Algorithm Assignment10

Xinyu Jiang

November 2018

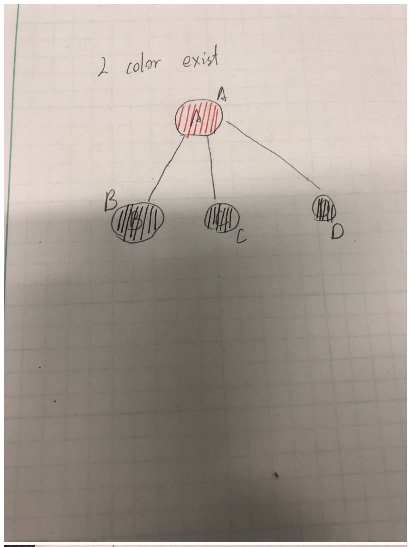
1

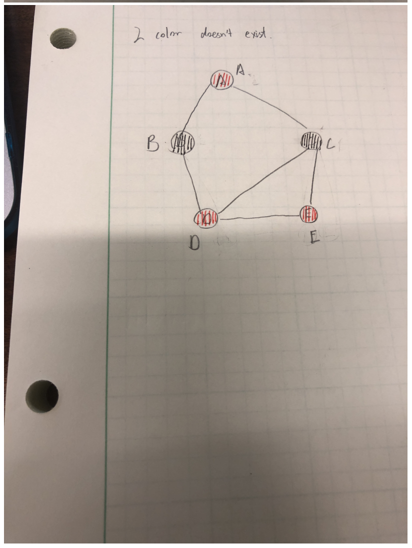
1. (10 pts) Design a polynomial-time algorithm for the graph 2-coloring problem, which determines whether vertices of a given graph can be colored in no more than two colors so that no two adjacent vertices are colored the same color.

   (a) Generate two sample graphs to demonstrate how your algorithm performs. You should have one graph where a 2-coloring exists and one graph where it doesn't exist. Explain how your algorithm performs on each graph. Include your graphs in your assignment submission.

a).
Algorithm description: First, choose a vertex in the graph(could choose it randomly) as start vertex and change the color of that vertex to white. Then check the vertex's neighbor(the previous vertex that just change the color to white), if its neighbor/neighbors have not been assigned color, change the color to the color different with the source vertex(since there are only two colors for this algorithm, so that it could only be black or white) . Keep this step, if at some point find its neighbors have already been assigned color, and if the color is the same with the source vertex, leave the loop(at this point, we have already go through all the point in this graph). Because the algorithm above using BFS to assign all the points, so the running time is O(V+E) which is polynomial.

Note: The red color indicates white in the graph





(35 pts total) Every young wizard learns the classic NP-complete problem of determining whether some unweighted, undirected graph $G = (V, E)$ contains a Hamiltonian cycle, i.e., a cycle that visits each vertex exactly once (except for the starting/ending vertex, which is visited exactly twice). The following figure illustrates a graph and a Hamiltonian cycle on it.

1. (15 pts) Ginny Weasley is working on a particularly tricky instance of this problem for her Witchcraft and Algorithms class, and she believes she has written down a "witness" for a particular graph G in the form of a path P on its vertices. Explain how she should verify in polynomial time whether P is or is not a Hamiltonian cycle. (And hence, demonstrate that the

problem of Hamiltonian Cycle is in the complexity class NP.)

2. (20 pts) For the final exam in Ginny's class, each student must visit the Oracle's Well in the Forbidden Forest. For every bronze Knut a young wizard tosses into the Well, the Oracle will give a yes or no response as to whether a particular graph contains a Hamiltonian cycle. Ginny is given an arbitrary graph G, and she must find a Hamiltonian cycle on it to pass her exam.

   Describe an algorithm that will allow Ginny to use the Oracle to solve this problem by asking it a series of questions, each involving a modified version of the original graph G. Her solution must not cost more Knuts than a number that grows polynomially as a function of the number of vertices in G. (Hence, prove that if we can solve the Hamiltonian cycle *decision* problem in polynomial time, we can solve its *search* problem as well.)

a). For Hamiltonian cycle, because it go through each vertex exactly once. We need to first check if the path is connected(if each vertex in the given path is connected), if the path is not connected, then the path is not a valid path, so it is not a Hamiltonian cycle. If the path is connected, we need to check if all the vertex in the path appear exactly once(expected the starting vertex, because it needs to be connected), if some vertexes appear more than once, then it is not a valid cycle. The running time for this algorithm is $O(n)$(check if the path is connected)+ $O(n^2)$(check if all the vertex appear exactly once, expect the starting vertex), therefor the running time is $O(n^2)$(drop the lower one).
b). For finding the Hamiltonian cycle, because the Oracle only response if the graph contain a Hamiltonian cycle, therefore, if we remove one edge from the graph(not really remove it, just "if" we remove the edge) and ask the Oracle if the graph still contain a Hamiltonian cycle, if it is not a Hamiltonian cycle, we can add the edge to the Hamiltonian cycle array, keep doing this step until all the edges in the graph have been gone through, the array is the solution for Hamiltonian cycle. Because the algorithm go through all the edges in the graph, therefore, the running time is $O(E)$ which is polynomial.

(20 pts extra credit) In the review session for his Deep Wizarding class, Dumbledore reminds everyone that the logical definition of NP requires that the number of *bits* in the witness w is polynomial in the number of bits of the input n. That is, $|w| = poly(n)$. With a smile, he says that in beginner wizarding, witnesses are usually only logarithmic in size, i.e., $|w| = O(\log n)$.

1. Because you are a model student, Dumbledore asks you to prove, in front of the whole class, that any such property is in the complexity class P.

2. Well done, Dumbledore says. Now, explain why the logical definition of NP implies exponential time for any brute force algorithm.

a). For 2 bits, there are 4 possible solution(00, 01, 10, 11) which is 2^n possible solutions. Thus, for logarithmic $|w| = O(\log n)$, the running time is $O(2^{logn})$, which is in P.
b). For NP problem, for example, the travele sales problem(TSF),there are $!(n-1)$ possible paths, because we don't know how much time we need to go(it could be one time, or it could be all the possible paths), so it is non-polynomial. And for brute force algorithm, for example, guessing one password, it is listing all possible solutions, which is also non-polynomial(people may try once to get the password, or they need to try all the passwords). Therefore, the logical definition of NP implies exponential time for any brute force algorithm.