Computability and Complexity – Exercise 10

NP-Hard and NP-Complete, Search & Optimization, Cook & Karp Reductions

Due Thursday, January 21, 2020

Fill in the **collaboration statement**.

Submit your solution in a **single PDF format of size 5MB at most** (the grading module doesn't support other formats).

Fill in your answers **on this form**. The allocated number of lines is an indication of the expected length of the answer. You may add a few more lines if you absolutely must, but too long answers will result in point deduction.

Your PDF should include everything from the original homework file, including instructions, statements of problems, etc.

Make sure pages are properly **ordered** and **oriented**, **text is sharp and highly legible**, **contrast** between text and background is high, and the **resolution** of the submitted PDF file is such that at zoom 100% the page width fits a standard computer screen and text is large enough to easily read (the grading module lacks zooming and rotating tools).

The page size of your PDF file should be **standard A4** (the right proportions are not enough; this instruction refers to the absolute size). Orientation should be portrait.

Submit your solution electronically via the course Moodle page. After submitting, download the submitted file back from Moodle and review it. This enables you to make sure you have submitted the correct file.

A **5 pt bonus** will be given to solutions typed in a word processor (as opposed to handwritten). Hand-sketched illustrations/diagrams won't deny you this bonus.

Failure to meet any of these requirements will result in either a **25 pt fine** or **your submission not being graded**, depending on whether it's technically possible and **easy** to grade your submission or not. **No second chance will be given**.

Collaboration statement:

I collaborated with _____ (names or id's of other students) and used material from _____ (cite external sources of information). I understand that I may interact with others in order to understand how to solve the homework problems, but that the interaction should stop before I begin writing my solution. I must phrase and write the solutions on my own. I declare that I phrased and wrote my submitted solution entirely on my own, without using any written material other than the slides, textbook, and notes from the lectures and recitations. Id: 311124499

In some of the following problems, you're requested to add a *summary* of the answer. The summary is supposed to be read **instead of the full answer**. That is, a reader with the appropriate background (such as the grader) should be able to understand your solution **from the summary alone**, without reading the full answer **at all**. The summary **must not exceed the allocated number of lines** at a standard font or handwriting size (**longer summaries will be disregarded**, and you will lose the points).

[7 pts] **Problem 1**

Let $BSH = \left\{ < M, w, 1^t > \middle| \begin{matrix} M \text{ is a NTM that accepts } w \text{ using} \\ \text{at most } t \ \underline{\boldsymbol{cells}} \text{ of the tape} \end{matrix} \right\}$.

$BSH = BOUNDED - SPACE - HALTING$.

In this question we will that $BOUNDED - SPACE - HALTING$ is $NP - HARD$.

Show that for any $A \in NP$, $A \leq_P BSH$. Describe the reduction, justify correctness and running time.

$Let\ A \in NP, then\ exists\ an\ NTM\ N\ that\ decides\ A\ in\ polynomial\ time,$
$Then\ \exists d, k \in N\ s.t.\ every\ branch\ of\ NTM\ N\ runs\ for\ a\ maximum\ of$
$d * |w|^k\ steps\ for\ every\ input\ w \in \Sigma^*.$
$Now, we\ define\ from\ A\ to\ BSH\ a\ polynomial\ reduction\ f(w) =< N, w, 1^{d*|w|^k} >.$

$\underline{Running\ time}:$
$1. |< N >|\ is\ constant, |w|\ is\ linear, and\ \left|1^{d*|w|^k}\right|\ is\ polynomial\ in\ |w|$
$as\ d * |w|^k\ is\ polynomial\ in\ |w|. Therefore, \left|< N, w, 1^{d*|w|^k} >\right|\ is\ polynomial$
$in\ the\ input\ length.$
$2. f\ only\ returns < N, w, 1^{d*|w|^k} > without\ any\ aditional\ computations.$
$Therefore, f\ is\ polynomially\ computable\ in\ the\ input\ length.$

