Justin Phillips
HW7
CS325 spring 19

1. (6pts)
Let X and Y be two decision problems. Suppose we know that X reduces to Y in polynomial
time. Which of the following can we infer? Explain
a. If Y is NP-complete then so is X.
NO we only know that x reduces to y therefore we can not determine that if y is npc then x is npc. x might not be np
b. If X is NP-complete then so is Y.
No this isn't true because we only can determine that x reduces to y this not determine that x is npc. y may not be np
c. If Y is NP-complete and X is in NP then X is NP-complete.
no, we only know that x reduces to y so we cant not determine the opposite
d. If X is NP-complete and Y is in NP then Y is NP-complete.
Yes, x reduces to y and since npc complete is the hardest np since y is np we can determine that y is npc
e. If X is in P, then Y is in P.
No, if x is <= y it could mean to y is harder the x and y may not be in p
f. If Y is in P, then X is in P.
Yes, since x reduces to y in p time and x is easier or = to y since p is a subset of np and y is in p. We can determine to x must be in p since it is easier or = to y.

2(4 pts)
Consider the problem COMPOSITE: given an integer y, does y have any factors other than one and itself?
For this exercise, you may assume that COMPOSITE is in NP, and you will be comparing it to the well-known NP-complete problem SUBSET-SUM: given a set S of n integers and an integer target t, is there a subset of
S whose sum is exactly t? Clearly explain whether or not each of the following statements follows from that fact that COMPOSITE is in NP and SUBSET-SUM is NP-complete:
a. SUBSET-SUM ≤p COMPOSITE.
False, This does not reduce down since subset-sun is npc and composite is np, therefore if x reduces to y and x isn't harder than y, npc can't be reduced to an np

b. If there is an $O(n^3)$ algorithm for SUBSET-SUM, then there is a polynomial time algorithm for COMPOSITE.
This is True since n^3 is polynomial run time and subset sum is a npc we can determine that p=np so all np problems will have a poly run time which would include composite.

c. If there is a polynomial algorithm for COMPOSITE, then P = NP.

False, just because composite is np doesn’t mean its npc therefore is doesn’t imply composite is a p=np

d. If P≠NP, then `no` problem in NP can be solved in polynomial time.

false, p is in np and p problems can be solved in poly time. p!=np only implies the npc problems cant be solved in poly time

3.(8pts)
A Hamiltonian path in a graph is a simple path that visits every vertex exactly once. Prove that HAM-PATH = {(G, u, v): there is a Hamiltonian path from u to v in G} is NP-complete. You may use the fact that HAM-CYCLE is NP -complete.

1)show that HAM-Path is in NP

a graph G’(u,v) has a HAM-Path starting node is u and traverse the path visiting every vertex once till reaching the end node of v. This can be done in poly time therefore HAM-Path is NP

2) Show that HAM-CYCLE <=p HAM-PATH for HAM-CYCLE E NPC

a)it is known that HAM-CYCLE is NPC

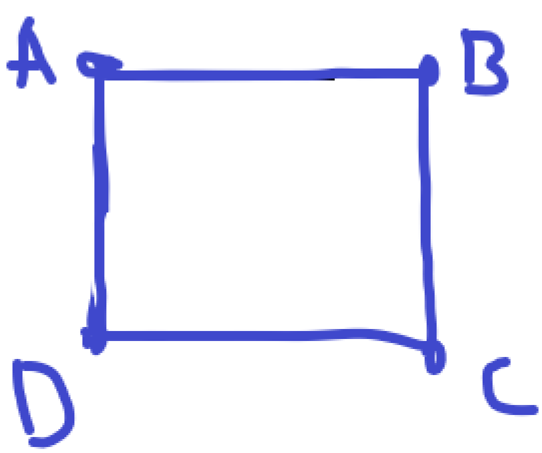
b) show a poly algorithm so that HAM-CYCLE is a instance of HAM-PATH

Given a graph G we make a new G’ such that G has a HAM-CYCLE if and only if G’ has a HAM-PATH. G’ is created by choosing a vertex u in G and creating a new vertex u’. u ‘ is connected to the neighbor of u, then we add vertex v and connect it to u, then add vertex v’ the connect that to u’.

