Theoretical Computer Science 2

lecture notes, university of technology Graz

Lukas Prokop

January 22, 2014

Contents

Date. 30th of Oct 2013

1 Exercise 3

Given: 2SAT instance with n variables
B is a randomized assignment. B^* is a satisfying assignment.

$$\mathbb{P}(B = B^*) = \frac{1}{2^n}$$

$$\mathbb{P}(\text{good flip}) \geq \frac{1}{2}$$

If $B \neq B^* \Rightarrow \mathbb{P}(\text{good flip}) = 1$. $t(j)$ is the number of expected steps with j incorrect values until assignment B is satisfied (therefore j values until satisfying assignment B^*).

$$t(n) \leq t(n-1) + 1$$

$$t(0) = 0$$

$$t(j) \leq \frac{1}{2}t(j-1) + \frac{1}{2}t(j+1) + 1$$

We can assume that the first and third lines have the same value.

$$\sum_{j=0}^{n} t(j) = t(0) + t(n) + \sum_{j=1}^{n-1} \frac{1}{2}(t(j-1) + t(j+1)) + (n-1)$$

We are looking for the upper boundary.

$$t(1) = 2n - 1$$

$$t(n) = 2n - 1$$

$$t(i) = 2in + i^2$$

$$t(n) = 2n^2 - n^2 = n^2$$

2 Exercise 4

$$x \in L : \text{proportion of yes answers} \geq \frac{1}{2} + n^{-c} \quad \text{evaluations with YES}$$

$$x \notin L : \text{proportion of evaluations NO answers} \geq \frac{1}{2} + n^{-c}$$

An NTM N^* which decides this language L is given. We construct a turingmachine N which decides L with portion $\frac{3}{4}$.
Idea: Run N^* $2k+1$ times sequentially. So for each result we start the computation again ($2k+1$ layers). We count the number of yes/no answers per computation path. The overall answer is given by a majority vote of the $2k+1$ runs.
Question: Which k shall we consider to achieve portion $\frac{3}{4}$?

$$P(\text{computational path of } N \text{ leads to false answer}) =$$

$$P(\sum_{j=1}^{2k+1})X_j > k \quad \text{with } X_j = \begin{cases} 1 & \text{if } N^* \text{ leads to wrong answer} \\ 0 & \text{else} \end{cases}$$

$$P(X_j = 1) = \frac{1}{2} - n^{-c}$$

$$P(X_j = 0) = \frac{1}{2} + n^{-c}$$

Chernoff estimation: $X_j \overset{\text{iid}}{\sim} \text{Bernoulli}(p)$

$$X = \sum_{j=1}^{m} X_j, \theta \in [0,1]$$

$$P(X \geq (1+\theta) \cdot p \cdot m) \leq e^{-\frac{\theta^2}{3} \cdot p \cdot \hat{m}}$$

Here:

$$n = 2k+1$$

$$p = \frac{1}{2} - n^{-c}$$

We want:

$$(1+\theta)(\frac{1}{2} - n^{-c})(2k+1) \overset{!}{=} k+1$$

$$\theta = \frac{k+1}{(\frac{1}{2} - n^{-c})(2k+1)} - 1$$

$$P(\ldots) \leq e^{-2 \cdot (n^{-c})^2 \cdot (2k+1)}$$

Choose:

$$k = \left\lceil \frac{\ln 2}{((n^{-c})^2)} \right\rceil$$

$$= e^{-2 \cdot (n^{-2c})(2 \ln 2 \cdot n^{2c} + 1)}$$

$$\leq \exp -4 \cdot \ln 2 = 2^{-2} \leq \frac{1}{4}$$

2.1 Conclusions of exercise 4

Regarding the approach: Especially Chernoff-estimation is important.

- The specific constant $\frac{3}{4}$ is irrelevant in definition of BPP. A constraint of $\frac{1}{2}$ is enough.
- BPP is (from the point of view in practice) not superseded by deterministic polynomial algorithms.

To prove point b, we just have to increase k. With $8 \cdot |x|^{2c+d} + 1$ repetitions, we have achieve the same result with the Chernoff estimation (variant with $\frac{1}{4}$).

3 Exercise 7

Given: (G, k)
Find: Is k the size of the greatest independent set?

$$L = \{(G,k) | \exists V' \subseteq V \forall V'' \subseteq V : (|V'| = k \wedge V' \text{ is independent set}) \wedge$$
$$(|V''| > k \Rightarrow V'' \text{ is not an indep. set})\}$$

It is also possible to swap V' and V'' (leaving the quantifiers). This set is polynomially balanced (because indepedent set requires $\forall u, v \in V' : \{u, v\} \notin E$ and for the independent set: $\exists u, v \in V'' : \{u, v\} \in E$). Polynomial verification is given.

3.1 $L \in DP$

Given: (G, k)
Find: $\exists$ an independent set of cardinality $k \in$ NP?

$$L_1 := \{(G,k) | \exists \text{independent set with card. k in G}\} \in \text{NP}$$
$$L_2 := \{(G,k) | \nexists \text{independent set with card. k+1 in G}\} \in \text{co-NP}$$

Therefore $L_1 \cap L_2$ because when there is no independent set with $k+1$ vertices.

3.2 L is DP-complete

Given a 3SAT-problem ϕ. For each of m clauses we add a graph with 3 vertices to G (let's call them "triangles"). We connect two nodes between the triangles if both endpoints (clauses) contain the same variable but negated.
Claim. There exists an independent set with k nodes in G. This corresponds to "ϕ is satisfiable". I is an independent set. Then I contains a vertex from every triangle.
In every clause at least 1 literal is true. Therefore the corresponding nodes are independent.
Given ϕ, ϕ' as 3SAT problems. Construct a $G = R(\phi)$ and $G' = R(\phi')$.

$$G_1 = G \cup G'$$

m_1 number of clauses in ϕ
m_2 number of clauses in ϕ'

ϕ satisfiable $\Rightarrow 2m_1$ is cardinality of maximum independent set in G_1

ϕ unsat $\Rightarrow$ cardinality of maximum independent set in $G_1 \leq 2m_1 - 2$

$G_2 \cap G'$ has $m_2 - 1$ more new vertices.