$\underline{Correctness}:$
$1. Let\ w \in A \rightarrow\ \ N\ accepts\ w\ in\ a\ maximum\ of\ d * |w|^k\ steps$
$\rightarrow\ since\ a\ TM\ cannot\ use\ more\ cells\ than\ it's\ total\ runtime,$
$N\ accepts\ w\ using\ at\ most\ d * |w|^k\ cells \rightarrow f(w) =< N, w, 1^{d*|w|^k} > \in BSH.$
$2. Let\ w \notin A \rightarrow N\ doesn't\ accept\ w\ (in\ particular\ when\ using\ only\ d * |w|^k\ cells)$
$\rightarrow < N, w, 1^{d*|w|^k} > \notin BSH$

Problem 2

1. $H3S = \left\{ < \varphi > \middle| \begin{matrix} \varphi \text{ is a } 3-CNF \text{ boolean formula containing an even} \\ \text{number of variables and there exists a satisfying} \\ \text{assignment } V \text{ that assigns exactly half of the} \\ \text{variables in } \varphi \text{ True (the rest is False)} \end{matrix} \right\}$

$H3S = HALF - 3 - SAT.$

Assume P≠NP. Determine whether $H3S$ is in P or it is $NP - Complete$.

[2 pts] Summary (2 lines):

$H3S$ is NP Complete. We'll first show H3S is in NP, and then show a reduction from 3SAT by adding another n variables and n clauses that always evaluate to true in the original 3CNF.

[14 pts] Full answer:

First, we will show a polynomial reduction F from 3SAT which is NP hard and conclude that H3S is also NP hard:

$F =$ on input φ with variables $X = \{x_1, \dots, x_n\}$ and clauses $C = \{c_1, \dots, c_m\}$ do:

1. Create φ' by adding variables $\{x'_1, \dots, x'_n\}$ and clauses $c'_1, \dots, c'_n$ where $c'_i = (x'_i \cup \overline{x'_i} \cup x'_i)$ for $1 \le i \le n$.
2. Return φ'.

(i.e. adds n additional true clauses created from n additional variables, and returns)

<u>Correctness</u>:

1. Let $\varphi \in 3SAT$, so exists an assignment V on variables $x_1, \dots, x_n$ that satisfies φ
→ Let t be the number of variables that V assigned to true.

The assignment V' defined as $V'(x_i) = V(x_i)$ for $1 \le i \le n$,
$V'(x'_k) = 0$ for $1 \le k \le t$, and $V'(x'_s) = 1$ for $t < s \le n$ (in other words,
it assigns false to t of the new variables and assigns true to the rest $n - t$ of the new varibales).

V' satisfies φ' as V satisfies φ, and we only added to φ' n clauses that always evaluate to true.

Also, the number of true assigned variables in V' is $t + (n - t) = n$, where the total amout of variables in φ' is $2n$, so the requirement for H3S holds

$\Rightarrow\ F(\varphi) = \varphi' \in H3S$
$2.\ Let\ \varphi' \in H3S, then\ exists\ an\ assignment\ V'\ that\ satisfies\ \varphi'.$
$We\ define\ an\ assignment\ V\ for\ \varphi\ as\ V(x_i) = V'(x_i)\ for\ 1 \le i \le n.$
$V\ satisfies\ \varphi\ as\ V'\ satisfies\ \varphi'\ and\ in\ particular\ clauses\ c_1, \dots, c_m\ in\ \varphi', which\ all$
$consist\ only\ out\ of\ variables\ in\ \{x_1, \dots, x_n\}\ and\ for\ those\ variables\ the\ assignments\ in$
$V\ and\ V'\ are\ identical \Rightarrow\ \varphi \in 3SAT.$

$\underline{Time\ complexity}$:
$1.\ |\varphi'|\ is\ polynomial\ in\ |\varphi|\ since\ we\ only\ doubled\ the\ amount\ of\ variables\ and\ added$
$a\ clause\ for\ each\ additional\ variable, so\ the\ number\ of\ variables\ and\ clauses\ in\ \varphi'$
$is\ polynomial\ (even\ linear) in\ |\varphi|.$
$2.\ Adding\ n\ clauses\ and\ n\ variables\ to\ \varphi\ and\ returning\ is\ done\ in\ polynomial\ time$
$in\ |\varphi|.$
$\Rightarrow F\ is\ computable\ in\ polynomial\ time\ of\ the\ input\ length.$

$Second, H3S \in NP\ since\ for\ any\ 3CNF\ \varphi\ we\ can\ use\ a\ certificate\ V\ that\ is\ the$
$assignment\ itself\ that\ satisfies\ \varphi\ with\ half\ of\ the\ variables\ assigned\ to\ true,$
$and\ verifying\ the\ certificate\ takes\ polynomial\ time.$

$\Rightarrow H3S\ is\ NP - complete.$

2. An $almost - clique$ in an undirected graph $G = (V, E)$, is defined to be a set of vertices $S \subseteq V$ such that S is a clique except for at most one missing edge.

