CSL7630: Assignment #1

Suman Kundu
suman@iitj.ac.in

CSE, IIT Jodhpur — February 10, 2022

Due Date — February 28, 2022

Instructions

This is the first assignment of the course. All the assignments are from the module 'Property Testing'. There are 3 analytical assignment totaling 60 marks and one coding assignment of 40 marks.

Submit all the analytical proves and the report of the coding assignment in a single PDF file. Also submit the code files (in original text file, e.g., for python submit .py file). Do not submit notebook file. No marks will be given if code is submitted as jupyter notebook or pdf.

Warning: Please do by yourself and remember the policy of 'no tolerance on plagiarism'. Visit the course website for penalties.

1 Testing Majority (20 Marks)

Suppose that two candidates are running for an election. There are n electors each voting for one candidate, and the candidate with more votes will be elected. Of course, all the votes need to be counted to get the final result, but is it possible to get some preliminary result by sampling a small number of votes?

We can rephrase this question as a property testing problem. Suppose that the n votes are stored in an array $A[1], ..., A[n]$. Then, the election of the first candidate can be seen as a property of A, that is, the property is satisfied if the first candidate has more than $n/2$ votes in A. Let P denote this property. Then, A is ϵ-far from P if we must flip more than ϵn votes to elect the first candidate. In other words, the second candidate wins by more than $2\epsilon n$ votes, which means that the second candidate is a clear winner.

Let the value is 1 if the first candidate got the vote and 0 otherwise. Then this problem can be defined as a Majority Testing Problem.

Question 1 *(Majority Testing)*

Construct a tester that given n, ϵ and access to a function $f : [n] \to \{0, 1\}$ accepts with probability $1 - \delta$ if $|f^{-1}(1)| \geq n/2$, rejects with probability $1 - \delta$ if $|f^{-1}(1)| < (1 - \epsilon)n/2$ and makes $O(\epsilon^{-2} \log(\delta^{-1}))$ queries.

Hint: Sum estimator can be of help.

2 Palindromes (20 Marks)

For $n \in \mathbb{Z}_{>0}$, we say that a string s is a palindrome if it reads the same backward or forward. Consider the restricted case where the number of character is even. That is, we define the language as:

$$L_{pal} = \{s \in \Sigma^* | \exists w \in \Sigma^*, s = ww^R\} \tag{1}$$

For a string $s \in \Sigma^n$ (of an even length), we say that an index $i \in [n/2]$ is violating if $s[i] \neq s[n - i + 1]$. The following proposition is immediate.

Proposition 2.1 *A string $s \in \Sigma^n$ is a palindrome if and only if there is no violating index.*

Based on this we can develop our tester as follows:

Algorithm 1: `Palindrome Tester`

Input: Even integer $n, \epsilon \in (0, 1)$, and the query access to the string $s \in \Sigma^n$

for $q = \Theta(\epsilon^{-1})$ *times* **do**
  Sample i from $[n/2]$ uniformly random;
  if $s[i] \neq s[n - i + 1]$ **then**
    Reject;
  end
end
Accept;

Question 2 (*Prove the following theorem*)

Theorem 2.1 *Algorithm 1 is a one-sided error tester for L_{pal} with $O(\epsilon^{-1})$ queries.*

Question 3 *(Design Algorithm)*

Develop an algorithm to test palindrome which can handle string of odd length.

Hint: Try to extend Algorithm 1.

3 Connectivity in Bounded Degree Graph (20 Marks)

Consider the bounded degree graph model where each node can have at most d number of neighbors. In this model our goal is to identify whether the graph is connected or ϵ-far from being connected.

Question 4 *(Prove the following proposition)*

Proposition 3.1 *If a graph G is ϵ-far from being connected, then the number of connected components in G is at least ϵdn.*

We have the following corollary of Proposition.

Question 5 *(Prove the following corollary)*

Corollary 3.1 *If a graph G is ϵ-far from being connected, then the number of connected components of size at most $\frac{2}{\epsilon d}$ in G is at least $\frac{\epsilon dn}{2}$.*

Note that we can reject if we detect a connected component (of size less than n). Corollary 3.1 states that, if the input graph is ϵ-far from being connected, then there are $\Omega(n)$ many small connected components, where we say that a connected component is small if its size is at most $\frac{2}{\epsilon d}$. In particular, this means that we can easily obtain a vertex in a small connected component by sampling vertices uniformly at random. Also, we can easily detect that the obtained vertex belongs to a small connected component by conducting a BFS. The discussion above suggests the following algorithm.

Algorithm 2: `Graph Connectivity`

Input: Integer $n, \epsilon \in (0, 1)$, and the query access to a graph G on n vertices.

for $q = \Theta(\frac{1}{\epsilon d})$ *times* **do**
  Sample a vertex $v \in V$ uniformly random;
  Construct a BFS from v until it reaches $\frac{2}{\epsilon d}$ vertices or no more vertices left.;
  if *we found a connected component of size less than n* **then**
    Reject;
  end
end
Accept;

Question 6 *(Prove the following lemma)*

Lemma 3.1 *Algorithm 2 is a one-sided error tester for connectivity with query complexity* $O(\frac{1}{\epsilon^2 d})$.

4 Coding Assignment: Graph Connectivity Query (40 Marks)

Attached are 50 graphs in the form of adjacency list. The maximum degree is bounded by 20. Some of them are connected, some are disconnected. With these graph perform the following:

Question 7

1. Write an exact algorithm to test whether the graph is connected or not.
2. Write a program to identify whether a graph is ϵ-far from being connected or not.
3. Write a program to test the connectivity of the graphs with $\epsilon = 0.001$. The program should accept the graph if it is connected and reject with high probability if it is ϵ-far from connected.
4. Report the count of the following:
   (a) For each graph, report the number of query required for all the above three algorithms.
   (b) Number of times a connected graph (identified by #1) is accepted and rejected by #3.
   (c) Number of times a disconnected graph (identified by #1) is accepted and rejected by #3.
   (d) Number of times an ϵ-far graph (identified by #2) is accepted and rejected by #3.

You can use your favourite programming language. Use of any existing graph library is not recommended.