Assignment 2

Advanced Algorithms and Datastructures

Authors:
Jenny-Margrethe Vej (rwj935)
Martin Gielsgaard Grünbaum (wrk272)
Martin Nicklas Jørgensen (tzk173)

May 21, 2015

1 Hash functions for sampling

1.1 Exercise 1(a')

We must show that $p \leq Pr[h_m(x)/m < p] \leq 1.01p$. To do so, we first look at finding a different way to express the probability of sampling (i.e. probability of $h_m(x)/m < p$). We then make use of various re-writes and the fact that h_m is strongly universal, to re-express the equality and find a tight bound.

We are given some $p \geq 100/m$, a suitably large m and a strongly universal hashing function $h_m : U \to [m]$. Note that $p \geq 100/m$ implies that $p/100 \geq 1/m$ and that for some generic a (such as pm in the following), $a \leq \lceil a \rceil < a + 1$.

We observe that

$$\begin{aligned}
&Pr[h_m(x)/m < p] \\
&= \sum_{0 \leq k < pm} Pr[h_m(x) = k] \\
&= \sum_{0 \leq k < pm} \frac{1}{m} \\
&= \frac{1}{m} \sum_{0 \leq k < pm} 1 \\
&= \frac{1}{m} \left| [0, pm) \right| \\
&= \frac{1}{m} \cdot \lceil pm \rceil \\
&= \frac{\lceil pm \rceil}{m}
\end{aligned}$$

Therefore, we have that

$$p = \frac{pm}{m} \leq Pr[h_m(x)/m < p] = \frac{\lceil pm \rceil}{m} \leq \frac{pm + 1}{m} \leq p + \frac{p}{100} = 1.01p$$

1.2 Exercise 1(b)

The probability of a collision ($h_m(x)/m = h_m(y)/m$) is the probability that there exists two elements in A such that they hash to the same thing. Therefore, we have that:

$$
\begin{aligned}
& Pr[\exists \{x, y\} \in A : h_m(x)/m = h_m(y)/m] \\
& \leq \sum_{\{x,y\} \in A} Pr[h_m(x)/m = h_m(y)/m] && \text{Union bound} \\
& = \frac{\binom{|A|}{2}}{m} && \frac{1}{m} \text{ probability for each pair } \{x, y\} \\
& \leq \frac{|A|(|A|-1)}{2 \cdot 100|A|^2} && \text{Because } m \geq 100|A|^2 \\
& \leq \frac{|A|(|A|-1)}{200|A|^2} \\
& \leq \frac{1}{200} && \text{As the numerator is } \textbf{almost } |A|^2
\end{aligned}
$$

2 Bottom-k sampling

2.1 Exercise 2

We assume that C and $S_h^k(A)$ are independently drawn, and that $S_h^k(A)$ is a uniformly random subset of A of size k. We know that the probability of an element in A being in the bottom-k sample $S_h^k(A)$ is $p = k/n$.

The probability that a randomly drawn element from A is also in the subset C must be $|C|/|A|$. Intuitively, as C grows in size, so does the probability of picking an element from A that is also in C.

Then we have that

$$
\begin{aligned}
& \mathbb{E}\left[|C \cap S_h^k(A)|\right] \\
& = \sum_{a \in A} \mathbb{E}\left[a \in C \wedge a \in S_h^k(A)\right] \\
& = \sum_{a \in A} Pr\left[a \in C \wedge a \in S_h^k(A)|\right] \\
& = \sum_{a \in A} \left(Pr\left[a \in C\right] \cdot Pr\left[a \in S_h^k(A)\right]\right) && (1) \\
& = \sum_{a \in A} \left(\frac{|C|}{|A|} \cdot \frac{k}{|A|}\right) && (2)
\end{aligned}
$$

where (**??**) because the probability of the events is independent, and (**??**) makes use of (**??**) as well as the probability for an element of A being in $S_h^k(A)$.