We define the following language:

$$ALMOST - CLIQUE\ = \left\{ < G, k > \middle| \begin{matrix} G = (V, E)\ is\ an\ undirected\ graph\ with \\ an\ almost - clique\ of\ size\ k \end{matrix} \right\}.$$

Prove that if $ALMOST - CLIQUE \in P$ then $SUBSET - SUM \in P$.

[2 pts] Summary (4 lines):

$Since\ CLIQUE\ is\ NP\ hard\ and\ SUBSET - SUM \in NP\ showing$
$CLIQUE \le_p\ ALMOST - CLIQUE\ is\ enough.$
$The\ reduction\ is\ done\ by\ adding\ 2\ verticies\ to\ the\ original\ graph\ G, and\ connecting$
$them\ to\ all\ vertices\ in\ G\ except\ for\ themselfs.\ Finally\ we\ return\ < G', k + 2 >$

[14 pts] Full answer:

$Proving\ CLIQUE \leq_p ALMOST-CLIQUE\ is\ enough$:
$Since\ CLIQUE\ is\ NP\ hard\ we\ can\ conclude\ that\ ALMOST\ CLIQUE\ is\ NP\ hard\ as\ well$
$\Rightarrow Every\ language\ in\ NP\ is\ polynomially\ reducable\ to\ ALMOST-CLIQUE,$
$and\ in\ particular\ SUSET-SUM \leq_p ALMOST-CLIQUE$
$\Rightarrow if\ ALMOST-CLIQUE \in P\ then\ SUBSET-SUM \in P.$

<u>$CLIQUE \leq_p ALMOST-CLIQUE$</u>:
$Given\ a\ graph\ G=(V,E)\ with\ n\ vertices\ and\ an\ integer\ k, we\ create\ the\ graph$
$G'=(V',E')\ where$:
$V'=V \cup v_{n+1} \cup v_{n+2}$
$E'\ contains\ all\ edges\ in\ E\ and\ in\ addition\ \forall u \in V\ (v_{n+1},u) \in E', (v_{n+2},u) \in E'.$
$(Note\ that\ (v_{n+1},v_{n+2}) \notin E'.)$
$Finally, we\ define\ f(<G,k>) =< G', k+2 >.$

<u>*Correctness*</u>:
$Let < G=(V,E), k > \in CLIQUE \rightarrow \exists S \subseteq V\ s.t.\ every\ 2\ vertices\ in\ S\ share\ an\ edge$
$and\ |S|=k \rightarrow Let\ S'=S \cup v_{n+1} \cup v_{n+2} \subseteq V'. By\ definition\ of\ E', v_{n+1}, v_{n+2}\ are$
$connected\ to\ all\ vertices\ in\ S'\ besides\ themself, and\ S\ is\ a\ clique, so\ S'\ is\ a\ clique$
$besides\ the\ 1\ edge\ (v_{n+1},v_{n+2}) and\ is\ of\ size\ k+2\ in\ G'$
$\Rightarrow\ f(<G,k>) =< G', k+2 > \in ALMOST-CLIQUE.$

