EECS 4111/5111/6111 Computability
Jeff Edmonds
Assignment 6: NP & Reductions
Due: One week after shown in slides

First Person:	Second Person:
Family Name:	Family Name:
Given Name:	Given Name:
Student #:	Student #:
Email:	Email:

Guidelines:

- You are strongly encouraged to work in groups of two. Do not get solutions from other pairs. Though you are to teach & learn from your partner, you are responsible to do and learn the work yourself. Write it up together. Proofread it.
- Please make your answers clear and succinct. helpful hints.
- Relevant Readings:
  - NP-Slides
  - NP from "Thinking About Algorithms Abstractly"
- This page should be the cover of your assignment.

Problem Name	Max Mark	
1 Software Packages	10	
2 NP-Complete Cards	10	
3 NP-Complete Vertex Cover	10	
4 NP-Complete Bounded Witness	10	
5 Purpose of steps	10	

EECS 4111/5111/6111 Computability
Jeff Edmonds
Assignment 6: NP & Reductions
Due: One week after shown in slides

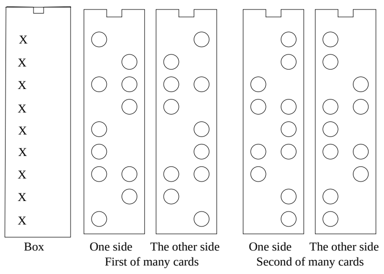
1. (a) There is a collection of software packages $S_1, \ldots, S_n$ which you are considering buying. These are partitioned into two groups. For those $i \in N \subseteq [n]$, the costs of buying it out ways the benefits and hence it effectively costs you a given amount $b_i \geq 0$ to buy it. For those $j \in P \subseteq [n]$, the benefits out way the costs of buying it and hence it effectively costs you a given amount $b_j \geq 0$ to *not* buy it. Some of these packages rely on each other; if S_i relies on S_j, then you will incur an additional cost of $a_{\langle i,j \rangle} \geq 0$ if you buy S_i but not S_j. Provide a polytime algorithm to decide the subset $S \subseteq [n]$ of $S_2, \ldots, S_n$ that you should buy. The cost of your solution is $cost(S) = \sum_{i \in S \cap N} b_i + \sum_{j \in \overline{S} \cap P} b_i + \sum_{i \in S, j \in \overline{S}} a_{\langle i,j \rangle}$. Hint: Do not design a new algorithm but do a reduction to max flow - min cut similar to that done for matching the boys and girls.

   (b) In the old version, for each i, you will gain an overall benefit of b_i if you acquire package S_i. Possibly b_i is negative. How does this effect the reduction (beyond taking the absolute value).

2. NP-Completeness

   (a) Is *Clique* $\leq_{poly}$ 3-*Clique* true? Is 3-*Clique* $\leq_{poly}$ *Clique* true? Give one sentence for each about how you know.

   (b) Consider the following puzzle.
   The input consists of a collection of cards as indicated in the figure and a box to place them in. The box and each card has r rows and two columns of positions in the same places. The box contains an Xs for each row of the left column. In each card at each of these positions (left and right), there is either a hole punched out or there is not. Each card must be placed in the box either face up or flipped over left to right. It has a notch at its top which must be at the top of the box.



   Consider some row in some card. Suppose its left position has a hole but its right position does not. Putting the card in face up does not cover the X in the box of this row, because the X is on the left and the hole does not cover it. Putting the card in flipped over, however, does cover the X. The goal is to cover each and every X in the box. A solution specifies for each card whether to flip the card or not. A solution is valid if it covers all the Xs. Recall, that *all* the cards are stacked in the box. So saying that each X is covered amounts to saying that for each X, there is at least one of the cards that is flipped in a way that will cover the X.

   Prove that *Card* is NP-Complete by reducing it to/from Clause-SAT. Be sure to think about each of the 12 steps. You are to write up steps 0, 5, 6, and 7. For each of these have a clear paragraph with pictures.

   The problem Clause-SAT is given a set of clauses where each clause is the OR of a set of literals and each literal is either a variable or its negation. The goal is know whether you can set each variable to true or false so that each clause is satisfied.

   0) $P_{card} \in$ NP:

   5) InstanceMap:

   6) SolutionMap: Map a potential solution S_{card} of the card problem to a potential solution $S_{Clause-SAT}$ of Clause-Sat.

