University of Sussex
Informatics

Spring 2020

Limits of Computation
Feedback for Exercises 7
Dr Bernhard Reus

Time Bounds and Complexity Classes
(Lectures 12-17)

1. Consider the following program:

```
prog read X {
 Y:= cons X X
}
write Y
```

   (a) Give a time bound for program for `p`.
   Answer: On Sheet 5 we already computed that $\text{time}_{\texttt{p}}^{\texttt{WHILE}}(d) = 2 + (3 + 1) = 6$. we observe that the runtime is actually constant and does not depend on the input. A time bound gives for any input size an upper bound of the runtime for all inputs of that size. So any function $f : \mathbb{N} \to \mathbb{N}$ with $f(n) = c$ with $c \geq 6$ works as answer.

   (b) In which of the following classes is program `p`?
      i. $\texttt{WHILE}^{\text{time}(2n)}$
      Answer: No, as it is not the case that $6 \leq 2n$ for $n = 1$ (recall that our time bounds are *worst case*).
      ii. $\texttt{WHILE}^{\text{time}(4n)}$
      Answer: No, as it is not the case that $6 \leq 4n$ for $n = 1$ (recall that our time bounds are *worst case*).
      iii. $\texttt{WHILE}^{\text{lintime}}$
      Answer: Yes, as `p` is in $\mathbf{TIME}(f)$ where $f(n) = 6n$. It is important to observe here that we don't have any input for which the size is 0. If it were, we would not have that $6 \leq 6 \times 0 = 0$.

iv. $\texttt{WHILE}^{\text{ptime}}$

Answer: Yes, as above. Any linear function is also a polynomial.

2. In Lecture 13, Slide 14 we claimed that

$$\texttt{L}_1 \preceq^{\text{lintime}} \texttt{L}_2 \text{ implies that } \mathbf{LIN}^{\texttt{L}_1} \subseteq \mathbf{LIN}^{\texttt{L}_2}$$

Let us prove this now. In other words, assuming that
(1) $\texttt{L}_1 \preceq^{\text{lintime}} \texttt{L}_2$ and
(2) $A \in \mathbf{LIN}^{\texttt{L}_1}$
show that also (3) $A \in \mathbf{LIN}^{\texttt{L}_2}$.

Hint: Unfold the definitions of the assumptions (1) and (2) and the proof goal (3), then rearrange the assumptions so that you see that you already have (3).

Answer:
$\mathbf{LIN}^{\texttt{L}}$ is the class of all problems of words/strings containing of 0 and 1 that are decidable by a L program that has a running time bounded by a linear function. Formally,

$$\mathbf{LIN}^{\texttt{L}} = \{A \subseteq \{0,1\}^* \mid A \text{ is decided by a program in } \texttt{L}^{\mathit{lintime}}\}$$

$\texttt{L}_1 \preceq^{\text{lintime}} \texttt{L}_2$ means that language $\texttt{L}_2$ can simulate language $\texttt{L}_1$ up to linear time difference .
This means that for any $\texttt{L}_1$ program p there is a $\texttt{L}_2$ program q with exactly the same behaviour but for runtime of q we can guarantee only a time bound that is up to a constant (program dependent) factor the time bound of the original program p. The latter, in turn, means for all input data d we have

$\text{time}^{\texttt{L}_2}_{\texttt{q}}(\texttt{d}) \leq a_{\texttt{p}} \times \text{time}^{\texttt{L}_1}_{\texttt{p}}(\texttt{d}))$

Let us show that $\texttt{L}_1 \preceq^{\text{lintime}} \texttt{L}_2$ implies that $\mathbf{LIN}^{\texttt{L}_1} \subseteq \mathbf{LIN}^{\texttt{L}_2}$.

So we assume that $\texttt{L}_1 \preceq^{\text{lintime}} \texttt{L}_2$. (Assumption 1) We have to show that for an arbitrary problem (set) $A \in \mathbf{LIN}^{\texttt{L}_1}$ (Assumption 2) that it is also in $\mathbf{LIN}^{\texttt{L}_2}$, i.e. $A \in \mathbf{LIN}^{\texttt{L}_2}$ (3).