$Let < G', k+2 > \in ALMOST-CLIQUE \rightarrow \exists S' \subseteq V'\ s.t.\ S'\ is\ a\ k+2\ sized\ clique$
$with\ at\ most\ 1\ missing\ edge.$
$If\ \{v_{n+1}, v_{n+2}\} \subseteq S', then\ since\ we\ know\ that\ (v_{n+1}, v_{n+2}) \notin E'$
$then\ all\ other\ edges\ must\ be\ connected\ (as\ we\ are\ allowed\ only\ 1\ missing\ edge)$
$so\ S=S'\ without\ \{v_{n+1}, v_{n+2}\}\ is\ a\ clique\ of\ size\ k\ in\ G.$
$if\ w.l.o.g\ only\ v_{n+1}\ is\ in\ S', then\ we\ remove\ it\ and\ remove\ 1\ of\ the\ 2\ vertices\ that$
$miss\ an\ edge\ between\ them\ (if\ no\ edges\ miss\ we\ chose\ 2\ arbitrary\ vertices).$
$We\ get\ a\ clique\ of\ size\ k\ in\ G\ since\ we\ no\ longer\ have\ missing\ edges.$
$If\ both\ v_{n+1}, v_{n+2}\ not\ in\ S', then\ we\ remove\ the\ 2\ vertices\ that\ don't\ share\ an\ edge$
$(if\ no\ edges\ miss\ we\ chose\ 2\ arbitrary\ vertices). We\ get\ a\ clique\ of\ size\ k\ in\ G.$
$\rightarrow < G,k > \in CLIQUE.$

<u>*Time complexity*</u>:
$Complexity\ is\ polynomial\ in\ |V|\ as\ we\ only\ add\ O(|V|)\ edges\ and\ 2\ vertices\ to\ the\ input$

$graph\ and\ return\ it. Also, k+2\ is\ polynomial\ in\ input\ length.$

[6 pts] **Problem 3**

Prove or disprove.

If A is $NP-Hard$ and $A \in coNP$ then $NP = coNP$.

$True.$
$Assume\ that\ A\ is\ NP-Hard, then\ \forall L \in NP,\ \ L \leq_p A.\ Also, since\ A \in coNP\ then$
$also\ L \in coNP\ \Rightarrow\ \ NP \subseteq coNP. On\ the\ other\ direction, \forall L' \in coNP,\ \ \overline{L'} \in NP,$
$and\ since\ NP \subseteq coNP\ then\ \overline{L'} \in coNP, implying\ L' \in NP \Rightarrow coNP \subseteq NP$
$\Rightarrow NP = coNP$

Problem 4

In this problem we prove that every $NP-Complete$ language is self-reducible.

Let A be an $NP-Complete$ language and let V be a verifier of A. That is: $A = \{x | \exists c \in \Sigma^*\ s.t. V(< x, c >)\ accepts\}$.

[9 pts] **1**. Let $A' = \{< x, y > | \exists c' \in \Sigma^*\ s.t. V(< x, y \circ c' >)\ accepts\}$. Prove (full proof) that $A' \in NP$.

$We\ will\ prove\ by\ building\ a\ verifier\ V'\ for\ A'$:
$V' = on\ input \ll x, y >, c >, run\ V\ on < x, y \circ c >\ and\ do\ what\ V\ does.$

<u>*Correctness*</u>:
$Let < x, y > \in A', then\ by\ definition\ of\ A'\ \exists c' \in \Sigma^*\ s.t. V(< x, y \circ c' >)\ accepts,$
$and\ since\ V'\ does\ what\ V\ does, V'\ will\ accept \ll x, y >, c' >.$
$Let < x, y > \notin A', then\ by\ definition\ of\ A'\ \forall c' \in \Sigma^*, V(< x, y \circ c' >)\ rejects,$
$and\ since\ V'\ does\ what\ V\ does, V'\ will\ reject \ll x, y >, c' > \forall c' \in \Sigma^*$.

<u>*Time complexity*</u>:
1. $Since\ V\ is\ a\ polynomial\ verifier\ for\ A, running\ V\ on < x, y \circ c >\ takes$
$polynomial\ time\ in\ |< x >|\ by\ definition.\ Since\ | < x, y > | \geq | < x > |,$
$V\ runs\ in\ polynomial\ time\ of\ |< x, y >|\ as\ well.$
2. $Accepting\ or\ rejecting\ is\ O(1).$

$\Rightarrow V'\ runs\ in\ polynomial\ time\ of\ | < x, y > |.$

[9 pts] **2**. Let O_d an oracle for the decision problem A_d, you may assume that the oracle do that in $O(1)$ time. Prove that O_d can be used in order to decide A' in polynomial time.

