CSL7630: Algorithms for Big Data

Assignment 1: Property Testing

Saurabh Shashikant Burewar (B18CSE050)

All analytical questions are hand-written followed by the report of the coding question.

Rest of the page is blank. Please move to next page

Date : Page :

~~CSL 7630~~

CSL 7630. Assignment 1

Saurabh Buvewar
B18CSE050

i) We have an array $A[1] \ldots A[n]$ which contains n votes.
The algo first samples $i_1 \ldots i_q$ from this array uniformly randomly where $q = \Theta(\epsilon^{-2} \log 1/\delta)$

↳ Since we are running for $\log \delta^{-1}$, success prob. is improved to $(1-\delta)$

This algo has access to function $f : [n] \to \{0, 1\}$. This function takes samples and estimates x to give a result of 0 or 1 for satisfaction of property A.

So, if estimate is more than $n/2$, it accepts with probability $(1-\delta)$ and if it is less than $\left(\frac{n}{2} - \frac{\epsilon n}{2}\right)$, it rejects with $1-\delta$.

↓

It is ϵ far, if we have ϵn votes less than required.

So, for $t = 1$, $|f^{-1}(1)| \geq \epsilon n$.
accepts $|f^{-1}(1)| \geq n/2$ and rejects $|f^{-1}(1)| \leq \frac{n}{2} - \frac{\epsilon n}{2}$

So, this algorithm acts as a tester which samples at $\Theta(\epsilon^{-2} \log 1/\delta)$ and accepts with prob. $(1-\delta)$ if $|f^{-1}(1)| \geq n/2$ and rejects with $(1-\delta)$ if $|f^{-1}(1)| \leq \left(\frac{n}{2} - \frac{\epsilon n}{2}\right)$.

2) To prove algo is one sided tester, have to prove that it rejects with $\geq 2/3$ for ϵ-far from palindrome.
For any int $i \in [0, n]$

0 be the no. of times $f(n-i+1) = 0$ if $f(i) = 1$

2 be the no. of time $f(n-i+1) = 1$ if $f(i) = 0$.

Date : Page :

$d_H = \underbrace{0+z}$

↳ these are the bits that are wrong and need to be changed to make it palindrome.

Here $\epsilon \in (0,1)$,

so, f is ϵ-far from P if, $d_H > \epsilon n$

$0+z > \epsilon n$

Pr [a sample $i < i^*$ with $f(i)=0$, $f(n+i+1)=1$]

Pr [a sample $j > j^*$ with $f(j)=1$, $f(n-j+1)=0$]

3) For a string with odd number of characters, the middle character is at index $(n+1)/2$.

1 0 1 0 1 1 0 1 1 0 1 0 1

same.

We same i from $[(n-1)/2]$ and check if $s[i] \neq s[n-i+1]$.

Algorithm :-

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

if n is even,

follow algorithm 1.

if n is odd,

Date : Page :

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

4) We define a graph G to be ϵ-far from property P (connectedness) if we need to add ϵdn edges to G in order to obtain a graph in P.

So, if a graph G is indeed ϵ-far from being connected and has less than ϵdn components, then, we can make this graph connected in $\epsilon dn - 1$ edges.

Lets say we have a graph ϵ-far from connected and less than ϵdn components, namely, $C_1, C_2, \ldots C_k$. Now, add edge (C_i, C_{i+1}) where $i \in (1, k-1)$. Also add one last edge to complete i.e. (C_k, C_1). The graph is now connected. This is done in $\epsilon dn - 1$ edges which contradicts what we said before.

5) Assuming there are more than $\frac{\epsilon dn}{2}$ components of size $> \frac{2}{\epsilon d}$

Here, size is number of vertices, so total number of vertices in the graph will be product of no. of components and size of each comp.

$\Rightarrow \frac{\epsilon dn}{2} \cdot \left(\frac{k2}{\epsilon d}\right) \geq kn$, where $k > 1$

This says there are more than n vertices which is contradiction.

Date : Page :

6) Any graph which is connected with skip the condition and get accepted. We have to prove that algo rejects graphs ϵ-far from being connected with $\geq 2/3$.

From the corollary in Q5, we can say that there are more than $\frac{\epsilon dn}{2}$ small components.