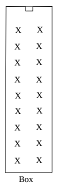
   7) Valid to Valid: Assume that S_{card} is a valid solution, i.e. every rows r has some card c' that covers the left X in that row. Our goal is to prove that $S_{Clause-SAT}$ is a valid solution, i.e. every clause c has some variable x that satisfies it.

$$\forall r\ \exists c'\ covers(r, c') \quad \Rightarrow \forall c\ \exists x\ satisfies(c, x)$$

   Such a proof is reminiscent of that in Assignment 0 Quantifiers Question 2c.

(c) Repeat the last question except for the following change.

The box now has Xs in both the left and the right columns and the cards must cover all of these Xs. Hint: Remember that you are able to create the set of cards and you can choose to create one extra card. What is that card that you create? What changes do you have to made to steps 6 and 7 above?



3. NP-Completeness: The problem Vertex Cover is given an instance $I_{VC} = \langle G, k \rangle$ where G is an undirected graph and k is integer k. A solution S_{VC} is a subset of the nodes of G. It is valid if every edge has at least one of it's nodes is in S_{VC} and $|S_{VC}| \leq k$. Prove that VC is NP-Complete by reducing it to/from 3-SAT. Be sure to think about each of the 12 steps. You are to write up steps 0, 5, 6, and 7. For each of these have a clear paragraph with pictures.

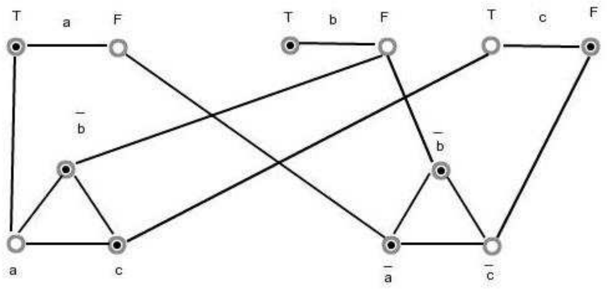
0) $P_{VC} \in$ NP:

5) InstanceMap:

6) SolutionMap: Map a potential solution S_{VC} of the card problem to a potential solution S_{3-SAT} of Clause-Sat.

7) Valid to Valid:

Hint: Given the clauses (a or $\neg b$ or c) and ($\neg a$ or $\neg b$ or $\neg c$) consider the following graph.



4. Both NP-complete and acceptable-complete problems have a witness that a fairy godmother could provide so that you could verify in poly-time that a given instance is a *yes* instance. The difference is that for the NP-complete problem, this witness can't be more than polynomially bigger than the instance itself, while the witness for the acceptable-complete (uncomputable) problem can be arbitrarily big. Suppose now that you took an acceptable-complete problem and changed it so that an instance I is a *yes* instance iff it has a polynomial sized witness. The intuition is that the resulting problem would automatically become NP-complete. This problem examines this idea.

A *non-deterministic Turing machine* is the same as a deterministic one, except that its moves are not deterministic. In a deterministic TM, the transition function $\delta(q_i, c) = \langle q_j, c', right \rangle$ states that if the TM is in state q_i and sees the character c under its head, then it changes to state q_j, writes the character c', and moves the head right. In a non-determinist TM, this transition function is changed to a transition relation δ. If both tuples $\langle q_i, c; q_j, c', right \rangle$ and $\langle q_i, c; q_k, c'', left \rangle$ are in the relation δ, then when the TM is in state q_i and sees the character c under its head, then it has a choice whether to change to state q_j, write the character c' and move the head right or to change to state q_k, write the character c'' and move the head left. A non-deterministic TM M on input I is said to *accept* its input if there exists an accepting computation.

Note that just as the problem Det-Accept $= \{\langle M, I \rangle \mid$ the deterministic TM M accepts the input $I\}$ is acceptable but uncomputable, so is the problem NDet-Accept $= \{\langle M, I \rangle \mid$ the non-deterministic TM M accepts the input $I\}$.

(a) Give a non-deterministic TM M_{SAT} that takes an *and/or/not* circuit C as input and accepts it iff it has a satisfying assignment $\langle x_1, \ldots, x_n\rangle$. If C has n input wires x_i and N gates, what is the running time of your TM?

(b) Prove that the problem NDet-Accept' $= \{\langle M, I\rangle \mid$ the non-deterministic TM M accepts the input I in at most n^3 time $\}$ is NP-complete. Reduce this problem to SAT. Do the key steps.