ϕ' satisfiable $\Rightarrow m_2$ maximum vertices independent set in G_1

$G_3 = G_1 \uplus G_2$ and $k = 2m_1 + m_2 + 1$.
Claim. ϕ is satisfiable, ϕ' is not satisfiable $\Leftrightarrow (G_3, k_3) \in L$.
We have shown $\Rightarrow$. We have to show $\Leftarrow$.
ϕ and ϕ' are satisfiable. ϕ is not satisfiable. ϕ' is not satisfiable.
Date. 6th of Nov 2013

4 Exercise 6

Remark. cubic = 3-regular.

$$e \in E(G)$$

We show that there is an even number of Hamiltonian cycles intersecting e.
Proof. $e = (u, v) \in E(G)$. We construct a helper graph H as follows: $V(H)$ is the set of hamiltonian paths which has V as destination vertex and contains e. Is P a path with $v, w \in V(G)$. For each edge $\{x, w\} \in E(G)$ with $x \neq v$, add an edge to H reaching from P to one of the other paths P', which is contained in $P + (x, w)$. Now a vertex in H has an odd degree iff this Hamiltonian path can be extended to a Hamiltonian cycle.
Furthermore for every Hamiltonian cycle which contains e the Hamiltonian path which TODO: Weiters für jeden Ham. Kreis der e enthält, ist der Ham Pfad der durch entfernen der anderen Kante inzident mit V im Ham. Kreis entsteht, ein Knoten in H mit ungeradem Grad.
a new node in H with even degree.
Thus the number of Hamiltonian cycles which contain e is equal to the number of vertices with odd degree in H and this one has to be even (Handshake-Lemma).

5 Exercise 8

Show. CRITICAL SAT $\in$ DP?
There exists a language $L_1 \in$ co-NP, $L_2 \in$ NP and CRITICAL SAT $= L_1 \cap L_2$. Select $L_1 :=$ UNSAT $\in$ co-NP and $L_2 =$ SAT $\circ$ MODIFICATION. SAT is in NP.
Consider m the input length of ϕ. Mod:

1. Omit k clauses [$\leq m$].
2. Overwrite clause $k + 1$ with deletion symbol [$\leq m$]. and go one step to the right.
3. Copy ($\leq m$ times)
   - Remember and delete symbol.
   - Go one to the left til 1. We have reached the deletion symbol [$\leq m$].
   - Write value [1].
   - Go right until you reach a non-deletion symbol [$\leq m$].
4. Delete deletion symbol [$\leq m$].
5. Go to the beginning [$\leq m$].

The number of steps is in $\mathcal{O}(m^2)$.

5.1 Alternative approach

Is based on SAT-UNSAT. Given is an instance of CRITICAL SAT with SAT equation ψ. ψ has m clauses and uses n variables. Now let's consider ψ_i as SAT equation which can be derived from ψ by removing the i-th clause ($i = 1..m$). Now let's consider ϕ_i as a duplicate of ψ_i and replace the variables $x_1, x_2, \ldots, x_n$ with the variables $y_{i1}, y_{i2}, \ldots, y_{in}$. Consider the SAT equation ϕ which is a result of the conjunction $\phi_1, \phi_2, \ldots, \phi_m$.

$$\phi = \phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_m$$

Claim. (ϕ, ψ) is an equivalent instance of SAT-UNSAT.
Proof. ψ is an accepted instance of CRITICAL SAT $\Leftrightarrow$ ψ is not satisfiable and ψ_i is satisfiable $\forall i \in \{1, \ldots, m\} \Leftrightarrow \phi$ is not satisfiable and ϕ_i is satisfiable $\forall i \in \{1, \ldots, m\} \Leftrightarrow (\phi, \psi)$ is an accept instance for SAT-UNSAT.

6 Exercise 10.

Given a $m \times n$ 01-matrix A.

$$C \subseteq \mathrm{col}(A) \qquad |C| = d$$

A hits C if the constraint of A to C contains all strings of $\{0,1\}^d$. VC-Dimension of A: greatest d such that A of set C for columns with cardinality d are hit.
Show. Decision problem for A, d given $\in \Sigma_2^P \cap \Pi_2^P$.

$$L = \left\{ (A,d) | \exists C \subseteq col(A) \forall C' \subseteq col(A) : \underbrace{|C| = d}_{\text{not C'}} \wedge \text{A hits C} \right\}$$

$$R((A,d), C, C') and |C| \leq |\,\mathrm{col}(A)| bzw. |C'| \subseteq |\,\mathrm{col}(A)|$$

Show. $R((A,d), C, C')$ is polynomially decidable.
$2^d > m \Rightarrow$ nicht erschlagbar,
$2^d \leq m \Rightarrow$ polynomial time to test all strings.

7 Exercise 5

Given a turingmachine M with expected runtime $T(n)$.
Definition. TM M' which simulates M, but executes at most $100 \cdot T(n)$ steps.
RP: answer No.
BPP: answer No/Yes uniformly random.
Is X a probabilistic variable for number of steps required by M.

$$P(X \geq 100 \cdot \underbrace{E(X)}_{T(n)}) \leq \frac{1}{100}$$

8 Exercise 9.

Given. TSP instance
Question. $\exists$ master tour?
Claim. Master tour $\in \Sigma_2^P$.
Proof. $V = \{v_1, \ldots, v_n\}$. A master tour exists iff there exists a tour $\sigma : v_1 \to v_2 \to \ldots \to v_n \to v_1$ such that

$$\forall S \subseteq \{1, \ldots, n\} \, \forall \tau = \underbrace{\langle u_1^s, \ldots, u_{|S|}^s \rangle}_{\text{subtour created by omitted vertices}}$$

$$\underbrace{\text{cost}(\tau)}_{\text{cost of t}} \geq \underbrace{\text{cost}(\sigma_S)}_{\text{tour resultings from } \sigma \text{ by omitting vertices}}$$

(The two subsequent forall-operators can be combined.) This proved that MASTER TOUR $\in \Sigma_2^P$, but this is not the smallest class. There exists a master tour if distance matrix can be permuted to a Kalmanson matrix and can be decided in polynomial time.
Note. PQ tree data structure.

