

1. A clique in a graph $G = (V, E)$ is a set $C \subseteq V$ such that $\forall u \neq v \in C, \{u, v\} \in E$. Show that the following language CLIQUE is in **NP**:

$$\text{CLIQUE} = \{\langle G, k \rangle \mid G \text{ has a clique of size at least } k\}$$

Assume that graphs are represented by their adjacency matrices.

Solution: Define an NTM M that will decide CLIQUE in polynomial time: $M =$ “on input $\langle G, k \rangle$...

1. Guess a subset $C \subseteq V$
2. If $|C| < k$ then reject
3. For each pair $u \neq v \in C$: if $\{u, v\} \notin E$ then reject
4. Accept

We need to show that M runs in polynomial time. Let $n = |V|$ be the number of vertices of the input graph (the input size is n^2). The first two steps take approximately $O(n)$ time. The loop requires checking $O(n^2)$ pairs, and each check takes time $O(n^2)$ to search through the matrix for the correct entry. So the total running time is $O(n^4)$, which is polynomial in the input size. Note that we are ignoring a small amount of time necessary for “overhead”, for such things as incrementing counters and so on. This may add a factor of $O(\log n)$ or even $O(n)$ to the running time, but will not affect the “polynomiality” of it.

Now we need to show that $L(M) = \text{CLIQUE}$. Recall that an NTM accepts an input iff it there exists an accepting computation (among the many possible computations) on that input. So we need to show that M has an accepting computation on input $\langle G, k \rangle$ iff $\langle G, k \rangle \in \text{CLIQUE}$. If $\langle G, k \rangle \in \text{CLIQUE}$ then G has a clique C with $|C| \geq k$, so there exists an accepting computation of M on input $\langle G, k \rangle$: the computation where it correctly guesses C . On the other hand, if $\langle G, k \rangle$ is accepted by M then M found a clique of size at least k in G , implying that $\langle G, k \rangle \in \text{CLIQUE}$. So $L(M) = \text{CLIQUE}$.

2. Show that the following language is in **NP**:

$$L = \{\langle M, w, 1^t \rangle \mid M \text{ is an NTM, } M \text{ has an accepting computation of length } \leq t \text{ on input } w\}$$

Solution: The idea is to define a non-deterministic Turing machine that M_L that guesses a computation of the input machine and checks that it is indeed an accepting computation. Define $M_L =$ “on input $\langle M, w, 1^t \rangle$...

1. Let B be the maximum size of $|\delta(q, a)|$, where δ is M 's transition function
2. Guess a sequence of numbers $b_1, b_2, \dots, b_t \in \{1, \dots, B\}$

3. Simulate M on input w for t steps, using the numbers in the guessed sequence to decide which of the possible transitions M should make at each step, as follows: if M is in state q and is scanning a symbol a , let $B' = |\delta(q, a)|$. $B' \leq B$ is the number of possible moves that M could make at this point. If the next b_i in the guessed sequence is no larger than B' , then choose the b_i -th move from the available choices; otherwise reject.
4. If M is in an accepting configuration, accept; else reject

What is the time complexity of M_L ? To simulate a step of M , it must scan M 's transition table and update the work-tape accordingly. Since M 's transition table is part of the input, the first part takes linear time. To update the work-tape, M_L might have to move through the entire tape; but as only t steps are being simulated, M can have at most t non-blanks on its tape, so this takes $O(t)$ time. So simulating a step takes $O(n)$ time, where n is the input size, and as only $t \leq n$ steps are simulated, the total running time is $O(n^2)$.

If $\langle M, w, 1^t \rangle \in L$ then M has an accepting computation of length at most t on input w , so there exists a good sequence b_1, \dots, b_t corresponding to the transitions M uses in this accepting computation. Thus M_L has an accepting computation on input $\langle M, w, 1^t \rangle$, where it correctly guesses b_1, \dots, b_t .

On the other hand, if M_L has an accepting computation on input $\langle M, w, 1^t \rangle$ then M has an accepting computation of length at most t on input w , since M_L only accepts if it discovers such a computation. So $L(M_L) = L$.

Notice that in the definition of L , it is important that the third component of the input is 1^t (i.e. a sequence of t ones), rather than a binary encoding of the number t . The reason is that, if t were represented in binary, then the representation of t would require only $\log t$ bits, and simulating t steps would require time exponential in the input size. The notation 1^t is called a "unary" encoding of the number t .