1 Proving NP-completeness

In general, proving \( NP \)-completeness of a language \( L \) by reduction consists of the following steps.

1. Show that the language \( A \) is in \( NP \)
2. Choose an \( NP \)-complete \( B \) language from which the reduction will go, that is, \( B \leq_p A \).
3. Describe the reduction function \( f \)
4. Argue that if an instance \( x \) was in \( B \), then \( f(x) \in A \).
5. Argue that if \( f(x) \in A \) then \( x \in B \).
6. Briefly explain why is \( f \) computable in polytime.

Usually the bulk of the proof is 2a, we often skip 1 and 1d when they are trivial.

2 Some examples of NP-completeness reductions

2.1 Hamiltonicity problems

Definition 1. A Hamiltonian cycle (path, s-t path) is a simple cycle (path, path from vertex \( s \) to vertex \( t \)) in an undirected graph which touches all vertices in the graph. The languages HamCycle, HamPath and stHamPath are sets of graphs which have the corresponding property (e.g., a hamiltonian cycle).

We omit the proof that HamPath is \( NP \)-complete (see Sipser’s book page 286). Instead, we will do a much simpler reduction. Assuming that we know that HamCycle is \( NP \)-complete, we will prove that stHamPath is \( NP \)-complete. It is easy to see that all problems in this class are in \( NP \): given a sequence of \( n \) vertices one can verify in polynomial time that no vertex repeats in the sequence and there is an edge between every pair of subsequent vertices.

Lemma 2. HamCycle \( \leq_p \) stHamPath

Proof. Let \( f(G) = (G', s, t) \) be the reduction function. Define it as follows. Choose an arbitrary vertex of \( G \) (say, labelled \( v \)). Suppose that there is no vertex in \( G \) called \( v' \). Now, set vertices of \( G' \) to be \( V' = V \cup \{v'\} \), and edges of \( G' \) to be \( E' = E \cup \{(u, v') \mid (u, v) \in E\} \).

That is, the new vertex \( v' \) is a “copy” of \( v \) in a sense that it is connected to exactly the same vertices as \( v \). Then, set \( s = v \) and \( t = v' \).
Now, suppose that there is a Hamiltonian cycle in $G$. Without loss of generality, suppose that it starts with $v$, so it is $v = v_1, v_2, \ldots, v_n, v$. Here, it would be more correct to use numbering of the form $v_{i_1}, \ldots, v_{i_n}$, but for simplicity we assume that the vertices are renumbered. Now, replacing the final $v$ with $v'$ we get a Hamiltonian path from $s = v$ to $t = v'$ in $G'$.

For the other direction, suppose that $G'$ has a Hamiltonian path starting from $s$ and ending in $t$. Then since $s$ and $t$ correspond to the same vertex in $G$, this path will be a Hamiltonian cycle in $G$.

Lastly, since $f$ does no computation and only adds 1 vertex and at most $n$ edges the reduction is polynomial-time.

Note that this reduction would not work if we were reducing to HamPath rather than stHamPath. Then the part 1c of the proof would break: it might be possible to have a Hamiltonian path in $G'$ but not a Hamiltonian cycle in $G$ if we allow $v$ and $v'$ to be in different parts of the path.

**Definition 3 (Travelling salesperson problem).** For TSP, consider an undirected graph in which all possible edges $\{u, v\}$ (for $u \neq v$) are present, and for which we have a nonnegative integer valued cost function $c$ on the edges. A tour is a simple cycle containing all the vertices (exactly once) – that is, a Hamiltonian cycle – and the cost of the tour is the sum of the costs of the edges in the cycle.

**TSP Instance:** 
$\langle G, c, B \rangle$ where $G$ is an undirected graph with all edges present, $c$ is a nonnegative integer cost function on the edges of $G$, and $B$ is a nonnegative integer.

**Acceptance Condition:**
Accept if $G$ has a tour of cost $\leq B$.

**Theorem 4.** TSP is NP-Complete.

*Proof.* It is easy to see that TSP $\in$ NP.

We will show that HamCycle $\leq_p$ TSP.

