Computational Classes of Problems

For each of these problems, or languages, give its best known computational class. For example, the answer could be \( P \), \( \mathcal{NP} \), \( \mathcal{NP} \)-complete, \( \mathcal{P}\text{-SPACE} \), recursive, recursively enumerable, to name just a few. For example, if a problem is known to be in the class \( \mathcal{NP} \), but is not known to be in \( \mathcal{P} \), and is also not known to be \( \mathcal{NP} \)-complete, you answer would be “\( \mathcal{NP} \).” If there is no class with a standard definition which contains the problem, you can say, “Not a member of any class that I can find.” That could be the correct answer!

1. Given a graph \( G \), is \( G \) planar? (That is, can it be drawn in a plane with no crossings?)

\( \mathcal{NC} \). Planarity has been known to be \( \mathcal{P} \) since 1963, was shown to be linear \( O(n) \) time in 1974, and was shown to be \( \mathcal{NC} \) in 1985. It actually can be proved to be in classes even more restrictive than \( \mathcal{NC} \), but we never discussed those in class, so \( \mathcal{NC} \) is the answer I want to see.

2. Given a room and various pieces of furniture and equipment, it is possible for those items to fit into the room?

\( \mathcal{NP} \)-complete. Partition reduces to this problem. If there are \( n \) item where the \( i^{th} \) item has weight \( x_i \). By multiplying all weights by a sufficiently large factor, we may assume that \( x_i > 2 \) for all \( i \). let \( S = \frac{1}{2} \sum_{i=1}^{n} x_i \). Let \( F_i \) be a piece of furniture with a \( 1 \times x_i \) rectangular base. All furniture can be fit into a rectangular room of size \( 2 \times S \) if and only if the items can be partitioned into two sets of equal weight. The rule that \( x_i > 2 \) ensures that every piece of furniture must be inserted lengthwise, to eliminate the possibility of an “extraneous” solution that might be obtained by placing one of them crosswise.

3. Given a room with a door, and various pieces of furniture and equipment, is it possible to move those items into the room through the door? (This is not the same question!)

I believe it is \( \mathcal{P}\text{-SPACE} \) complete, same as Rush Hour. I haven’t found a proof yet, but I have confidence.

4. Does a context-free grammar generate all string? More specifically, given a context-free grammar \( G \) where \( \Sigma \) is the set of terminals of \( G \), is it true that \( L(G) = \Sigma^* \)?

Undecidable, more specifically, co-\( \mathcal{R\Sigma} \), but not recursive.

5. Given an \( n \times n \) checkerboard, for some \( n \), and given a configuration of checkers on that board, can the black player win?

\( \mathcal{EXP} \)-TIME complete.
6. Given a Turing machine $M$ and a number $t$, will $M$ halt within $t$ steps?

$\mathcal{P}$, that is, $\mathcal{P}$-time.

7. Does a given general grammar $G$ generate a given string $w$?

Undecidable, more specifically, $\mathcal{RE}$, recursively enumerable, but not recursive.

8. Given a set of jobs and a set of workers, where each worker is trained to work some given subset of the jobs, each job takes a given amount of time, and pairs of jobs $(X, Y)$ are given, where $X$ must be finished before work on $Y$ begins, can all the jobs be finished within $T$ hours?

$\mathcal{NP}$-complete. Partition can be reduced to this problem as follows. Given a set of items of weights $x_1, \ldots, x_n$, create Jobs $J_1, \ldots, J_n$ where $J_i$ takes $x_i$ hours, and where there are no dependencies, and where there are two workers, each trained to do any job. Let $T = \frac{1}{2} \sum_{i=1}^{n} x_i$. Then all jobs can be finished within $T$ hours if and only if the original items can be partitioned into two equal weight sets.

9. We define a partial inversion of a string to be the string obtained reversing any substring. For example, $abaacdab$ is a partial inversion of $abadcaab$. Given strings $u$ and $v$ and a number $k$, is it possible to obtain $v$ from $u$ by a sequence of $k$ partial inversions?

$\mathcal{NP}$-complete. This is similar to the famous “pancake flipping” problem, introduced in 1975 in the American Mathematical Monthly, and made famous by a paper, by William H. Gates and Christos H. Papadimitriou, published in 1979. (Yes, that Bill Gates.) That problem was, how can a list be sorted most efficiently using only prefix reversal, i.e. substring inversion where the substring must be a prefix. The problem of whether the sorting can be done is at most $k$ steps was proven to be $\mathcal{NP}$-complete in 2011 by Laurent Bulteau, Guillaume Ferlin, and Irena Rusu. I have not tried to generalize their result to the partial inversion problem, but I have no doubt it is also $\mathcal{NP}$-complete.