CS-E4600 — Homework #2

fall 2018

Due: Fri, Oct 19, midnight

Return your answers as a PDF file via `mycourses.aalto.fi`

Discussing the problems with your colleagues is allowed. However, you should write the answers by yourself. If you discuss with others, give proper acknowledgments. Looking for the answers in the internet is discouraged. You should at least make a serious effort to solve a problem by yourself before looking online. If you do, however, give proper acknowledgments.

Remember that there is a budget of 5 late days, which you can use in any way you want for the three homeworks and the programming assignment. Weekend days count as late days.

Typed solutions are strongly encouraged, especially if your hand writing is messy.

"I do not know" answers receive 15% of the max score for each problem. Partial credit will be given for partial solutions, but long off-topic discussion that leads nowhere may receive 0 points. Overall, think before you write, and try to give concise and crisp answers.

Problem 1 [30 points]

In class we discussed about representing documents as *sets of words*, i.e., each document is represented by the set of the words it contains. For example, the document *a b a c d a c* is represented by the set $\{a, b, c, d\}$.

In many cases, in order to have a more accurate representation, we use *bags of words* (or *multisets of words*). According to this model, in addition to keeping the words in the document, we also record the *number of times* that each word appears. For example, the document *a b a c d a c* is represented by the bag $\{(a, 3), (b, 1), (c, 2), (d, 1)\}$.

To deal with bags, we define some useful operations:

Bag size. The size of a bag A is defined as

$$||A|| = \sum_{(x,n)\in A} n.$$

Bag union. The bag union $\sqcup$ between two bags A and B is defined as

$$A \sqcup B = \{(x, \max\{n, m\}) \text{ where } (x, n) \in A \text{ and } (x, m) \in B\}.$$

Bag intersection. The bag intersection $\sqcap$ between two bags A and B is defined as

$$A \sqcap B = \{(x, \min\{n, m\}) \text{ where } (x, n) \in A \text{ and } (x, m) \in B\}.$$

For example, if $A = \{(a, 3), (b, 1), (c, 2), (d, 1)\}$ and $B = \{(a, 1), (b, 2), (d, 4), (e, 2)\}$, we have

$$||A|| = 7, \quad ||B|| = 9,$$

$$A \sqcup B = \{(a, 3), (b, 2), (c, 2), (d, 4), (e, 2)\},$$

and

$$A \sqcap B = \{(a, 1), (b, 1), (d, 1)\}.$$

We also extend the Jaccard coefficient between bags as

$$J(A, B) = \frac{||A \sqcap B||}{||A \sqcup B||}.$$

Question 1.1. Argue that the extension of the Jaccard coefficient to bags, as defined above, is meaningful and well-motivated.

Question 1.2. Provide a locality-sensitive hashing (LSH) scheme for the Jaccard coefficient to bags. In other words, design a family of hash functions $\mathscr{F}$ such that

$$\Pr[f(A) = f(B)] = J(A, B),$$

when f is drawn uniformly at random from $\mathscr{F}$.

Prove that the previous equality holds.

Question 1.3. Discuss how exactly you will implement the locality-sensitive hashing scheme you designed. Provide pseudocode for finding similar documents in a document collection given a query document, where documents are represented as bags of words. Your pseudocode should describe both the preprocessing and the querying part of the similarity-search algorithm.

Problem 2 [20 points]

Consider an array X of m numbers, where each number $X[i]$ is drawn independently and uniform at random from the interval $(0,1)$. Consider a simple linear algorithm to compute the maximum number in X: start with $\texttt{max} = -\infty$ and scan the array; each time you encounter a number $X[i]$ that is larger than the current value of max update its value with $\texttt{max} = X[i]$.

Show that the update step ($\texttt{max} = X[i]$) will be executed $\mathcal{O}(\log m)$ times, on expectation.

Use this result to argue that the "priority sampling for sliding window" discussed in class requires space $\mathcal{O}(\log w \log n)$. Recall that w is the length of the sliding window, and n the size of the universe where numbers are drawn from.

Problem 3 [10 points]

In class we discussed the "reservoir" algorithm for sampling 1 element in a data stream. The algorithm has the property that it obtains a uniform sample. Uniform here means the following: Consider the algorithm at any time t. Up to that point the algorithm has seen t items in total. Any of these t items has equal probability to be the sampled item.

In this problem we want to make a simple extension of the "reservoir" algorithm to sample k items. Again we want to obtain a uniform sample. We are thinking that the following algorithm is a meaningful extension of the 1-sample reservoir algorithm.

Algorithm k-RESERVOIR
Initialize array $S[1...k]$
$t \leftarrow 1$
while (1)
    read input x_t
    if ($t \leq k$)
        $S[t] \leftarrow x_t$
    else
        with probability p
            $j \leftarrow$ random$[1..k]$
            $S[j] \leftarrow x_t$
    endif
    $t \leftarrow t + 1$
endwhile

Question 3.1. Write in English what does it mean that a k-sample in a data stream is *uniform*.

Question 3.2. What should the value of the probability p be for the k-RESERVOIR algorithm to give uniform samples?

Question 3.3. Prove that for the value of p that you specified in **4.2** the k-RESERVOIR algorithm gives indeed uniform samples.

Problem 4 [40 points]

We consider a set of items U. The size of U is potentially very large. A subset G of U has a property of interest. We say that G are the "good" items of U. We want to estimate the number of good items $|G|$. We assume that the fraction of good items is $\rho = |G|/|U|$. We also assume that given an item of U we can easily check whether it is a good item.

The challenge is that as U is very large we cannot enumerate all its items. So we are thinking to resort to sampling. We consider the following MONTECARLO sampling algorithm, which returns a variable Z as an estimate of $|G|$.

Algorithm MONTECARLO
1. Sample a random subset $X \subseteq U$, of size $|X| = N$
2. Let $Y \subseteq X$ be the items in X that are good
3. Return $Z = \frac{|U||Y|}{N}$

Question 4.1. Describe a real-world application that you may consider applying the above algorithm.

Question 4.2. Explain the intuition of MONTECARLO algorithm.

An important question is how large the sample size N should be so as to obtain a good approximation or $|G|$. We are interested in (ϵ, δ) approximations. In particular, given numbers $\epsilon, \delta > 0$, we say that Z is an (ϵ, δ)-approximation of $|G|$ if the following is true:

$$Pr\left[(1-\epsilon)|G| \leq Z \leq (1+\epsilon)|G|\right] \geq 1-\delta. \tag{1}$$

Question 4.3. Explain Equation (1) in English.

Question 4.4. Use the Chernoff bound to show that if

$$N \geq \frac{4}{\epsilon^2 \rho} \ln \frac{2}{\delta}, \tag{2}$$

then the MONTECARLO algorithm returns an estimate Z that is an (ϵ, δ)-approximation of $|G|$.

Question 4.5. Discuss the bound (2) in terms of the dependence of N with respect to the approximation parameters ϵ and δ and the problem parameters $|U|$ and ρ. What are the implications of the bound (2)?