$We\ proved\ in\ 4.1\ that\ A' \in NP\ and\ we\ know\ that\ A\ is\ NP-Complete, so\ by\ definition\ of\ NP\ completeness\ there\ is\ a\ polynomial\ reduction\ F_A\ from\ A'\ to\ A. Therefore, we\ can\ build\ the\ polynomial\ decider\ D_{A'}\ for\ A'\ as\ the\ following:$
$D_{A'} = on\ input < x, y >, run\ O_d\ on\ F_A(< x, y >)\ and\ do\ what\ O_d\ does.$

$\underline{Correctness}$:
$Let < x, y > \in A' \rightarrow F_A(< x, y >) \in A \rightarrow O_d\ accepts\ F_A(< x, y >)$
$\rightarrow D_{A'}\ accepts < x, y >.$
$Let < x, y > \notin A' \rightarrow F_A(< x, y >) \notin A \rightarrow O_d\ rejects\ F_A(< x, y >)$
$\rightarrow D_{A'}\ rejects < x, y >.$

$\underline{Time\ complexity}$:
$F_A\ is\ a\ polynomial\ reduction\ so\ it\ runs\ in\ polynomial\ time\ of\ |< x, y >|, and\ O_d\ runs\ in\ O(1), so\ D_{A'}\ runs\ in\ polynomial\ time\ of\ |< x, y >|.$

[no pts] **3**. Prove (full proof) that $A_s \leq A$ where A_s is the search problem of A and A is the decision problem of A. In other words, prove that A is self-reducible.

Hints:

- Try to expand the idea of proof you saw in class that $3-SAT$ is self- reducible in order to prove that this claim is true for any language that is $NP-Complete$.
- Try to define what is the search problem of A (A_s) by using the polynomial verifier V of A.
- Note that A' is the language of all the strings $< x, y >$ for which y is the prefix of the certificate that prove that $x \in A$. Try to think about how you use A' to extract the certificate c that proves that $x \in A$ character after character.
- In your proof use V, O_d and what you showed in question 2.
- In the correctness use induction on the number of times you call V.

Problem 5

Recall that a $3-coloring$ of an undirected graph $G = (V, E)$ is a function $c: V \rightarrow \{1,2,3\}$ such that for every $\{u, v\} \in E$, $c(u) \neq c(v)$. In other words, a $3-coloring$ o an

undirected graph is an assignment of 3 colors to its vertices so that no two adjacent vertices are assigned the same color.

Let $3-COLOR = \{< G = (V,E) > | G\ has\ a\ 3-coloring\}$.

Define the search problem of $3-COLOR$ and show directly, without using the theorem that every $NP-complete$ language is self-reducible or its proof, that the search problem is polynomial-time reducible to the decision problem.

[2 pts] Summary (3 lines):

$Given\ a\ graph\ G\ and\ an\ oracle\ O_{3c}\ for\ 3\ COLOR, we\ add\ 3\ vertices\ v'_1, v'_2, v'_3\ and\ connect\ them\ together. Then, we\ go\ over\ all\ vertices\ of\ G\ and\ iteratively\ connect\ every\ vertex\ to\ 2\ of\ v'_1, v'_2, v'_3\ and\ rerun\ O_{3c}. Finally, (v'_i, u) \in G'\ implies\ Color(u) = i.$

[30 pts] Full answer:

$Given\ an\ oracle\ O_{3c}\ for\ 3-COLOR\ that\ runs\ in\ O(1), we\ define\ a\ TM\ S\ that\ solves\ the\ search\ problem\ for\ 3\ color\ in\ polynomial\ time, i.e.\ given\ a\ graph\ G = (V = \{v_1, \dots v_n\}, E)\ S\ returns\ a\ mapping\ C = \{c_1, \dots, c_n\}\ so\ coloring\ every\ v_i\ with\ color\ c_i\ gives\ a\ correct\ 3\ color\ mapping\ of\ G.$