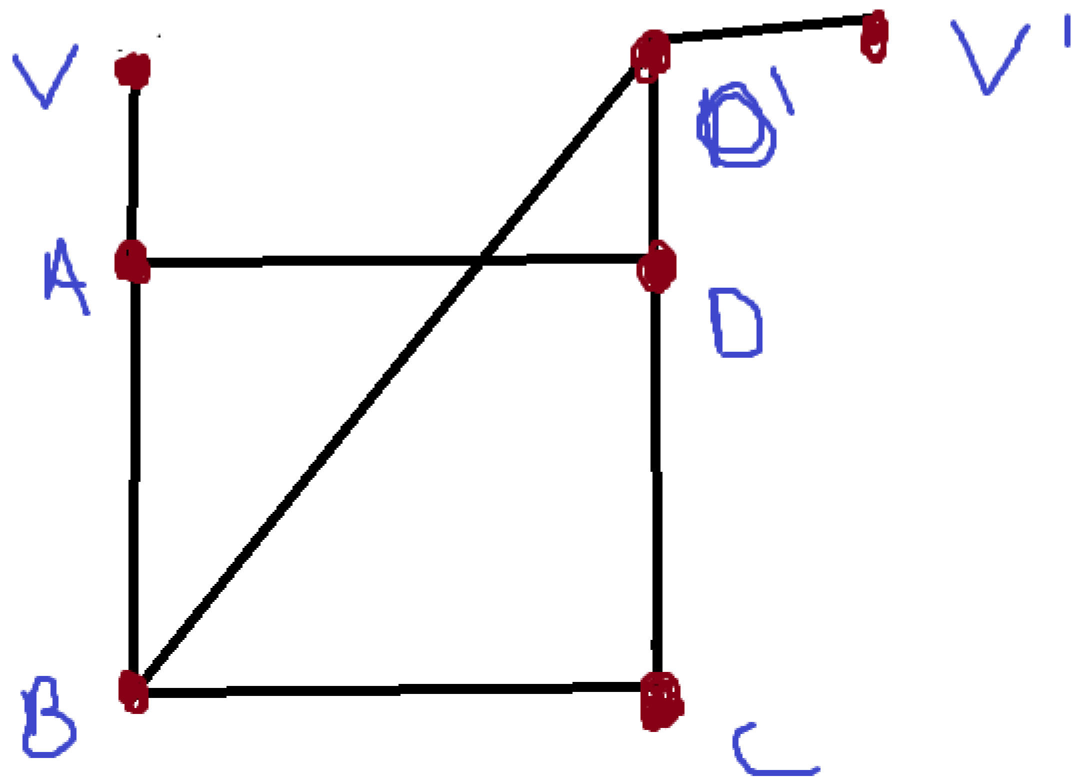
c) Now we prove that you are able to solve a HAM-CYCLE by using HAM-Path, proving that HAM-Path is as hard as HAM-CYCLE.

Graph G

G’




graph g’




(PAINT SKILLS ELITE)

as seen above G has a HAM-CYCLE if and only if G' has a HAM-PATH

- now if G has a HAM-CYCLE a, b, c, d, a then G' has a HAM-PATH v, a,b,c,d,d',v'
- if G' has a HAM-PATH v, a,b,c,d,d',v' and we then remove the v, d' and v' the graph becomes = G and the HAM-CYCLE a,b,c,d,a
- since the above are both true we can determine that HAM-PATH is in NPC

4.(12pts)
K-COLOR. Given a graph G = (V,E), a k-coloring is a function c: V-> {1, 2, ... , k} such that $c(u) \neq c(v)$ for every edge $(u,v) \in E$. In other words the number 1, 2, .., k represent the k colors and adjacent vertices must have different colors. The decision problem K-COLOR asks if a graph can be colored with at most K colors.

a. The 2-COLOR decision problem is in P. Describe an efficient algorithm to determine if a graph has a 2-coloring. What is the running time of your algorithm?

```
Input = G(V,E)
    while a vertex in V is left in G
        if all vertices in V are uncolored do
            for each vertex connected to v referred to as v' do
                if v'.color == v.color return no
                else if v.color == red and v'.color = blue
```

```
                else v'.color =red
            remove v and edges from G
    return true
```

using a BFS search we start at a vertex and make it red and set each other vertex to a different color. alternating between red and blue. we run through checking that the proper vertex return true or false verifying the correct color is int the correct spot.

b. It is known that the 3-COLOR decision problem is NP-complete by using a reduction from SAT. Use the fact that 3-COLOR is NP-complete to prove that 4-COLOR is NP-complete

1) show that 4-color is np

   input: G(V,E, c) graph vertices edges and color for each
   check the number of different colors is <= 4, then color each node on G as specified. Next check each nodes neighbor don't have a color that match the node. If all are different return true if not return false.

   The run time in poly time would be at most O(n*(n-1), this is the worst-case scenario and is poly time so 4-color is np

2) Show that 3-color <=P 4-color

   need to show a poly algorithm to transform an arbitrary instance of 3-color into an instance of 4-color
   let a graph be G=(V,E,c) graph vertices edges and color for each
   let G' be = to G except for the following change, G' has an extra vertex which is connected to the same point that are in G and G' has the same number of edges that G has vertices, then color the vertex the fourth new color and this is the only place this color is used.

   Prove that this transformation holds that 3-color yes if and only if 4-color is also a yes and no if and only if 4-color is a no.
   G' will only pass 4-color if G passes 3-color because the changes from G to G' meant that the questions is whether G or G' pass their respective problems is defendant on if G passes the 3-color. The additional vertex adds to G' are the fourth color and no other vertex has this color so none of the added edges in G' would change anything that has happened in G thus wouldn't cause any conflict with any of the colors in G. In conclusion there would be no case which would break the transformation and since 1 and 2 are both true then 4-color is NPC.