$Pr[\text{missing one sample}] \geq 1 - \frac{\epsilon dn}{2n} = 1 - \frac{\epsilon d}{2}$

$Pr[\underset{A}{\text{missing}} \text{ a vertex in all samples}] \geq \left(1 - \frac{\epsilon d}{2}\right)^k$

where, k = number of samples.

$\geq e^{\left(-\frac{\epsilon d}{2} \cdot k\right)}$ ← Very small value.

$Pr[!A] = 1 - e^{\left(-\frac{\epsilon d}{2} \cdot k\right)}$

For $k = \Theta(1/\epsilon d)$, $1 - e^{\left(-\frac{\epsilon d}{2} \cdot k\right)} \geq 2/3$.

Proved

Coding Assignment

Saurabh Shashikant Burewar (B18CSE050)

Requirements

The algorithms are implemented in Python 3.8 which is the only requirement.

Graphs

The graphs used are in the form of an adjacency list. Since I am using python, I have stored it in a dictionary, where every key is a node and the value of the key is the list of nodes that the key node has an edge with. One observation I got from looking at the adjacency lists is that every graph here has a key with the the value as an empty list, meaning at least one node which doesn't have an edge with any other node.

Algorithms

1. Graph Connectivity

   The first algorithm performs a DFS and checks if all nodes are connected. It checks for connectivity from both directions of edges here. But, as observed from the adjacency list, every graph has a node which has no edge, so there are no connected graphs. The number of queries in this algorithm ranges from 90 to almost 600 with an average number of queries around 300.

   Results -

   graphs\01-bounded-d-20-c-0.adjlst - Not Connected, Queries - 441
   graphs\02-bounded-d-20-c-8.adjlst - Not Connected, Queries - 356
   graphs\03-bounded-d-20-c-27.adjlst - Not Connected, Queries - 219
   graphs\04-bounded-d-20-c-0.adjlst - Not Connected, Queries - 580
   graphs\05-bounded-d-20-c-10.adjlst - Not Connected, Queries - 405
   graphs\06-bounded-d-20-c-30.adjlst - Not Connected, Queries - 138
   graphs\07-bounded-d-20-c-0.adjlst - Not Connected, Queries - 550
   graphs\08-bounded-d-20-c-8.adjlst - Not Connected, Queries - 554
   graphs\09-bounded-d-20-c-19.adjlst - Not Connected, Queries - 306
   graphs\10-bounded-d-20-c-7.adjlst - Not Connected, Queries - 544
   graphs\11-bounded-d-20-c-27.adjlst - Not Connected, Queries - 91
   graphs\12-bounded-d-20-c-15.adjlst - Not Connected, Queries - 203
   …

2. ϵ-far from connected

   This is an implementation of the algorithm from the previous section of the assignment. Here, we are sampling x vertices uniformly random and performing BFS from each of them. We stop if we reach the $2x$ vertices or cannot proceed further. If the required number of vertices are not reached in any of the BFS runs, then we reject. Otherwise, we accept. The number of queries reduced significantly in this algorithm but they vary since the sampling is random and the algorithm is sequential (not parallel), so one failed BFS stops the algorithm. Also, I found that all graphs are marked not connected, while running the algorithm 20 times.

   Results -

   graphs\01-bounded-d-20-c-0.adjlst - Not Connected, Queries - 2
   graphs\02-bounded-d-20-c-8.adjlst - Not Connected, Queries - 4
   graphs\03-bounded-d-20-c-27.adjlst - Not Connected, Queries - 3
   graphs\04-bounded-d-20-c-0.adjlst - Not Connected, Queries - 43
   graphs\05-bounded-d-20-c-10.adjlst - Not Connected, Queries - 3
   graphs\06-bounded-d-20-c-30.adjlst - Not Connected, Queries - 8
   graphs\07-bounded-d-20-c-0.adjlst - Not Connected, Queries - 92
   graphs\08-bounded-d-20-c-8.adjlst - Not Connected, Queries - 3
   graphs\09-bounded-d-20-c-19.adjlst - Not Connected, Queries - 7
   graphs\10-bounded-d-20-c-7.adjlst - Not Connected, Queries - 2
   graphs\11-bounded-d-20-c-27.adjlst - Not Connected, Queries - 2
   graphs\12-bounded-d-20-c-15.adjlst - Not Connected, Queries - 147
   …

   For more detailed results, run the program to get results. The file “1.py” contains the first algorithm and “2.py” has the approximate algorithm.