$S = on\ input < G = V, E > where\ V = \{v_1, \dots, v_n\}, do:$
1. $Run\ O_{3c}\ on < G >, if\ O_{3c}\ rejects, return\ no.$
2. $Add\ v'_1, v'_2, v'_3\ to\ V\ and\ add\ edges\ between\ all\ 3\ of\ the\ vertices\ to\ E.$
3. $For\ each\ v_i \in V\ excluding\ \{v'_1, v'_2, v'_3\}:$
   - 3.1. $Add\ (v_i, v'_2), (v_i, v'_3)\ to\ E\ and\ run\ O_{3c}\ on\ resulted\ graph.$
   - 3.2. $If\ O_{3c}\ accepts, set\ c_i = 1\ and\ continue\ to\ next\ iteration.$
   - 3.3. $Else, remove\ the\ added\ edges, add\ (v_i, v'_3), (v_i, v'_1)\ to\ E\ and\ run\ O_{3c}\ on\ resulted\ graph.$
   - 3.4. $If\ O_{3c}\ accepts, set\ c_i = 2\ and\ continue\ to\ next\ iteration.$
   - 3.5. $Else, remove\ the\ added\ edges, add\ (v_i, v'_1), (v_i, v'_2)\ to\ E\ and\ set\ c_i = 3.$
4. $Return\ \{c_1, \dots, c_n\}.$

<u>*Correctness*</u>:
$-\ Let < G > \notin 3COLOR, then\ O_{3c}\ rejects < G >\ and\ S\ returns$ no $in\ line$ 1.
$-\ Let < G > \in 3COLOR$:
$\boldsymbol{Claim\ 1}: If\ G\ has\ a\ 3\ coloring, then\ after\ adding\ vertices\ v'_1, v'_2, v'_3\ and\ their\ connecting\ edges,$

G *still has a* 3 *coloring*.
Proof: G_c, *the subgraph containing* v'_1, v'_2, v'_3 *and their connecting edges is a graph of size* 3 *and therefore has a* 3 *coloring*.
Also, v'_1, v'_2, v'_3 *don't share any edges with the rest of* G, *and the original* G *has a a* 3 *coloring, so after adding* G_c *it still has a* 3 *coloring*.

***Claim* 2**: v'_1, v'_2, v'_3 *must be colored in* 3 *different colors*.
Proof: *Since every* 2 *vertices in* $\{v'_1, v'_2, v'_3\}$ *are connected by an edge, they must also be of different colors (by definition of graph coloring).*
** *Since the colors are arbitrary, we'll assume w.l.o.g. that* v'_1 *represents color* 1, v'_2 *color* 2, *and* v'_3 *color* 3 **

***Claim* 3**: *If there exists a* 3 *coloring of* G *after connecting a vertex* v_i *to* 2 *of* $\{v'_1, v'_2, v'_3\}$, *and assume w.l.o.g the* v_i *is connected to* v'_2, v'_3, *then in this coloring* v_i *has the same color as* v'_1 *(which is color* 1, *by our assumption in the end of claim* 2)
Proof: *By definition of* 3 *coloring, since by claim* 2 v'_2, v'_3 *are colored differentlly and* v_i *is connected to both of them, then* v_i *must have a different color than* v'_2, v'_3, *which must be the color of* v'_1, *color* 1.

***Claim* 4**: *If there is a* 3 *coloring for* G, *then after every iteration* i *there will also be a* 3 *coloring (i.e. after adding the* 2 *edges) where the colors of* $\{v_1, \dots, v_i\}$ *are* $\{c_1, \dots, c_i\}$.
Proof: *By induction on* i.
Base: *For* $i = 0$, *the graph* G *has a* 3 *coloring by definition.*
Induction hypothesis: *Assume correcness for* $i = k - 1$, *meaning after* $k - 1$ *iterations,* G *still has a* 3 *coloring where* $\{v_1, \dots, v_{k-1}\}$ *are colored* $\{c_1, \dots, c_{k-1}\}$.
Induction step: *Assume the induction hypothesis holds,*
Then in iteration $i = k$, *if in* 3.1. *after adding the edges* $(v_k, v'_2), (v_k, v'_3)$ *the oracle* O_{3c} *accepts, then by claim* 3 v_k *must be colored* 1, *and indeed in* 3.2. *S sets* $c_k = 1$.
If in 3.3. *after adding the edges* $(v_k, v'_3), (v_k, v'_1)$ *the oracle* O_{3c} *accepts, then by claim* 4 v_k *must be colored* 2, *and indeed in* 3.4. *S sets* $c_k = 2$.
If S reached 3.5 *then there does not exist a coloring where* v_k *is colored* 1 *or* 2.
Since the added edges in 3.1 3.3. *are removed our graph is back to the same state it was in iteration* $k - 1$, *and by the induction hypothesis we know that we have a* 3 *coloring for* G *in this state, so* v_k *must be colored* 3 *and indeed S sets* $c_k = 3$ *in line* 3.5.
Note that by claim 3 $\{v_1, \dots, v_{k-1}\}$ *must still be colored* $\{c_1, \dots, c_{k-1}\}$ *because the* 2 *edges*

