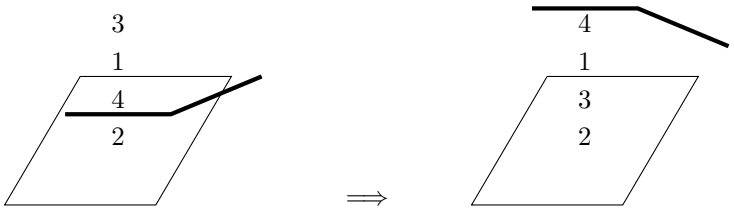
1 Construct a polynomial time algorithm that for any given (undirected) graph G finds out whether the nodes of G can be colored in red or blue so that every two adjacent nodes have different colors.

2 Construct a polynomial time algorithm that for any given integers a, b, c finds $(a^b \bmod c)$. The numbers a, b, c are written in binary notation.

3 Let RUBIK be the set of Rubik's cubes that can be solved. This set contains descriptions of the state of a 3 dimensional Rubik's cube. Show that RUBIK is in EXPSPACE. We give a formal mathematical description during the seminar. **Extra:** show that RUBIK is in PSPACE. (RUBIK is known to be in NP, in fact, every solvable configuration can be solved in at most $O(n^2/\log n)$ moves. It is an open question whether RUBIK is NP-complete.)

4 Suppose some pancakes are stacked on a surface such that no two pancakes have the same size. For convenience denote the smallest pancake by 1, the second smallest by 2, etc. The only permissible operation on the stack of pancakes is as follows: Insert a spatula between 2 pancakes or between the bottom pancake and the surface, then rotate the pancakes above the spatula. Give an algorithm that sorts any stack of n pancakes (smallest at the top largest at the bottom) with at most poly(n) many flips.



A solution for this example: $314|2 \to 4132| \to 23|14 \to 321|4 \to 1234$.

Remark: B. Gates and C. Papadimitriou gave a better algorithm that uses at most $5/3n$ flips in the worst case. Currently, the best known algorithm uses $\frac{18}{11}n$ flips. It is also known that no algorithm exists that always uses less than $\frac{15}{14}n$ flips. No better bounds are known.

5 Let A be a language over an alphabet Σ. The Kleene star A^* is a language consisting of words in the form

$$a_1 a_2 \dots a_k, \quad \text{where } a_i \in A,\ k \geq 0.$$

Prove that if $A \in \mathsf{P}$ then $A^* \in \mathsf{P}$.

6 In this exercise we prove the time hierarchy theorem, which is very similar to the space hierarchy theorem. A function f with $f(n) \geq n \log n$ is said to be *time constructable* if there exists a Turing machine that on input n computes $f(n)$ in time at most $O(f(n))$. Show that for each time constructable function f there exists a language that can not be decided in time $o(f(n))$, but can be decided in time $O(f(n) \log f(n))$.

7 Suppose some language is decided by a Turing machine that uses at most space $f(n)$, show that there is another Turing machine that decides the language and runs in space $n + \varepsilon f(n)$. Also prove precisely the same statement for time in stead of space. We use multitape Turing machines with arbitrarily finite alphabets. Why do we need to add n to $\varepsilon f(n)$?

8 This problem investigates *resolution*, a method for proving the unsatisfiability of CNF-formulas. Let $\phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$ be a formula in CNF, where C_i are its clauses. Let $\mathcal{C} = \{C_1, \dots, C_m\}$. In a *resolution step* we take two clauses C_a and C_b in $\mathcal{C}$ such that some variable occurs positively in C_a and negatively in C_b. Thus $C_a = (x \vee y_1 \vee y_2 \vee \dots \vee y_k)$ and $C_b = (\overline{x} \vee z_1 \vee z_2 \vee \dots \vee z_\ell)$. We form the new clause $(y_1 \vee \dots \vee y_k \vee z_1 \vee \dots \vee z_\ell)$ and remove repeated literals. Add this new clause to $\mathcal{C}$. Repeat the resolution steps until no additional clauses can be obtained. If the empty clause $()$ is in $\mathcal{C}$, then declare ϕ unsatisfiable.

- Show that resolution is *sound*, i.e., it never declares satisfiable formulas to be unsatisfiable.
- Show that resolution is *complete*, i.e., every unsatisfiable formula is declared to be unsatisfiable
- A 2CNF-formula is a conjunction of clauses of 2 variables. Let 2SAT be the set of satisfiable 2CNF-formulas. Show that 2SAT $\in$ P.

9 Extra. Let f be a function for which $f(n) \leq o(n \log n)$. Show that all languages that can be decided in time $f(n)$ are regular.

Problems for homework **Due: September, 29, 2017**

1 Prove that the function $n, m \mapsto \lfloor n^{1/m} \rfloor$ can by computed in polynomial number of steps (m, n are natural numbers given in binary notation).

2 The input of a problem UK is a sequence of words a_0, a_1, …, a_n in the 1-letter alphabet $\{1\}$. The problem is to decide whether the word a_0 is a concatenation of some words $a_1, \ldots, a_n$, i.e.

$$a_0 = a_{i_1} a_{i_2} \ldots a_{i_s}, \quad \text{where } 1 \leq i_\alpha \leq n, \text{ and } i_\alpha \neq i_\beta \text{ if } \alpha \neq \beta.$$

Prove that the problem UK is in the class P.