9 Exercise 11.

Given. (G, k).
Find. $\text{VC}_{\text{path}}(G) \geq K$.
Show. $(Q) \in \Sigma_3^P$.
Remark. Schaefer has shown that $(Q) \in \Sigma_3^P$-complete (via QSAT_3). This is one of the not-so-many popular Σ_3^P-complete problems.

$$\text{VC}_{\text{path}}(G) \geq k$$

$$\exists W \subseteq V \text{ with } |W| = k \text{ such that } \forall S \subseteq W$$

$\exists$ a subgraph $\tilde{G}$ of G such that $\tilde{G}$ is graph of C (specifically if its a path) and all vertices of $\tilde{G}$ are in S, but not in KS.

10 Exercise 12.

$\exists a_1 \in E \forall b_1 \in E \backslash \{a_1\} \, \exists a_2 \in E \backslash \{a_1, b_1\} \, \forall b_2 \in E \backslash \{a_1, a_2, b_1\} \, \exists a_3 \in E \backslash \{a_1, a_2, b_1, b_2\} \, \forall b_3 \in E \backslash \{a_1, a_2, a_3, b_1, b_2$

$\exists a_4 \in E \backslash \{a_1, a_2, a_3, b_1, b_2, b_3\} \, \forall b_4 \in E \backslash \{a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4\} \, \exists a_5 \in E \backslash \{a_1, \ldots, a_4, b_1, \ldots, b_4\} : \{a_1, \ldots, a$

$$\Rightarrow TA \in \Sigma_8 P$$

$$\Rightarrow TA \in \Sigma_q P$$

ok = true. Algorithm:
for $e_b \in E \setminus \{a_1, \ldots, a_4, b_1, \ldots, b_3\}$
if Alice can't find $e_a \exists E \setminus \{a_1, \ldots, a_4, b_1, \ldots, b_3, e_b\}$ for s-t-path
ok = false return ok

$$TA \exists \Sigma_0 P = P$$

Per turn $\frac{m}{10}$ edge weights can be selected. $m = |E|$ and $10|m$.

1. After 7 turns, $m - 7 \cdot \frac{m}{10} = \frac{3m}{10}$ weights are left.
2. B selects in the next step again linear-sized subset.
3. forall quantifier cannot be "removed easily".
4. We search for the simplest shortest s-t-path. Edge weights

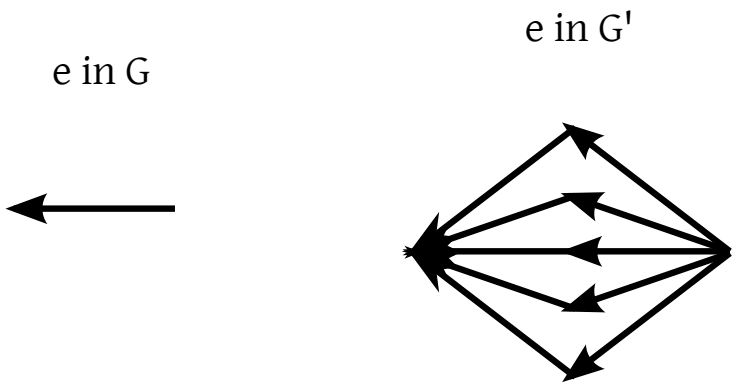
$$w(e) = \begin{cases} 0 & e \in E(A) \\ 1 & \text{else} \end{cases}$$

   We removed Bob's edge ...

11 Exercise 13

$$G = (V, A) \text{ is a directed graph}$$

Show: #CYCLE $\in$ FP $\Rightarrow$ NP = P.
Define G' such that we replace all edges in G with a gadget.



If G has a hamiltonian cycle, then G' has at least $(n^n)^n = (n^n)^2$ cycles. If G has no hamiltonian cycle, than the longest cycle has maximum length $n - 1$, maximal n^{n-1} cycles.
In G' there are at maximum $n^{n-1}(n^n)^{n-1}$ cycles.

$$n^{n-1}(n^n)^{n-1} = n^{n-1^2}$$

12 Exercise 16

Show. Permanent for integer matrix $\in \mathrm{FP}^{\#\mathrm{P}}$.
Each edge with weight 2^n is replaced with a path of length n and each weight is 2. Each edge with weight $2^n + 2^m$ is replaced with 2 parallel paths.
Representation of Z through binary representation: A' with $\mathrm{perm}(A) = \mathrm{perm}(A')$.

$$w(C) \in \{0, \pm 1\} \Rightarrow \text{A' with } \mathrm{perm}(A) = \mathrm{perm}(A')$$

$$\Rightarrow \mathrm{perm}(A') = |\left\{\pi \in S_n \mid \prod_{i=1}^{n}\right\} a'_{i\pi(i)} = 1| - |\left\{\pi \in S_n \mid \prod_{i=1} a'_{i\pi(i)} = -1\right\}|$$

13 Exercise 18

$\mathrm{P} \subseteq \mathrm{PP}$ because $\mathrm{NP} \subseteq \mathrm{PP}$.

13.1 $\Leftarrow$

$\mathrm{PP} \subseteq \mathrm{P}$: Is $L \in \mathrm{PP} \Rightarrow$ we have a turingmachine m which computes $x \in L \Leftrightarrow \#\mathrm{acc}_M(x) > 2^{p(n)-1}$ in $p(n)$ steps.
Definition:

$$f(x) = \#\mathrm{acc}_{\mathrm{M}}(x)$$

$$\text{by definition:} f \in \#\mathrm{P}$$

So there is a polynomial turingmachine M' which decides $f(x)$. Consider in M' additionally MSB of $f(x)$. This is a decision problem.

