15-455: UCT	**K. Sutner**
Assignment 3	Due: **February 11**, 2022.

1. Kleene Star (20)

Background

For any language $L \subseteq \Sigma^\star$, recall the definition of the Kleene star

$$L^\star = \{\, x_1 x_2 \ldots x_k \mid k \geq 0, x_i \in L \,\}$$

Define the marked Kleene star as

$$L^\star_{\#} = \{\, x_1 \# x_2 \# \ldots \# x_k \mid k \geq 0, x_i \in L \,\}$$

where # is a new symbol not in Σ.

It is well-known that $L^\star_{\#}$ and $L^\star$ are regular whenever L is regular and there is a simple algorithm to construct the corresponding finite state machines. Here is an analogous result for L being polynomial time.

Task

Assume that $L \in \mathbb{P}$.

A. Show that $L^\star_{\#}$ is in $\mathbb{P}$.

B. Show that $L^\star$ is in $\mathbb{P}$.

Comment

Don't try to argue directly in terms of Turing machines for this; give a reasonable recognition algorithm for the two languages and then make sure that the algorithm could be implemented on a Turing machine with only a polynomial slow-down.

2. Linear Time Reductions (20)

Background

A is linear time reducible to B if there is a linear time computable function $f : \Sigma^\star \to \Sigma^\star$ such that $x \in A \iff f(x) \in B$. One can show that this is a pre-order, so we can consider the corresponding equivalence relation $\equiv_{\mathsf{lin}}$, producing the linear time degrees.

Notation: $A \leq_{\mathsf{lin}} B$ and $A \equiv_{\mathsf{lin}} B$.

Recall the standard decision problems from graph theory:

Problem:	**Vertex Cover** (VC)
Instance:	A ugraph G, a bound k.
Question:	Does G have a vertex cover of cardinality k?

Problem:	**Independent Set** (IS)
Instance:	A ugraph G, a bound k.
Question:	Does G have an independent set of cardinality k?

Problem:	**Clique** (CL)
Instance:	A ugraph G, a bound k.
Question:	Does G have a clique of cardinality k?

Assume that the graph is given as an $n \times n$ Boolean matrix, k is written in binary. So the size of an instance (G, k) is $n^2 + \log n = \Theta(n^2)$ where n is the number of vertices in G.

Task

A. Show that linear time reducibility is a pre-order.

B. Show that $A \leq_{\mathsf{lin}} B$, $B \in \mathrm{TIME}(n^k)$ implies $A \in \mathrm{TIME}(n^k)$.

C. Show that $\mathsf{VC} \equiv_{\mathsf{lin}} \mathsf{IS} \equiv_{\mathsf{lin}} \mathsf{CL}$.

D. What would happen if we used an adjacency list representation instead?

3. Collapsing Time (20)

Background

The Hartmanis/Stearns time hierarchy theorem says, in essence, that if g is smaller than f, then TIME(g) is properly contained in TIME(f). Recall that we generally require our machines to read the whole input, so there are no sub-linear time complexities.

Here is what happens when we drop that entirely reasonable condition.

Task

A. Cheat a little and use uniform cost. Find a natural arithmetical algorithm that, on input n, runs approximately in time $O(\sqrt{n})$.

B. Show that TIME($\sqrt{n}$) = TIME(1).

C. Can you push the last result a bit?

Comment

For part (A) you should argue in terms of actual algorithms, don't bother with Turing machines. But for part (B) you need to cope with the definition of TIME, and thus with Turing machines—but don't get bogged down.

4. One-One Reductions (40)

Background

In class we introduced the notion of many-one reduction: $A \leq_m B$ if there is a computable function f such that $x \in A \iff f(x) \in B$. This is perhaps the most natural definition, but there is a slightly more constrained form: one-one reductions where f is additionally required to be injective; in symbols $A \leq_1 B$. It is true that $A \leq_m B$ does not imply $A \leq_1 B$, but we won't go there.

In fact, very often a many-one reduction can be translated into a one-one reduction by exploiting the fact that our standard enumeration $(\mathcal{M}_e)$ of all Turing machines is repetitive in the sense that for each e there are infinitely many e' such that for all x: $\{e\}(x) \simeq \{e'\}(x)$. Recall that we cannot filter out just the minimal Turing machines.

For any two sets $A, B \subseteq \mathbb{N}$ define their disjoint union to be the set

$$A \oplus B = \{\, 2x \mid x \in A \,\} \cup \{\, 2x+1 \mid x \in B \,\}$$

For a reduction $\preceq$, disjoint union often plays the role of a least upper bound (lub): $A, B \preceq A \oplus B$, moreover $A, B \preceq C$ implies $A \oplus B \preceq C$.

As always, K denotes the Halting set and FIN the set of all indices of finite semidecidable sets:

$$K = \{\, e \mid \{e\}(e) \downarrow \,\}$$
$$\mathsf{FIN} = \{\, e \mid W_e \text{ finite} \,\}$$

Also let

$$\mathsf{COF} = \{\, e \mid W_e \text{ cofinite} \,\}$$

Task

A. Justify the claim about the infinitely many equivalent Turing machines functions from above.

B. Show that disjoint union is a lub for Turing reducibility.

C. Show that disjoint union is a lub for many-one reducibility.

D. This does not work for one-one reducibility. Can you see what goes wrong?

E. Show that K is one-one reducible to FIN.
Hint: consider an enumeration K_s of K in stages.

F. Show that FIN is one-one reducible to COF.
Hint: consider the stage when an element is enumerated into W_e.

Extra Credit: For any index e define

$$\mathsf{EQ}_e = \{\, e' \mid W_e = W_{e'} \,\}$$

Note that EQ_e is always infinite (why?). Show that $\mathbb{N} - K$ is one-one reducible to EQ_e for any e.

Comment

For all these claims, first come up with an intuitive reason why the claim ought to be true–then formalize your intuition.