$that\ we\ added\ to\ each\ one\ of\ the\ vertices\ \{v_1, \dots, v_{k-1}\}\ still\ exist\ at\ the\ end\ of\ this\ iteration.$

$\Rightarrow We\ are\ prommised\ after\ iteration\ i = n\ that\ \{c_1, \dots, c_n\}\ define\ a\ valid$
$3\ coloring\ for\ the\ vertices\ of\ graph\ G, which\ is\ exactly\ what\ we\ searched\ for.$

$\underline{Time\ complexity}:$
$1. Adding\ v_1', v_2', v_3'\ and\ their\ connecting\ edges\ to\ G\ is\ done\ in\ polynomial\ time.$
$2. We\ perform\ |V|\ iterations\ and\ in\ each\ iteration\ run\ O_{3n}\ and\ add\ a\ constant$
$number\ of\ edges\ at\ most\ 3\ times, so\ each\ iteration\ takes\ a\ polynomial\ time\ in$
$|< G >|.$

$\Rightarrow S\ calls\ O_{3c}\ a\ polynomial\ number\ of\ times\ and\ runtime\ is\ polynomial$
$in\ | < G > |.$

Problem 6

Prove or disprove.

[7 pts] **1**. If $A \leq_{Cook} B$ and $B \in R$, then $A \in R$.

$True.$
$B \in R\ so\ it\ has\ a\ decider\ M_B. A \leq_{cook} B\ so\ A\ can\ be\ solved\ using\ a\ polynomial$
$number\ of\ calls\ to\ the\ oracle\ of\ B.$
$Therefore, by\ replacing\ the\ oracle\ of\ B\ with\ M_B\ we\ get\ a\ decider\ for\ A,$
$so\ A \in R.$

[7 pts] **2**. If $A \leq_{Cook} B$ and $B \in coRE$, then $A \in coRE$.

$False.$
$A_{TM} \leq_{cook} \overline{A_{TM}}\ by\ the\ decider\ 'M_A = on\ input < M, w >, run\ \overline{A_{TM}}'s\ oracle\ on$
$< M, w >, if\ it\ accepts, reject, and\ if\ it\ rejects, accept.'$
$Also, \overline{A_{TM}} \in coRE\ but\ A_{TM} \notin coRE.$

[7 pts] **3**. If $P = NP$ then every non-trivial language $L \in P$ is self-reducible.

$True. Let\ L' \in P\ be\ non\ trivial. We\ can\ reduce\ every\ L \in P\ to\ L'\ by\ running\ a$
$decider\ of\ L\ and\ returning\ an\ element\ in\ or\ outside\ of\ L'\ by\ the\ result\ of\ the\ decider.$
$Therefore, and\ since\ P = NP, then\ L'\ is\ NP - Complete\ by\ definition.$

$Finally, as\ we\ proved\ in\ Q4\ that\ every\ NP\ complete\ language\ is\ also\ self\ reducable,$
$L'\ is\ self\ reducable\ as\ well.$

[no pts] **Problem 7**

Consider the proof of the Cook-Levin theorem shown in the lecture.

Recall that the proof fixes a language $A \in NP$ and then, given a nondeterministic Turing machine N that decides A in n^k time (where n is the size of the input and k is some constant), constructs for each word w a Boolean formula $f_N(w) = \Phi$ that consists of four parts: $\Phi_{cell}, \Phi_{start}, \Phi_{move}\ and\ \Phi_{accept}$.

For each of the following modifications, state which of the four parts of Φ may change. Explain your answers.

1. Replace w with w' such that $|w| = |w'|$. A and N remain the same.

2. Replace A and N with A' and N' such that N' decides A' in n^k time (the same k as before). w remains the same.

3. Replace A and N with A' and N' such that N' decides A' in $n^{k'}$ time. The set of states and the alphabet of N' are the same as those of N. w remains the same.