The latter means we need to show that there is a $\texttt{L}_2$ program q that decides A in linear time. How do we get this program? Well by Assumption 2 we have a $\texttt{L}_1$ program p that decides A in linear time. From Assumption 1 we know that we can simulate $\texttt{L}_1$ programs by $\texttt{L}_2$

programs, which increases the runtime by at most a constant (linear) factor. In other words, the given program $\mathtt{p}$ can be compiled into a $\mathtt{L}_2$ program $\mathtt{q}$ with the same behaviour, which means $\mathtt{q}$ also decides A. We're halfway there as we have found our decision procedure $\mathtt{q}$. We still have to show that its runtime is linearly bounded. But we know by Assumption 1 that:

$$\text{time}_{\mathtt{q}}^{\mathtt{L}_2}(\mathtt{d}) \leq a_{\mathtt{p}} \times \text{time}_{\mathtt{p}}^{\mathtt{L}_1}(\mathtt{d}))$$

and by Assumption 2 that

$$\text{time}_{\mathtt{p}}^{\mathtt{L}_1}(\mathtt{d}) \leq c \times |\mathtt{d}|$$

for some constant c. So from the latter inequality we can infer that

$$a_{\mathtt{p}} \times \text{time}_{\mathtt{p}}^{\mathtt{L}_1}(\mathtt{d})) \leq a_{\mathtt{p}} \times (c \times |\mathtt{d}|)$$

Now from the last and the first inequality, by transitivity of the ordering $\leq$, we get that

$$\text{time}_{\mathtt{q}}^{\mathtt{L}_2}(\mathtt{d}) \leq a_{\mathtt{p}} \times (c \times |\mathtt{d}|)$$

which by associativity of multiplication can be written as :

$$\text{time}_{\mathtt{q}}^{\mathtt{L}_2}(\mathtt{d}) \leq (a_{\mathtt{p}} \times c) \times |\mathtt{d}|$$

which completes the proof.

3. For which thesis do we have sufficient evidence and which?

   (a) **LIN** is robust and does not depend on the sequential notion of computation used.
   Answer: It is not very robust as compiling e.g. from `WHILE` to `TM` produces a polynomial slow-down leading out of **LIN**. So ths is clear evidence *against* the above thesis.

   (b) **P** is robust and does not depend on the sequential notion of computation used (Cook's Invariance Thesis).
   Answer: Yes, we have evidence for this thesis, as we can compile between all sequential notions of computation and the slow-down is always at most polynomial, thus not leading out of **P**.

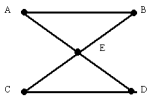
(c) **P** is the class of all feasible (= practically decidable) problems (Cook-Karp Thesis).

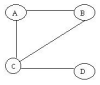
Answer: There is weak evidence for this thesis, as there are specific examples that contradict it. Problems that can be solved only in polynomial time where the polynomial has a large degree are not really feasible, although in practice natural problems that are known to be in **P** usually have polynomial bounds of low degree (up to 3 or 4, but this can already be problematic for large inputs). On the other hand there are algorithms with exponential worst case complexity that run reasonably well on data used in practice. Thus one can say that **P** is a good rule-of-thumb description of feasible problems, but it surely does *not* describe *exactly* the feasible ones.

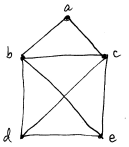
4. Consider the *Eulerian Circuit Problem* (also Postman Problem): "For a given (undirected) graph, is there a round trip that visits each edge exactly once?"

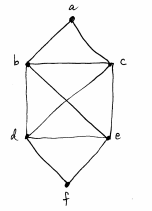
(a) For which of the four graphs (a-e) given below does such a tour starting, say, in vertex A exist?

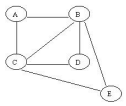
Answers:

a YES 

b NO 

c NO 

d YES 

e YES 

(b) Consider the graphs from question (a) for which a tour as described exists. What do they have in common?

Answer:

All vertices have an even degree (that's the number of edges going in/out). In the Yes cases above the degree of any vertex is either 2 or 4

(c) Can you guess Euler's theorem that states the condition on which a tour (in connected graphs) exists? (look at the edges!)

Answer:
If every vertex of a connected graph has even degree then the graph admits an Eulerian circuit.