Finally we have that:

$$
\begin{aligned}
&\mathbb{E}\left[\frac{|C \cap S_h^k(A)|}{k}\right] \\
&= \frac{1}{k} \cdot \mathbb{E}\left[|C \cap S_h^k(A)|\right] \\
&= \frac{1}{k} \cdot \sum_{a \in A} \left(\frac{|C|}{|A|} \cdot \frac{k}{|A|}\right) && \text{See (??)} \\
&= \frac{1}{k} \cdot \frac{|C|}{|A|} \cdot \frac{k}{|A|} \cdot \sum_{a \in A} 1 && \text{Components of sum do not depend on } a \in A \\
&= \frac{1}{k} \cdot \frac{|C|}{|A|} \cdot \frac{k}{|A|} \cdot |A| \\
&= \frac{1}{k} \cdot \frac{|C|}{|A|} \cdot k \\
&= \frac{|C|}{|A|}
\end{aligned}
$$

2.2 Exercise 3(a)

We would store the data in a max-heap structure H where we only store the k smallest keys. Since we only want to store a key if it is smaller than the biggest element in the heap, and the biggest element is on the top of the heap, we can do this check in $O(1)$ time. And furthermore insert elements in $O(\lg\, k)$ time.

2.3 Exercise 3(b)

As written above we would be able to process/insert the next key in $O(\lg\, n)$ time.

2.4 Exercise 4(a)

We will prove the equality $S_h^k(\mathrm{A} \cup \mathrm{B}) = S_h^k(S_h^k(A) \cup S_h^k(B))$. We can see each set as a sorted stack that keeps the smallest values at the top. The left hand part of the equality ($S_h^k(\mathrm{A} \cup \mathrm{B})$)) corresponds to merging the two stacks and taking the k top values. The right hand side ($S_h^k(S_h^k(A) \cup S_h^k(B))$) corresponds to taking the k topmost values from both stacks and then merging them and taking the k smallest values from the resulting stack.

Since we take the k smallest values from each stack we are guaranteed to have the smallest value from the union of A and B since even if the smallest values in B is bigger than the biggest values in A we still have the k smallest values from A, thus proving the equality.

2.5 Exercise 4(b)

We want to prove the equality $A \cap B \cap S_h^k(A \cup B) = S_h^k(A) \cap S_h^k(B) \cap S_h^k(A \cup B)$.

The lefthandside corresponds to taking the k smallest keys in the union of A and B and then eliminating all entries that is not also in both A and B. The righthandside is the same as taking

the k smallest keys in the union of A and B and then eliminating all the entries that is not part of the k smallest keys in both A and B.

Since $S_h^k(A \cup B)$ naturally limits both sides of the equality to only the k smallest entries in the union, $A \cap B$ gets limited by the intersect with $S_h^k(A \cup B)$.

2.6 Exercise 4(c)

We can divide

$$|S_h^k(A) \cap S_h^k(B) \cap S_h^k(S_h^k(A) \cup S_h^k(B))|/k$$

into several smaller pieces in order to estimate it's running time, on psudo BNF it will be

$$\begin{aligned} full &= |sets|/k \\ sets &= S_h^k(A) \cap S_h^k(B) \cap ext \\ ext &= S_h^k(union) \\ union &= S_h^k(A) \cup S_h^k(B) \end{aligned}$$

With these smaller fragments it will be easier to calculate the runningtime and see where the different time consumers are. We will start from the bottom and move up

union Assuming we're using our minimum-heap structure proposed in **??** $S_h^k(A)$ and $S_h^k(B)$ can both be done in $O(k \lg n)$. The union can be done in $\Theta(2k)$. Giving this part a running time of $O(k \lg n)) + \Theta(k)$.

ext The output from *union* now contains $2k$ elements meaning we can get the k smallest in $O(k \lg 2k))$ time.

sets Each of the extractions in this part can be done in $O(k \lg n)$ time, and since the heaps are ordered the intersect requires only linear time giving running time of $O(k \lg n) + O(k)$. The intersect with the *ext* part requires linear runningtime in the number of elements in the largest heap, since both are of maximum size k, the final time is $O(k \lg n)) + O(k) + O(k)$.

full Since the number of entries in the heap can be decided in $O(1)$ time if we assume the heap has an associated peorperty with the number of nodes that are updated in all operations (also done in constant time), and the division itself is also in constant time. This means we just need to determine the runningtime of all the above segments combined.

$$\begin{aligned} union &= O(k \lg n)) + \Theta(k) \\ ext &= O(k \lg 2k)) \\ sets &= O(k \lg n)) + O(k) + O(k) \\ full &= O(1) + O(k \lg n)) + O(k) + O(k) + O(k \lg 2k)) + O(k \lg n)) + \Theta(k) \\ &= O(k \lg n)) \end{aligned}$$

3 Bottom-k sampling with strong universality

3.1 Exercise 5

We would like to prove that if (I) and (II) are both false, then so is (4). Because a and b have been fixed as pr. the assignment text, we have that $p = \frac{k}{n(1-a)}$. We also recall that $n = |A|$.

