Kaylee Lewis

Assignment 08: Another Way To Soduku
Some Computational Complexity Thought Questions

Due : Fri Apr 19 , 2024 11 : 59pm

2. (15 pts) It is possible to do a problem reduction of sorting a set of numbers to the problem of computing the convex hull of points in a plane. We do this by mapping each number x to a point (x, x^2). This maps each integer to a point on the parabola $y = x^2$. The parabola is convex so every point must be on the convex hull. Furthermore, neighboring points on the convex hull have neighboring x values, the convex hull returns points sorted by the x coordinate. Explain why this problem reduction means that the best we can do performance-wise for a convex-hull algorithm is $O(n * \log(n))$.

- Reducing the sorting problem to the convex hull problem establishes a lower bound of O(n * log n) for the convex hull algorithm due to the proven lower bound of comparison-based sorting. Mapping suggests that sorting is at least as challenging as convex hull computation. The best-known convex hull algorithms have a time complexity of O(n * log n). That is why sorting also requires at least O(n * log n) time complexity.

3. (25 pts) Planar graphs are graphs that can be drawn in such a fashion that no edges cross each other. Propose an algorithm that you might use to reduce the maximum cut problem in a planar graph to the problem of finding a shortest tour in such a graph that visits each edge at least once.

- Algorithm 1: To reduce the maximum cut problem to finding a shortest tour in a planar graph, you could use the following approach

---

Problem

  Input:

    A planar graph.

  Output:

    The edges forming the maximum cut in the original graph.

  Steps:

  - Convert the planar graph into a dual graph.
  - Find a minimum spanning tree in the dual graph.
  - The edges not in the minimum spanning tree form the maximum cut in the original graph.
  - Use an Eulerian tour algorithm on the original graph to visit each edge at least once.

---

4. (10 pts) A brute force algorithm for checking to see if some number n is a composite number iterates though all numbers from 2 to the floor of n/2 and either terminates with a return of “YES” if a number divides n evenly or “NO” if you reach the floor of n/2. Explain why this algorithm does not put the composite number problem into the complexity class P.

- The brute force algorithm for checking composite numbers iterates through n/2 numbers that results in a time complexity of O(n). The algorithm does not put the composite number problem into P because its worst-case time complexity is O(n). O(n) is linear and not polynomial. Complexity class P can be solved in polynomial time which is typically O(n^k) for some constant k. The given algorithm does not work as it scales linearly with the input size n.