CSC263 Assignment 4

Yongzhen Huang, Xinyi Liu, Kewei Qiu

February 28th, 2019

Question 1

Written by: Yongzhen Huang Checked by: Kewei Qiu, Xinyi Liu

a)

There are $\frac{k}{n}$ of A that is filled with x. So in the first iteration, the probability of returning TRUE is the probability of hitting x so it is $\frac{k}{n}$.

b)

Consider the probability of this algorithm returning FALSE. This occurs when not one iteration ends up hitting x. For each iteration, such probability of not hitting is $1 - \frac{k}{n}$. So over r iterations, the probability of not hitting is $(1 - \frac{k}{n})^r$. ONLY in this case does the program return FALSE. So overall,

$$\Pr(\text{TRUE}) = 1 \text{ - } \Pr(\text{FALSE}) = 1 - (1 - \tfrac{k}{n})^r$$

c)

Let $N :=$ number of iterations, then

$$\begin{aligned}
E[N] &= \sum_{i=1}^{\infty} i \times Pr(\text{end at loop } i) \\
&= \sum_{i=1}^{\infty} i \times \left(1 - \frac{k}{n}\right)^{i-1} \frac{k}{n} \\
&= p \sum_{i=1}^{\infty} i \times (1-p)^{i-1} && (\text{Let } p = \tfrac{k}{n}) \\
&= p \left[-\frac{d}{dp} \sum_{i=1}^{\infty} (1-p)^i \right] \\
&= p \left[-\frac{d}{dp} (\frac{1-p}{p}) \right] \\
&= \frac{1}{p} = \frac{n}{k}
\end{aligned}$$

Therefore, the expected number of iteration is $\frac{n}{k}$.

Question 2

Written by: Xinyi Liu Checked by: Kewei Qiu, Yongzhen Huang

For this question, we use disjoint sets implemented by forest structure with Weighted Union by size. The algorithm is as follows: First we create n independent sets for each distinct variables $x_1, x_2, ..., x_n$. Then we loop through m constraints. If the constraint is an *equality*, we perform $Find$ on each side of the equality and we will get the set(s) that contain(s) these two variables. If these two variables are not in the same set, we do $Union$ on these two sets. After the first loop, we loop through m constraints again. This time, if the constraint is an *inequality*, we perform $Find$ on each side of the constraint and we will get the set(s) that contain(s) these two variables. If these two variables are in the same set or these two variables are the same, the algorithm stops and return NIL, otherwise it keeps iterating. After all the iterations, we assign a distinct integer to each set, that is to say, all variables in a set will be assigned to a same number and different variables from different sets will have different values.

Time complexity analysis: First we iterate n variables which takes $O(n)$. Since creating set takes $O(1)$, the total time of the first step is $O(n)$. Then for the first loop, since $Find$ takes $O(\log n)$ and $Union$ takes $O(1)$ each time by what we have learnt in the class, it takes at most $O(2\log n + 1)$ which is just $O(\log n)$ each iteration. Then the whole loop costs $O(m \log n)$. For the second loop, the worst case is that the loop never stops and never returns NIL. In this case, each iteration costs $O(\log n)$ for $Find$ operation. So the total cost is $O(m \log n)$. In the end, we assign integers to each set. Since there are at most n sets, this step costs at most $O(n)$. Adding all things up, the time complexity is $O(2n + 2m \log n)$ which is $O(n + m \log n)$. This is asymptotically better than $O(mn)$.