We assume the opposite of (I) and (II) to be true:

(I‘) The number of elements from A that hash below p is greater than k.

(II‘) The number of elements from C that hash below p is less than or equal to $(1+b)p|C|$.

(I‘) implies that all elements of S hash below p too, since there are k elements in S and there are more than k elements in A that hash below p. Per (II‘), $|C| \leq (1+b)p|C|$ and therefore also $|C \cap S| \leq (1+b)p|C|$. *Given* (I‘) and (II‘), we want to arrive at $|C \cap S| \leq \frac{1+b}{1-a}fk$, and if we do so we have proven (I) or (II) by contraposition.

$$
\begin{aligned}
|C \cap S| \leq & (1+b)p|C| \\
= & (1+b)\frac{k}{n(1-a)}|C| \\
= & (1+b)\frac{k}{|A|(1-a)}|C| && n = |A| \\
= & (1+b)\frac{k}{1-a}f && \text{Because } f = |C|/|A| \\
= & \frac{(1+b)k}{1-a}f \\
= & \frac{1+b}{1-a}fk
\end{aligned}
$$

3.2 Exercise 6

We have that

$$k = \mu_A(1-a) = \mu_A(1 - r\sqrt{k}) \tag{3}$$

$$u_A = E[X_A] = pn = k/(1-a) \tag{4}$$

$$Pr\left[|X - \mu| \geq r\sqrt{\mu}\right] \leq \frac{1}{r^2} \tag{5}$$

We wish to prove that $P_{(I)} = Pr\left[X_A < k\right] \leq 1/r^2$:

$$
\begin{aligned}
& Pr\left[X_A < k\right] \\
=& Pr\left[X_A < \mu_A(1 - \frac{r}{\sqrt{k}})\right] && \text{From (??)} \\
=& Pr\left[X_A < \mu_A - \frac{\mu_A \cdot r}{\sqrt{k}}\right] \\
=& Pr\left[X_A - \mu_A < -\frac{\mu_A \cdot r}{\sqrt{k}}\right] \\
=& Pr\left[\mu_A - X_A > \frac{\mu_A \cdot r}{\sqrt{k}}\right] && \text{Multiplying with -1} \\
\leq& Pr\left[|X_A - \mu_A| > \frac{\mu_A \cdot r}{\sqrt{k}}\right] \\
\leq& Pr\left[|X_A - \mu_A| > \frac{\mu_A \cdot r}{\sqrt{\mu_A}}\right] && \text{(??) implies } \mu_A > k \\
=& Pr\left[|X_A - \mu_A| > r\sqrt{\mu_A}\right] && \text{Because } \frac{a}{\sqrt{a}} = \sqrt{a} \\
\leq& Pr\left[|X_A - \mu_A| \geq r\sqrt{\mu_A}\right] \\
\leq& 1/r^2 && \text{See (??)}
\end{aligned}
$$

3.3 Exercise 7

This is the same approach as Exercise 6.

$$
\begin{aligned}
& Pr\left[X_C > (1 + \frac{r}{\sqrt{fk}})\mu_C\right] \\
\leq& Pr\left[X_C > (1 + \frac{r}{\sqrt{\mu_C}})\mu_C\right] && \text{Because } \mu_C > fk \\
=& Pr\left[X_C > \mu_C + \mu_C \cdot \frac{r}{\sqrt{\mu_C}}\right] \\
=& Pr\left[X_C - \mu_C > \mu_C \cdot \frac{r}{\sqrt{\mu_C}}\right] \\
\leq& Pr\left[|X_C - \mu_C| > r\sqrt{\mu_C}\right] \\
\leq& Pr\left[|X_C - \mu_C| \geq r\sqrt{\mu_C}\right] \\
\leq& 1/r^2
\end{aligned}
$$