13.2 $\Rightarrow$

Remark. $\mathrm{PP} = \mathrm{P}$
Show. $\#\mathrm{P} = \mathrm{FP}$
$\mathrm{FP} \subseteq \#\mathrm{P}$. Is $f \in \mathrm{FP}$. We need a polynomial non-deterministic turingmachine M_k with

$$\#\mathrm{acc}_{\mathrm{Mx}}(x) = f(x) \;\widehat{=}\; k \in \mathbb{N}$$

Firstly compute $f(x) \;\widehat{=}\; k \in \mathbb{N}_0$. Guess $\lceil \log_2 k \rceil$ bits. Acceptance corresponds to $y \leq k$ (y is the number of guessed bits). We get exactly k accepting evaluating paths

$$f(x) \;\widehat{=}\; \#\mathrm{acc}_{\mathrm{M^k}}(x)$$

Show $\#\mathrm{P} \leq \mathrm{FP}$. Is $f \in \#P$ with turing machine M such that $\#\mathrm{acc}_{\mathrm{M}}(x) = f(x)$.

$$k_m = \min\left\{0 \leq k \leq 2^{p(n)} | k + \#\mathrm{acc}_{\mathrm{M}}(x) > 2^{p(n)}\right\}$$

$$\Rightarrow 2^{p(n)} = k_m + \#\mathrm{acc}_{\mathrm{M}}(x) - 1$$

$$\Leftrightarrow \#\mathrm{acc}_{\mathrm{M}}(x) = 2^{p(n)} - k_m + 1$$

We define a decision problem:
Given. x, k
Find. $k + \#\mathrm{acc}_{\mathrm{M}}(x) > 2^{p(n)}$.
Construct TM N with $\#\mathrm{acc}_{\mathrm{N}}(x) = k + \#\mathrm{acc}_{\mathrm{M}}(x)$ and $\#\mathrm{paths}(N) = 2^{p(n)+1}$.

For this we need a turingmachine M_k with $p(n)$ steps and exactly k accepting paths. Combine those with M.
N guesses bit $b \in \{0,1\}$. If $b = 1$, simulate $M_k(x)$. If $b = 0$, simulate $M(x)$.

$$\#\mathrm{acc}_{\mathrm{N}}(x) = \#\mathrm{acc}_{\mathrm{M_k}}(x) + \#\mathrm{acc}_{\mathrm{M}}(x) = k + \#\mathrm{acc}_{\mathrm{M}}(x)$$

14 Exercise 22

Bernd is verifier and is color blind. Sister is prover and not color blind.

$$L = \{(\text{sock } 1, \text{sock } 2) | c(\text{sock } 1) \neq c(\text{sock } 2)\}$$
$$x = (\text{sock l}, \text{sock r})$$

Bernd throws secretly a coin and gets a random value $b \in \{0,1\}$ and selects permutation $i \in S_L, \pi_0 = \mathrm{id}, \pi_1(1) = 2, \pi_1(2) = 1$.
Bernd asks sister, whether sockets got exchanged. sister answers 1 for yes and 0 for no, if we can determine it unambiguously. Otherwise sister guesses and sends answer b'. Bernd accepts if $b = b'$.

$$x \in L : \text{Bernd accepts with probability } 1$$
$$x \notin L : \text{Bernd accepts with probability } \frac{1}{2}$$

15 Exercise 29

For each number from 1 to 9 27 identical numbers are created. At each cell 3 cards are placed of given value. Petra places covered cards at each empty cell. Valentin selects from each cell randomly three cards (firstly per row, then column, then square block). Pakets of 9 cards are created. Petra covers all cards of the 27 packets and mixes them. Valentin can inspect all packets and discover that every packet contains all values from 1 to 9.
Other variations include cards, paper and scissors. There are also other protocols for this kind of exercise. In general: Given $n \times n$ grid with $n = k^2$ and $k \times k$ blocks with numbers from 1 to n (here: specialcase $k = 3$ and $n = 9$).

15.1 Another protocol

- Prover selects of random permutation $\pi : \{1, \ldots, n\} \mapsto \{1, \ldots, n\}$ and shares with the verifier for every entry (i, j) with value v the bit commitment for $\pi(v)$.
- Verifier randomly chooses of the $3n+1$ (Sudoku: $= 28$) possibilities: Either
  - One of the n rows
  - One of the n columns
  - One of the n subgrids or
  - beforehand filled cells / entries

  and requires the uncovering of the corresponding entries.

- Prover uncovers the requested values.
- Verifier accepts if there are n distinct values in (a) to (c) and correspondence is given with the filled cells in case (d). Rejection in any other case.

The error probability is $\frac{27}{28}$ or $\frac{3n}{3n+1}$. Reduced by repetition.
The disadvantage of this protocol is that the error grows with n. With another approach we have remove the dependency of n.
Basic idea: Tripling of each cell (row, column and subgrid). Will be permuted randomly. Prover has to prove the following:

- The cells contain all values from 1 to n.
- The 3 copies of each cells have the same value.
- The cells with beforehand-filled values have the correct value.

Probability to discover cheating is $\frac{1}{3}$ (independent of n!).
Too short explanation. See "Cryptographic and physical zero knowledge proof systems for solutions of Sudoku puzzles" (Gradwohl, Naro, Pinkas, Rothblum) for details.

16 Exercise 30

$$X = \{a, N\}, a \in \mathbb{Z}_N^*, 0 \leq a \leq N-1$$

B claims $\exists r \in \mathbb{Z}_N^* : a \equiv r^2 \mod N$. 2 cases:

$$x \in L : \exists t \in \mathbb{Z}_N^* : y \equiv t^2 \mod N$$

$$t = 0 \vee (rt)^2 = ay \text{ for } b = 1$$

$$x \notin L : B' \text{ is any exponential algorithm}$$

$$\text{verify } y \in \mathbb{Z}_N^* \text{ by } V$$

Then 2 cases (SN = set of square numbers):

$$y \in \mathrm{SN}_{N_2} : b = 0 \text{ is satisfiable}$$

$$b = 1 : z^2 \equiv ya \rightarrow a \equiv z^2 y^{-1} \equiv (zt)^2$$

This is a contradiction.

$$y \notin \mathrm{SN}_N \rightarrow \text{ for } b = 0 \text{ is not satisfiable}$$

For $b = 1, ya \in \mathrm{SN}_N$ is undefined. V accepts with probability $\leq \frac{1}{2}$.
Showing Zero-Knowledge property. Without loss of generality: $a \in \mathrm{SN}_N$ and V' is a random polynomial algorithm. We have a simulator S^*: Select a random $b \in \{0, 1\}$ and $t \in \mathbb{Z}_N^*$. If $b = 0 : y \equiv t^2 \mod N$, otherwise $y \equiv t^2 a^{-1}$. V' provides bit b'. If $b = b'$ respond with the response of V', else restart.

