Homework 2 for CMSC 498U/644

Due March 1

February 20, 2018

1 Problem 1

Recall the second moment estimation algorithm we covered in class, please show step by step how it runs on the following stream (suppose the underlined letters are the ones you sampled):

$$a, a, \underline{a}, a, \underline{b}, a, a, c, a, b, \underline{b}, b, b, a, \underline{c}$$

2 Problem 2

In class, we talked about an algorithm that estimate second moment in streams. There is another algorithm (1996; Alon, Matias, Szegedy) that can accomplish the same task. Assume we have a (2-universal) hash function $h : x \to \{-1, 1\}$, each value with probability 1/2. "2-universal" guarantees that the hash value of any two possible inputs x_1, x_2 is independent. Or, to make it simple, you can use $\Pr[h(x_1) = v_1 \text{ and } h(x_2) = v_2] = \Pr[h(x_1) = v_1] \cdot \Pr[h(x_2) = v_2]$.

Algorithm 1: AMS-F_2-Estimate

```
Select h ∈ H, a 2-universal hash family from ℕ to {−1, 1};
Z ← 0;
for x in stream do
    Z ← Z + h(x);
end
return Z^2;
```

Please prove that the expected answer is exactly the second moment.

3 Problem 3

3.1 Part (a)

We covered the $O(\sqrt{n})$ space algorithm to find the median in a stream with two passes. Please implement this algorithm using your favourite language, and run it on supplied input file. Here we recap the algorithm:

Here are the details:

1. Assume you know n.
2. Use the first $\lfloor \sqrt{n} \rfloor$ numbers as separators.

Input format:

1. The first line is n,

Algorithm 2: Finding median in a stream

```
1  n ← number of elements in the stream ;
2  Let k ← ⌊√n⌋, and read the first k elements;
3  Sort them into x_1, ..., x_k, and use them to set up empty buckets B_0, ..., B_k, where B_i holds
     elements in (x_{i-1}, x_i] (B_0 holds elements in (−∞, x_1], B_k holds elements in (x_k, ∞));
4  for x in stream do                                                    /* First loop */
5  |  Increase the size of corresponding bucket by 1;
6  end
7  Find the bucket B_j that contains the median.;
8  for x in stream do                                                   /* Second loop */
9  |  if x ∈ B_j then
10 |  |  Record x;
11 |  end
12 end
13 Find the corresponding element in the recorded elements.
```

2. The remaining n lines each contains an unsigned integer number guaranteed to fit unsigned 32-bit integer

Output format: the median number. To make things simple, we define median to be the $\lceil n/2 \rceil$ smallest number.

For example, an input can be:

```
10
69
46
64
24
16
73
37
32
71
56
```

which defines a stream of size 10, with the following elements: 69, 46, 64, 24, 16, 73, 37, 32, 71, 56 . You would use the first 3 elements to set up 4 buckets: $[24, 16, 37, 32]$, 46, [56], 64, [], 69, $[73, 71]$, or to simplify things: $[24, 16, 37, 32, 46]$, $[56, 64]$, [69], $[73, 71]$. However, we do not have space to store everything, and what we actually store is the size of each bucket: $5, 2, 1, 2$. Since $n = 10$, we want the 5-th element, which should lie in the first bucket, and being the 5-th smallest element in this bucket. So we go over the stream again, and record all elements in the first bucket, i.e. $[24, 16, 37, 32, 46]$, and findout that 46 is the median we are looking for. So the output can be:

```
46
```

Corresponding input files: `input_0.txt`, `input_1.txt`, `input_2.txt`. Please also include your source code.

3.2 Part (b)

Modify your code so that it can find the k-th smallest element. Everything is the same except for the first line of the input, which includes two numbers n and k, separated by a single space. Example input:

```
10 7
69
46
```

```
64
24
16
73
37
32
71
56
```

Example output:

```
64
```

Corresponding input files: `input_k_0.txt`, `input_k_1.txt`, `input_k_2.txt`. Please also include your source code.