Notes on Sipser: § 7.3, “The Class NP”
CSC 341, “Automata, Formal Languages, and Computational Complexity”
Department of Computer Science · Grinnell College
December 9, 2020

“We don’t know the answer to this important question”: In other words, congratulations—you have now reached the current boundary of established theoretical computer science. This is the point at which, we hope, you and the other members of the next generation of computer scientists will discover things that no one yet has been able to figure out.

• In the description of the nondeterministic Turing machine $N_1$ (page 294), the one that decides the HAMPATH problem in polynomial time, skeptical readers might be inclined to question Sipser’s observation that step 1, the generation of the list of node numbers, “clearly runs in polynomial time,” since it results in the generation of a total of $m^m$ lists, each with $m$ elements. In a nondeterministic machine, though, each of these lists is on a different branch of computation, and the computation on that branch requires only $O(m)$ steps. Nondeterminism allows the machine, in effect, to make $m$ copies of itself at each choice point, producing a total of $m^m$ copies at the end of step 1, each examining a different list of nodes from the directed graph. The parallelism allows the computation to run in $O(m)$ time.

• The proof of Theorem 7.20 is not completely constructive. In the “forward” direction, we’re given the verifier $V$ and that it runs in polynomial time, but there may not be any straightforward way for us to discover what the exact time-complexity function of $V$ is. Sipser just assumes that it’s $n^k$, but we don’t actually need that assumption, which is fortunate, because we’re not actually entitled to make it. We can just name the time-complexity function $t$ without having to know what it is or anything about it except that it is $O(n^k)$ for some natural number $k$. We do get to assume that much.

Now we can write stage 1 in the prose description of $N$ at the top of page 295 as “Nondeterministically select string $c$ of length at most $t(n)$.” Since $t(n) \in O(k)$, the construction of the string $c$ is completed in polynomial time, as required.

Sipser’s use of the time-complexity function $n^k$ as a kind of stand-in for an arbitrary member of $O(n^k)$ may be just a convenient shorthand, but I find it confusing and sloppy.

• A multiset (sometimes called a bag) is a data structure that is unordered, like a set, but may contain duplicate elements, like a list. In testing whether one multiset $A$ is a “subcollection” of another multiset $B$, one must confirm not only that every element of $A$ is also an element of $B$, but also that the number of copies of an element $x$ of $A$ is less than or equal to the number of copies of $x$ in $B$.

For the SUBSET-SUM problem, this means that an addend selected from the given multiset can be used more than once in forming the target sum, but the number of times it is used may not exceed the number of copies of that value in the multiset.

For example, you can form the target sum 34 from the multiset $\{1, 3, 3, 9, 27, 81\}$, since $34 = 27 + 3 + 3 + 1$ and there are two 3s in the multiset; but you can’t form the target sum 49, even though $49 = 27 + 9 + 9 + 3 + 1$, because there is only one 9 in the multiset.
It’s important to be clear on the exact statement of the SUBSET-SUM problem, because the use of duplicate elements is an essential technique in the proof of a later theorem (Theorem 7.56).

Of course, the fact that “most researchers believe” that $P \neq NP$ (as Sipser says on page 298) has no probative value. The history of mathematics and computer science is littered with now-forgotten conjectures that were once, in the absence of proof, believed by most researchers to be true, but were subsequently disproven.

Skeptics point out that the number of polynomial-time algorithms is infinite and suggest that the reason we haven’t found polynomial-time algorithms for some of the hard problems in NP is that the algorithms are too long and intricate to be discovered or formulated in a human lifetime, or that their time-complexity functions are of such high degrees that inquirers instinctively recognize them as infeasible and never seriously consider them, or that they presuppose such unfamiliar, alien, or deeply counterintuitive ways of thinking about problems that they are undiscoverable for psychological reasons. Of course, this line of argument has no probative value, either. (The history of mathematics and computer science is also littered with lame excuses for not accomplishing certain goals, some of which later turned out to be mathematically impossible to achieve.)