$b = 0$ uniform distribution

$b = 1$ $y = t^2(r^2)^{-1} = (tr^{-1})^2$. b' is independent of b. Therefore probability $(b = b') = \frac{1}{2}$. $ya = t^2$.

17 Exercise 23

No, this is not a correct bit fixation method.
Viktor exploits that he can select very large graphs G_0 and G_1 and the roll will expire before Barbara cannot determine isomorphism in computational infeasible time.

18 Exercise 27

$$\hat{S} = \left\{ H : H \overset{=}{\sim} G_1 \vee H \overset{=}{\sim} G_2 \right\}$$

G_1 and G_2 have n vertices. A graph with n vertices has $\leq n!$ equivalent graphs. Assume that G_1 and G_2 have exactly $n!$ equivalent graphs.
If $G_1 \overset{=}{\sim} G_2$ then $|\hat{S}| = n!$. Else ($G_1 \overset{\neq}{\sim} G_2$) $|\hat{S}| = 2n!$.
Automorphisms are introduced for the general case. Results in the S of the exercise specification. Becase aut(G) provides subgroup, we can show that the $|S|$ properties (from above) are satisfied for our S.
Assumptions to G_1 and G_2: The case $G_1 \overset{=}{\sim} G_2$ is distinguishable from case $G_1 \overset{\neq}{\sim} G_2$ by cardinality of S.
This is the foundation for the AM[2] protocol (without private random bits) for $\overline{\text{GI}}$.

19 Exercises 31/32

In the lecture we have shown that for $L \in \text{PCP(r(n), q(n))}$ there is a non-deterministic turing machine, which takes $2^{\mathcal{O}(r(n))} \cdot q(n)$ time.

$$\text{PCP(r(n), q(n))} \subseteq \text{NTIME}\left(2^{\mathcal{O}(\text{r(n)}) \cdot \text{p(n)}}\right)$$

NTIME(x) is the set of language that can be decided by a non-deterministic turingmachine with upper time boundary of x.
This non-deterministic turingmachine guesses the proof in $2^{\mathcal{O}(r(n)) \cdot q(n)}$ time and the verifier deterministically through usage of the verifier for all $2^{\mathcal{O}(r(n))}$ possibilities of random bits. In accepting cases in all cases, then accept the constructed non-deterministic turing machine.
Exercise 32 is special case. $r(n) = \log n, q(n) = 1$. Therefore $\text{PCP}(\log \text{n}, 1) \subseteq \text{NTIME}(2^{\mathcal{O}(\log \text{n})})$. $\text{NTIME}(2^{\mathcal{O}(\log \text{n})}) = \text{NP}$.

20 Exercise 34

$\overline{\text{GI}}$.
Given. 2 graphs G_0 and G_1 with n vertices each.
The proof will contain for every graph H with n vertices one bit $b_H \in \{0, 1\}$ where b_H tells us whether H is isomorphic to G_0 or G_1 (set $b_H = 0$ if H is isomorphic to G_0 or $b_H = 1$ if H is isomorphic to G_1 in case $G_0 \overset{\neq}{\sim} G_1$ and set b_H arbitrarily if H is isomorphic to G_0 and G_1).

A proof can be considered as a long bit array. Indexing by listing of all graphs with n vertices. The verifier selects $b \in \{0,1\}$ randomly and random permutation $\in S_n$. Applies π to G_b to gain the graph H which is isomorphic to G_b. Followingly ask for the proof bit b_H. Accept if $b_H = b$.
If $G_0 \overset{\neq}{\sim} G_1$ there is a proof such that verification accepts with probability 1. In case $G_0 \overset{=}{\sim}$ that accepting probability for an arbitrary proof is $\leq \frac{1}{2}$. The proof itself has exponential length.

21 Exercise 36

$$\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee \neg x_4)$$

$$p_1 = (1 - x_1) \cdot x_2(1 - x_3) = x_2 - x_2x_3 - x_1x_2 + x_1x_2x_3$$

$$p_2 = x_1 \cdot x_3 \cdot (1 - x_4) = x_1 \cdot x_3 - x_1 \cdot x_3 \cdot x_4$$

$$p_3 = (1 - x_2)(1 - x_3)x_4 = x_4 - x_3x_4 - x_2x_4 + x_2x_3x_4$$

$$q = p_1 + p_2 + p_3$$

$$I_q^1 = \{2, 4\}$$

$$I_q^2 = \{(2,3), (1,2), (1,3), (3,4), (2,4)\}$$

$$I_q^3 = \{(1,2,3), (1,3,4), (2,3,4)\}$$

$$a = (1, 0, 11) \text{ satisfying assignment}$$

$$q(a) = 0$$

$$L_1^a(c_q^1) = \sum_{i=1}^{4} a_i \cdot c_q^i = a_2 + a_4 = 1$$

$$L_2^a(c_q^2) = a_1 \cdot a_3 + a_3 \cdot a_4 + a_2 \cdot a_3 + a_1 \cdot a_2 + a_2 \cdot a_4 = 1 + 1 = 0$$

$$L_3^a(c_q^3) = a_1a_2a_3 + a_1a_3a_4 + a_2a_3a_4 = 1$$

$$q(a) \equiv 0 + 1 + 0 + 1 \equiv 0$$

22 Exercise 20

Based on arithmetization and corresponding interactive protocol.
Given. 3SAT equation ϕ with n variables and m clauses. $k \in \mathbb{N}, k \leq m$.

