1, v_2), (c_1, \overline{v}_3)$, and (c_1, v_5).

Let x be a complex number such that $Bx = x$. Then,

$$x = Bx = \lim_{n\to\infty} A^n x = \lim_{n\to\infty} \left(A^{n-1} \cdot Ax\right) = \lim_{n\to\infty} \left(A^{n-2} \cdot A^2 x\right)$$
$$= \cdots = \lim_{n\to\infty} \left(A^2 x\right) = \lim_{n\to\infty} (A \cdot Ax) = \lim_{n\to\infty} \left(A^2 x\right) = \cdots = \lim_{n\to\infty} (A^n x) = Bx$$

Thus, $Bx = x$ for all complex numbers x. In other words, B is the identity matrix, so $B^2 = B$.

We say a node v is 2-step reachable if it can be reached from s within $\leqslant 2$ steps.

If the original 3SAT is satisfiable, let ν be a satisfying assignment. For each $i \in \{1, ..., n\}$ we connect (s, v_i) if $\nu(x_i) = 1$ and we connect $(s, \overline{v}_i)$ otherwise. In total we add n edges. Immediately, all literal nodes are 2-step

reachable (either an edge from s to itself or an edge from s to its negation followed by an edge between its negation and itself). Similarly, for each dummy node u_i, either v_i or $\overline{v}_i$ is directly reachable from s, so u_i is 2-step reachable. Finally, since ν satisfies the clauses, for each clause node c_j, at least one of its three outgoing edges connects to a literal node that is directly reachable from s, and hence all c_i's are also 2-step reachable.

Conversely, suppose there exists a set E' of edges, $|E'| \leqslant n$, that makes every node v 2-step reachable. Now what can these additional edges look like?

(1) An edge of form (s, u_i) (to dummy) ensures that both v_i and $\overline{v}_i$ are now reachable in 2 steps, but we can do better by simply connecting (e, v_i) or $(e, \overline{v}_i)$ — this means we can WLOG assume no edge is of form (s, u_i).

(2) In order for u_i to be 2-step reachable, we need to connect s with either v_i or $\overline{v}_i$. Since we can only add n edges and there are n dummy nodes to take care of, each newly added edge must be of form (s, v_i) or $(s, \overline{v}_i)$, and that different edges must correspond to different indices.

(2) implies that E' corresponds to a truth assignment (set all literals whose corresponding literal nodes is connected to s as TRUE)! Since by assumption each clause node c_i is also 2-step reachable, for each clause node c_i there exists a 2-step path, say $s \to$ some literal node $\to c_i$. By definition the corresponding literal is assigned TRUE and the clause C_i is satisfied. Since this holds for all clause node, we obtain a satisfying assignment for our 3SAT instance. This completes the reduction from 3SAT to our decision problem. It is clear that transforming $(3n + m + 1)$ nodes and $(3m + 3n)$ edges is done in polynomial time, so we are done.

4. ????

Solution. We reduce from HALT. Clearly, the following program correctly outputs a MST of G if and only if P terminates (if P does not, the program won't return anything). Kruskal's algorithm runs in polynomial time w.r.t. G, and we are done.

```
edge_set Q (graph_with_edge_costs G) {
     Run P(P);
     return Kruskal(G);
}
```