Assignment 2

Advanced Algorithms and Datastructures

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

May 22, 2014

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

2.2 Exercise 3(a)

We would store the buttom-k samples in a minimum heap structure H, sorted by their hashing value. This way we can insert new entries in $O(\lg\ n)$, and retrieve the $S_h^k(H)$ lowest hash values in $O(k\ \lg\ n)$ where n is the total number of input values.

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

2.5 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 smalles 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, thus proving the equality.

2.6 Exercise 4(b)

2.7 Exercise 4(c)

3 Bottom-k sampling with strong universality

3.1 Exercise 5

3.2 Exercise 6

3.3 Exercise 7

References