$$L = \{(\phi, k) : \text{number of satisfying clauses for } \phi = k\}$$

Find. Interactive protocol for L.
Consider the following arithmetization (is incorrect):

$$\begin{array}{rcl} x_i & \to & x_i \\ \neg x_i & \to & (1 - x_i) \\ \wedge & \to & + \end{array}$$

Clause. $x_i \vee \neg x_j \vee x_k$ becomes $p(x_i, x_j, x_k) = 1 - x_i \cdot (1 - x_j) \cdot x_k$.

$$p(x_i, x_j, x_k) = 1 \Leftrightarrow x_i(1 - x_j)x_k = 0$$

Goal. $p_q(x_1, \ldots, x_n) = 1$ (polynomial for q-th clause) $\Leftrightarrow$ j-th clause satisfied.
All p_q depend essentially on the 3 variables in the j-th clause, because for purposes of simplicity we write $p_q(x_1, \ldots, x_n)$ (p is polynomial of degree 3).
Total polynomial (degree $\leq 3m$):

$$P_\phi(x_1, \ldots, x_n) = \prod_{q=1}^{m} p_q(x_1, \ldots, x_m)$$

has representation of size $\mathcal{O}(m)$. Idea is to restrict values of $x_1, \ldots, x_n$ to $\{0, 1\}$ and we allow only integers.
Observation. Number of satisfying assignments for ϕ can be denoted as

$$\sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} P_\phi(b_1, \ldots, b_n)$$

The verifier has to check whether the sum above equals to k (this reduces to a checksum protocol). Now we continue with the sum and not directly with ϕ.
p is a prime number in $(2^n, 2^{2n}]$. The prover sends a prime number p to the verifier. The verifier can check whether p is a prime number. All computations are now done in $\mod p$ (this does not change anything for us, because the sum in the sum of the Observation is $\leq 2^n$).

22.0.1 Checksum protocol (general definition)

Given. polynomial $g(x_1, x_2, \ldots, x_n)$ of degree of prime number p. $\kappa \in \mathbb{N}$.
Find. Interactive protocol for testing whether

$$\kappa = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(x_1, \ldots, x_n)$$

All computations are done $\mod p$.
Necessary requirement. g must have representation with size poly(n).
Verifier must be able to evaluate $g(b_1, b_2, \ldots, b_n)$ in polynomial time (for any $b_1, \ldots, b_n$ regarding $\mathbb{F}_q$). When fixating values of $x_2, \ldots, x_n$ to values $b_2, \ldots, b_n$ a polynomial in x_1 of degree d results from g.

$$h(x_1) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(x_1, b_2, \ldots, b_n)$$

Now we can represent κ with h.

$$\kappa = h(0) + h(1)$$

Consider the following interactive protocol.
Verifier does:

- If $n = 1$, test whether $g(0) + g(1) = \kappa$.
  - If yes, accept

– else, reject

- If $n \geq 2$, request $h(x_1)$ as answer of the prover.

Prover sends polynomial $s(x_1)$ (if he does not cheat, it satisfies $s(x_1) = h(x_1)$).
Verifier:

- Rejects if $s(0) + s(1) \neq \kappa$.
- Otherwise selects $a \in \{0, \ldots, p-1\}$. Uses recursively the same protocol, to test whether

$$\underbrace{s(a)}_{\text{known constant}} = \sum_{b_2 \in \{0,1\}} \ldots \sum_{b_n \in \{0,1\}} g(a, b_2, \ldots, b_n)$$

Claim. If the κ definition/equation is wrong, then the verifier rejects with probability $\geq \left(1 - \frac{d}{p}\right)^n$. If it is correct, the prover can achieve that the verifier always accepts it.
This is a correct interactive protocol.
The correct case is immediate. For the wrong case use complete natural induction to n:

Induction base $n = 1$. Verifier rejects with probability 1, if $g(0) + g(1) \neq \kappa$.

Induction hypothesis The claim is true for polynomials g of degree d with $n-1$ variables.

Show that this is true for n variables In the first round prover sends polynomial s. If he actually sends h, then the verifier rejects immediately (because $s(0) + s(1) = h(0) + h(1) \neq \kappa$). Consider now the case that the prover provides polynomial s with $s \neq h$. Polynomial $s(x_1) - h(x_1)$ is of degree $d \Rightarrow \leq d$ roots therefore there are $\leq d$ values $a \in \{0, \ldots, p-1\}$ with $s(a) = h(a)$. Because the verifier selects a randomly:

$$\text{probability}_a[s(a) \neq h(a)] \geq 1 - \frac{d}{p}$$

If $s(a) \neq h(a)$, the prover does have to prove a wrong statement in the recursion. According to the induction hypothesis the verifier rejects with probability

$$\geq (1 - \frac{d}{p})^{n-1}$$

$$\text{probability (verifier rejects)} \geq (1 - \frac{d}{p})^{n-1}(1 - \frac{d}{p})$$

23 Exercise 26

We can assume that the quantifiers are alternating (without loss of generality). Adaption to other cases is trivial.

$$\psi : \forall x_1 \exists x_2 \forall x_3 \ldots \exists x_n : \psi(x_1, \ldots, x_n)$$

We arithmetize ϕ like previously: $p_\phi(x_1, \ldots, x_n)$.

$$\psi \in \text{TQBF} \Leftrightarrow \prod_{\mathrm{b_1}\in\{0,1\}} \sum_{\mathrm{b_2}\in\{0,1\}} \prod_{\mathrm{b_3}\in\{0,1\}} \ldots \sum_{\mathrm{b_n}\in\{0,1\}} \mathrm{p}_\phi(\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n})$$

If this equation is unlike zero, the QBF is satisfiable. Ww need a more general checksum protocol: 2 approaches.

23.1 Idea 1

Do it like before. Create a product instead of a sum at the corresponding positions. For example at the beginning test whether $s(0) \cdot s(1) = \kappa$ and analogously for all variables x_i with $\forall$ quantifiers in ψ. This creates a runtime problem (multiplication results in higher-degree polynomial) (degree up to 2^n).