Let $\alpha$ be an input for HamCycle, and as above assume that $\alpha$ is an instance of HamCycle, $\alpha = \langle G \rangle$, $G = (V, E)$. Let 
$f(\alpha) = \langle G', c, 0 \rangle$ where:

$G' = (V, E')$ where $E'$ consists of all possible edges $\{u, v\}$;

For each edge $e \in E'$, $c(e) = 0$ if $e \in E$, and $c(e) = 1$ if $e \notin E$.

It is easy to see that $G$ has a Hamiltonian cycle $\iff G'$ has a tour of cost $\leq 0$. \qed

Note that the above proof implies that TSP is NP-complete, even if we restrict the edge costs to be in $\{0, 1\}$.  

2
2.2 SubsetSum and Partition

SubsetSum

Instance:
\(\langle a_1, a_2, \cdots, a_m, t \rangle\) where \(t\) and all the \(a_i\) are nonnegative integers presented in binary.

Acceptance Condition:
Accept if there is an \(S \subseteq \{1, \cdots, m\}\) such that \(\sum_{i \in S} a_i = t\).

We will postpone the proof that SubsetSum is NP-complete until the next lecture. For now, we will give a simpler reduction from SubsetSum to a related problem Partition.

\textbf{PARTITION}

Instance:
\(\langle a_1, a_2, \cdots, a_m \rangle\) where all the \(a_i\) are nonnegative integers presented in binary.

Acceptance Condition:
Accept if there is an \(S \subseteq \{1, \cdots, m\}\) such that \(\sum_{i \in S} a_i = \sum_{j \notin S} a_j\).

\textbf{Theorem 5.} \textit{PARTITION} is NP-Complete.

\textit{Proof.} It is easy to see that PARTITION \(\in\) NP.
We will prove SubsetSum \(\leq_p\) PARTITION. Let \(x\) be an input for SubsetSum. Assume that \(x\) is an Instance of SubsetSum, otherwise we can just let \(f(x)\) be some string not in PARTITION. So \(x = \langle a_1, a_2, \cdots, a_m, t \rangle\) where \(t\) and all the \(a_i\) are nonnegative integers presented in binary. Let \(a = \sum_{1 \leq i \leq m} a_i\).

\textbf{Case 1:} \(2t \geq a\).
Let \(f(x) = \langle a_1, a_2, \cdots, a_m, a_{m+1} \rangle\) where \(a_{m+1} = 2t - a\). It is clear that \(f\) is computable in polynomial time. We wish to show that \(x \in \text{SubsetSum} \iff f(x) \in \text{PARTITION}\).

To prove \(\Rightarrow\), say that \(x \in \text{SubsetSum}\). Let \(S \subseteq \{1, \cdots, m\}\) such that \(\sum_{i \in S} a_i = t\). Letting \(T = \{1, \cdots, m\} - S\), we have \(\sum_{j \in T} a_i = a - t\). Letting \(T' = \{1, \cdots, m+1\} - S\), we have \(\sum_{j \in T'} a_i = (a - t) + a_{m+1} = (a - t) + (2t - a) = t = \sum_{i \in S} a_i\). So \(f(x) \in \text{PARTITION}\).

To prove \(\Leftarrow\), say that \(f(x) \in \text{PARTITION}\). So there exists \(S \subseteq \{1, \cdots, m+1\}\) such that letting \(T = \{1, \cdots, m+1\} - S\), we have \(\sum_{i \in S} a_i = \sum_{j \in T} a_j = [a + (2t - a)]/2 = t\). Without loss of generality, assume \(m+1 \in T\). So we have \(S \subseteq \{1, \cdots, m\}\) and \(\sum_{i \in S} a_i = t\), so \(x \in \text{SubsetSum}\).

\textbf{Case 2:} \(2t \leq a\). You can check that adding \(a_{m+1} = a - 2t\) works. \(\square\)

\textbf{Warning:} Students often make the following serious mistake when trying to prove that \(L_1 \leq_p L_2\). When given a string \(x\), we are supposed to show how to construct (in polynomial time) a string \(f(x)\) such that \(x \in L_1\) if and only if \(f(x) \in L_2\). We are supposed to construct \(f(x)\) without knowing whether or not \(x \in L_1\); indeed, this is the whole point. However, often students assume that \(x \in L_1\), and even assume that we are given a certificate showing that \(x \in L_1\); this is completely missing the point.