(c) Recall the *Post correspondence problem*, PCP, defined in the slides. The input is a finite collection of dominoes $P = \{\binom{b}{ca}, \binom{a}{ab}, \binom{ca}{a}, \binom{abc}{c}\}$. A solution is a finite sequence of the dominoes with repeats so that the combined string on the top and on the bottom are the same, $S = \binom{a}{ab}\binom{b}{ca}\binom{ca}{a}\binom{a}{ab}\binom{abc}{c}\}$. The slides prove that this problem is undecidable by reducing it to the problem Det-Accept $\{\langle M, I\rangle \mid$ the deterministic TM M accepts the input $I\}$. What would change if the reduction was to non-deterministic TM instead of deterministic TMs?

(d) Change the PCP problem as stated to PCP' in which the dominoes in the input set can be repeated as many times as the user likes but unlike before, the solutions can use each domino as most once. Prove that this new version is NP-complete. Reduce PCP' to NDet-Accept'.

5. Reductions: The purpose of this question is to learn to appreciate why there are so many steps when doing a reduction.

0) $P_{oracle} \in$ NP: Prove that problem P_{oracle} is in NP.

5) InstanceMap: Define a polynomial time mapping $I_{oracle} = InstanceMap(I_{alg})$.

6) SolutionMap: Define a polynomial time algorithm mapping each potential solution S_{oracle} for I_{oracle} to a potential solution $S_{alg} = SolutionMap(S_{oracle})$ for I_{alg}.

7) Valid to Valid: Prove that if S_{oracle} is a valid solution for I_{oracle}, then $S_{alg} = SolutionMap(S_{oracle})$ is a valid solution for I_{alg}.

8) ReverseSolutionMap: Define a polynomial time algorithm mapping each potential solution S'_{alg} for I_{alg} to a potential solution $S'_{oracle} = ReverseSolutionMap(S'_{alg})$ for I_{oracle}.

9) Reverse Valid to Valid: Prove that if S'_{alg} is a valid solution for I_{alg}, then $S'_{oracle} = ReverseSolutionMap(S'_{alg})$ is a valid solution for I_{oracle}.

Consider the problem *Block* whose instance $\langle s, k\rangle$ is a 01-string s and an integer k. Such an instance is a yes instance if it has a solution consisting of a block $s_i, s_{i+1}, \ldots s_j$ of all ones whose length is k.

(a) (2 marks) Is this Block problem in NP? If so, what would this mean?

(b) (2 marks) Is this Block problem is NP-Complete? What would it mean?

(c) (2 marks) Your home work is to prove that Block is NP-complete by proving that 3-SAT $\leq_{ploy}$ Block. You are to work in pairs. Your partner took on step 5 as follows. P_{alg} is 3-SAT. As an example, $I_{\text{3-SAT}} = (x \text{ or } \overline{y} \text{ or } z)$ AND $\ldots$ AND $(\overline{u} \text{ or } v \text{ or } w)$ is an instance of 3-SAT. Given such an instance, he maps it to an instance $I_{\text{Block}} = \langle s, k\rangle$ where $s = 000\ldots0$ is the string consisting of n zeros and k is 5. Besides a few hints which he ignored, he noted that he did in fact meet the formal requirements of step 5. Being a bit of a know it all, your partner thought steps 8 and 9 are silly and hence decided to skip them. He left you the task of doing steps 6 and 7. Carefully, review what these steps are and complete them if you can. Careful.

(d) Because your previous partner dropped out of school, you got a new one. Having learned, he took a new approach. He decided to map an instance $I_{\text{3-SAT}}$ of 3-SAT to an instance $I_{\text{Block}} = \langle s, k\rangle$ where $s = 111\ldots1$ is the string consisting of $n \geq 5$ ones and k is 5. He went on to do steps 6 and 7 as follows. Given a solution $s_i, s_{i+1}, \ldots s_j$ to I_{Block}, he instructed how to construct a valid solution to $I_{\text{3-SAT}}$ as follows. For each of the m clauses $(x \text{ or } \overline{y} \text{ or } z)$ of $I_{\text{3-SAT}}$ choose one of the variables. If, like x, it is not negated in the clause then set the variable to true. If, like y, it is negated, then set it to false. Either way the clause is satisfied. Doing this for each clause, satisfies each clause. Hence, the assignment obtained is a valid assignment of the variables. This new partner learned (from the guy who dropped out) the importance of doing steps 8 and 9 so he left them for you to do.

i. (2 marks) Carefully, review what these steps are and complete them if you can. Careful.

ii. (2 marks) Are you happy with your new partner? Did he follow steps 5, 6, and 7 correctly? Explain.