23.2 Idea 2

Consider $x^k = x \forall k \in \mathbb{N}, x \in \{0,1\}$. Linearization is possible. Every polynomial $p(x_1, \ldots, x_n)$ can be transformed to a multi-linear polynomial $q(x_1, \ldots, x_n)$ (degree of q in every $x_i \leq 1$) such that p and q corresponds for all assignments of length n. Introduce linearization operator $L_{x_i}(p)$ (abbr. $L_i(p)$) such that $L_{x_i}(p)(x_1, \ldots, x_n) = x_i, p(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) + (1 - x_i), p(x_1, \ldots, x_i - 1, 0, x_{i+1}, \ldots, x_n))$ (L_{x_i} is linear in x_i) and equals to p for $x_i \in \{0,1\}$.

$$L_1(L_2(\ldots L_n(p)\ldots))$$

is a multilinear polynomial of desired type (corresponds to p for $x_1, \ldots, x_n \in \{0,1\}$). Now we can consider the all- and exists quantifier as operator.

$$\forall x_i p(x_1, \ldots, x_n) : p(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \cdot p(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

$$\exists x_i p(\ldots) : p(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) + p(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

The statement

$$\prod_{b_i\in\{0,1\}} \sum_{b_2} \ldots P_\phi(\ldots) \neq 0$$

can be reformulated if we apply alternating (starting with $\exists$) $\exists$ and $\forall$ operators to P_ϕ and a value $\kappa \neq 0$ results.
Nothing changes if linear operator is used:

$$\forall x, L_1 \exists x_2 L_1 L_2 \forall x_3 L_1 L_2 L_3 \ldots \exists x_n L_1 L_2 \ldots L_n P_\phi(x_1, \ldots, x_n)$$

This expression has length $\mathcal{O}(n^2)$.
Protocol is again inductive/recursive as previously. Assume the prover is capable convincing the verifier with probablity 1 (for a polynomial $g(x_1, \ldots, x_n)$) that $g(a_1, \ldots, a_k) = c$ if that is the case and probability $\leq \varepsilon$ if this is not that case.
Now $U(x_1, \ldots, x_l) = \mathcal{O}_g(x_1, \ldots, x_k)$ with $\mathcal{O} = \exists x_i, \forall x_i$ or L_{x_i}.
In the first two cases $l = k - 1$ and $l = k$ in the third case. Is d an upper bound for the degree of U (which is known to the verifier) in regards of x_i (here:

$d \leq 3m$). We can show that the prover can convince the verifier with probability 1 that $U(a_1, \ldots, a_l) = c$ if this is true and with probability

$$\leq \varepsilon + \frac{d}{p}$$

in case of failure.
Assume $i = 1$ (otherwise reenumerate variables). The check of the verifier is the following:

Case 1: $O = \exists x_i$ The proof provides a polynomial $s(x_i)$ of degree d such that $g(x, a_2, \ldots, a_k)$ shall be satisfied. Verifier check whether $s(0) + s(1) = c'$. If no, rejection. Otherwise acceptance and select a random $a \in \{0, \ldots, p-1\}$ and ask prover to prove $s(a) = g(a, a_2, \ldots, a_k)$.

Case 2: $\mathbf{O} = \forall x_i$ Check $s(0) + s(1) = c'$.

Case 3: $O = L_{x_i}$ Prover sends again polynomial $s(x_1)$ in case 1. Verifier check whether $a_1 \cdot s(0) + (1 - a_1) \cdot s(1) = c'$.
If no, rejection. If yes, select $a \in \{0, \ldots, p-1\}$ randomly and ask prover for proof of $s(a) = g(a, a_1, \ldots, a_k)$.

$$\exists x_i p(\ldots) = p(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) + p(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

24 Exercise 39

Accidently we show that a 2-approximation algorithm exists. Not that the greedy algorithm for MAX-SAT is a 2-approximation algorithm.
We take the SAT-RANDOM algorithm: Set every variable with uniform probability to true/false. Is W the weight of satisfiable clauses. W_c is the weight of clause c, if c is satisfiable. Otherwise 0.

$$\Rightarrow W = \sum_c W_c$$

$$E(W_c) = w_c \cdot P(c \text{ is satisfied})$$

$$|c| \ldots \text{ number of literals in } c$$

$$\alpha_k = 1 - \frac{1}{2^k} \qquad k \geq 1$$

$$\Rightarrow \alpha_k \geq \frac{1}{2}$$

If we apply SAT-RANDOM:

$$P(c \text{ satisfied}) = \alpha_k \forall \text{clauses c}$$

Proof. c is not satisfied $\Leftrightarrow$ for all literals l of c: $l = \bot$.

$$P(l = \top) = \frac{1}{2} \forall l \quad \Rightarrow \quad P(c \text{ is not satisfied}) = \frac{1}{2^{|c|}}$$

Claim. $E(W) \geq \frac{1}{2}\text{OPT}$.

$$E(W) = \sum_c E(W_c) = \sum_c \alpha_{|c|} \cdot w_c \geq \sum_c \frac{1}{2} w_c \geq \frac{1}{2}\text{OPT}$$

Derandomization. We can compute the expectancy value of W. We know, that

$$\frac{1}{2}\text{OPT} \leq E(W) = \frac{1}{2}(E(W|x_1 = \top) + E(W|x_1 = \bot))$$
$$\Rightarrow E(W) \leq E(W|x_1 = \top) \text{ for } t \in \{\top, \bot\}$$

Compute $E(W|B)$ (B is partial assignment) for clauses c with literal $= \top$. Remove clause c and add w_c.
Remove literals with value $\bot$. Determine E $(\sum_c \alpha_{|c|} \cdot w_c)$.
We get the assignment of x_1 and iteratively $x_i \forall i$.

$$\Rightarrow \frac{1}{2}\text{OPT} \leq E(W) \leq E(W|x_1 = t_1) \leq \ldots \leq E(W|x_i = t_i \forall i)$$

25 Exercise 40

Given. FPTAS for NP-complete optimization problem with OPT $= \mathcal{O}(n^c)$
Conclude. P = NP
Consider a maximization problem.

$$\frac{\text{OPT}}{\text{approximation}} \leq 1 + \varepsilon$$

Choose $\varepsilon = \frac{1}{n^{c+1}}$.

$$\frac{\text{OPT}}{\text{approximation}} \leq 1 + \frac{1}{n^{c+1}}$$

$$\text{OPT} \leq \text{approximation} + \underbrace{\frac{\text{approximation}}{n^{c+1}}}_{<1 \text{ because OPT} \leq n^c}$$

$$\Rightarrow \text{OPT} \leq \text{approximation}$$

$$\text{approximation} \leq \text{OPT}$$

$$\text{OPT} = \text{APP}$$

FPTAS provides exact solution.
Runtime: $\mathcal{O}(p(n, \frac{1}{\varepsilon})) = \mathcal{O}(p(n))$.

$$\text{P} = \text{NP}$$

26 Exercise 46

Select S randomly.

$$E = E(\text{weight of intersection}(S, V \setminus S)) = \sum_{e \in E} w_e P(e \in \text{intersection})$$

$$P(e = (v, w) \in \text{intersection}(S, V \setminus S)) = P(v \in S \wedge w \in V \setminus S) + P(w \in S \wedge v \in V \setminus S) = \frac{1}{2}$$

$$E = \sum_{e \in E} \frac{1}{2} w_e \geq \frac{1}{2} \text{OPT}$$

Derandomization:

1. Traverse all vertices in arbitrary ordering.
2. Evaluate expectancy for all current vertices v for4 $v \in S$ and $v \notin S$.
3. Wherever the expectancy is high, we assign v to this and fix v for the next vertices. If all vertices are set, finished.

Expectancy value evaluation for partially fixed vertices. Is $e = (v, w)$:

1. v, w are fixed. Then the weight of e to the expectancy if $v \in S$ and $w \in V \setminus S$ or $w \in S$ and $v \in V \setminus S$.

missing

$$\begin{aligned} \frac{1}{2}\text{OPT} &\leq E(\text{weight of intersection}(S, V \setminus S)) \\ &= \frac{1}{2}(E(x|v_1 \in S) + E(x|v_1 \notin S)) \\ E(x) &\leq E(x|v_1 \in T) \qquad T \in \{S, V \setminus S\} \end{aligned}$$

missing

27 Exercise 48: HITTING-SET-b

Given. a set U, $k \in \mathbb{N}$.
Family $\mathcal{C}$ of partial sets of U. $\forall B \in \mathcal{C} : |B| \leq b$.
Question. Is there $H \subseteq U$ with $|H| < k$ and $H \cap B \neq 0 \forall B \in \mathcal{C}$.
Show. HITTING-SET-b is NP-complete for $b \geq 2^a$ where $a \in$ NP.

27.1 a, Showing NP-hardness

VERTEX COVER is NP-complete $\Leftrightarrow$ HITTING-SET-2.
Is $G = (V, E)$ a graph. $U = V, \mathcal{C} = E$. For $b > 2 : \{i, j\} \mapsto \{i, j, j\}$.
S is independent set $\Leftrightarrow V - S$ vertex cover.

27.2 b, b-Approximation

Definition. $\mathcal{C} \setminus B$ for $B \subseteq U$

$$\{D \in \mathcal{C} : D \cap B = 0\}$$

Algorithm:

1. $H = \{\}$
2. While $\mathcal{C} \neq 0$
   (a) Is $B \in \mathcal{C}$ arbitrary.
   (b) $\mathcal{C} = \mathcal{C} \setminus B$
   (c) $H = H \cup B$

H is hitting set.
$\forall B$ at least one $x \in B$ is contained in optimal solution.

$$|B| \leq b \Rightarrow \text{guarantee} b$$

28 Exercise 50

G_i describes all clauses with i intersected variables.

$$G_0(x,y,z) = x \vee y \vee z \qquad \text{is clause}$$

$$G_1(x,y,z) = \neg x \vee y \vee z$$

$$G_2(x,y,z) = \neg x \vee \neg y \vee z$$

$$G_3(x,y,z) = \neg x \vee \neg y \vee \neg z$$

S is the set of variables with value true. V is the set of variables.

$$\max_{S \subseteq V} | \{(x,y,z) : \phi(G_0, G_1, G_2, G_3, S, x, y, z)\} |$$

$$\phi : [G_0(x,y,z) \wedge (S(x) \vee S(y) \vee S(z))] \vee$$
$$[G_1(x,y,z) \wedge (\neg S(x) \vee S(y) \vee S(z))] \vee$$
$$[G_2(x,y,z) \wedge (\neg S(x) \vee \neg S(y) \vee S(z))] \vee$$
$$[G_3(x,y,z) \wedge (\neg S(x) \vee \neg S(y) \vee \neg S(z))]$$

29 Exercise 47

Given. Items have profit $p_1, \ldots, p_n$. Knapsack capacity W. **Find.** Knapsack with maximal profit

$$\max \sum_{i=1}^{n} p_i \cdot x_i$$

under $\sum_{i=1}^{n} w_i \cdot x_i \subseteq W$ with $x_i \in \{0,1\} \qquad i = 1, \ldots, n.$

29.1 a, no constant approximation guarantee

The classical greedy algorithm sorts items such that

$$\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \ldots \geq \frac{p_n}{w_n}$$

and packs items as long as capacity is not reached.
Consider the following instance:

$$n = 2 \qquad W = k \quad k \in \mathbb{N}$$

$$p_1 = 1 \qquad p_2 = k - 1$$

$$w_1 = 1 \qquad w_2 = k$$

$$\frac{p_1}{w_1} = 1 \qquad \frac{p_2}{w_2} = \frac{k-1}{k} < 1$$

Ordering is fine. Take 1 item, profit 1. Take 2 elements, profit $k-1$. This gives guarantee $\frac{k-1}{1}$.
Arbitrary bad for $k \to \infty$.

29.2 b, approach for guarantee 2

There are several modifications of the original greedy algorithm, which result in factor 2.

- For example compare the greedy solution with the solution which only packs the most valuable item and select the better one. This gives us 2-approximation.
- Apply greedy algorithms to set of items $\{1, \ldots, n\}$ and then again to the set of remaining items. Select the better solution of them.