CS 270: Combinatorial Algorithms and Data Structures — Spring 2023

Lecture 12 — February 23, 2023

Prof. Jelani Nelson — *Scribe: Matthew Ding, Jonathan Tay*

1 Overview

In the last lecture we introduced hashing with linear probing, and proved that it achieves constant expected query time with a fully-random hash function.

In this lecture we show linear probing with a k-wise independent hash function also achieves constant expected query time for $k = 7$ and $k = 5$. The proof comes from Pagh, Pagh, and Ružić [1], and utilizes the "symmetrization trick". We also briefly introduce the approximate membership and dictionary problems.

2 Linear Probing with k-wise Independent Hashing

2.1 7-wise Independent Hashing

We first assume that the length of the hash table, $m = 2n$. In the previous lecture we showed that

$$\mathbb{E}[\#\text{ probes to query}(z)] \le \sum_{i=1}^{\infty} k \cdot \mathbb{P}(\text{a specific length } k \text{ interval containing } h(z) \text{ is full}) \tag{1}$$

Note that the fullness of an interval is about the actual location that z is stored, as opposed to $h(z)$ which is about the location that the key hashes to.

We define E_k to be the indicator random variable

$$E_k = \begin{cases} 1 & \text{if } z \text{ is contained in a full interval of length} \ge k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$\begin{aligned}
\sum_{k=1}^{\infty} E_k &= \sum_{k=1}^{\infty} P(E_k = 1) && (E_k \text{ is an indicator variable}) \\
&= \sum_{k=1}^{\infty} \mathbb{P}\left(\vee_{i=1}^{k} \text{ the } i^{\text{th}}\ k\text{-interval containing } h(z) \text{ is full}\right) \\
&\le \sum_{k=1}^{\infty} k \cdot \mathbb{P}(\text{a } k\text{-interval containing } h(z) \text{ is full}) && (\text{symmetry between intervals}) \\
&\le \sum_{k} k \cdot e^{-\Omega(k)} && (\text{by the Chernoff bound}) \\
&= O(1)
\end{aligned}$$

Problem: We need the use of fully random variables to use the Chernoff bound in the last step.

Solution: Use 7-wise independent hashing, and bound $\mathbb{P}(\text{a } k\text{-interval containing } h(z) \text{ is full}) = O\left(\frac{1}{k^3}\right)$

Let I denote the specific (arbitrary) interval in Eq. (1). Define the indicator random variable X_i to be

$$X_i = \begin{cases} 1 & \text{if } h(i\text{th key}) \in I \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Additionally we define the load on interval I as $L(I) = \sum_{i=1}^{n} X_i$. Note that $\mathbb{E}[L(I)] = \frac{k}{2}$.

Definition 2.1 (Full Interval)**.** An interval I is considered full if $|\{x : h(x) \in I\}| \geq |I|$.

We now seek to bound that probability that a length k interval is full.

$$\mathbb{P}\left(|L(I) - \mathbb{E}[L(I)] > \frac{k}{2}\right) < \left(\frac{k}{2}\right)^{-6} \cdot \mathbb{E}[\ L(I) - \mathbb{E}[L(I)]\ ] \qquad \text{(by Theorem 2.2)} \tag{4}$$

$$= \left(\frac{k}{2}\right)^{-6} \cdot \mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - \frac{k}{2}\right)^6\right] \tag{5}$$

To deal with the final term, it is doable with combinatorics by expanding out the powers, but it is better dealt with a probability trick of "symmetrization".

Probability and Vector Detour The l_p norm of a vector is defined as: $||x||_p := (\sum |x_i|^p)^{1/p}$. For random variables, we can also define the l_p norm as $\|X\|_p := (\mathbb{E}[|X|^p])^{1/p}$.

We will also use the following inequalities without proof:

Theorem 2.2 (Extended Markov Inequality)**.** *For a nonnegative random variable X, we have*

$$\mathbb{P}(|X| > a) \leq \frac{E[|X|^n]}{a^n} \tag{6}$$

Theorem 2.3 (Minkowski's Inequality)**.** *L^p spaces are normed vector spaces. Therefore, for $p \geq 1$, we have the triangle inequality*

$$\|x + y\|_p \leq \|x\|_p + \|y\|_p \tag{7}$$

Theorem 2.4 (Jensen's Inequality)**.** *If f is a convex function and X is a random variable, then*

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)] \tag{8}$$

Lastly, we also note that we are able to define a new random variable Y that is drawn from the same distribution as X but independent from X. This allows us to do the following:

$$\mathbb{E}[|X - \mu|] = \mathbb{E}[|X - \mathbb{E}[Y]|] = \mathbb{E}_X[|\mathbb{E}_Y[X - Y]|] = \mathbb{E}[|X - Y|] \tag{9}$$

Now, back to simplifying Equation (5). To make things easier, we deal with the expectation term without the exponent first. We also define another random variable X' drawn from the same distribution as X but independently selected; and define $\sigma_i \in \{-1, 1\}$ to be an uniform independent random variable.

$$\begin{aligned}
\left\|\sum_{i=1}^{n} X_i - \mathbb{E}[X_i]\right\|_6 &= \left\|\mathbb{E}_{x_i}\left[\sum X_i - \sum X_i'\right]\right\|_6 && \text{(from Equation 9)} \\
&\leq \left\|\sum (X_i - X_i')\right\|_6 && \text{(by Theorem 2.4)} \\
&\leq \left\|\sum (\sigma_i (X_i - X_i'))\right\|_6 && \text{(by symmetry)} \\
&\leq 2\left\|\sum (\sigma_i \cdot X_i)\right\|_6 && \text{(by Theorem 2.3)}
\end{aligned}$$

We complete that last step, by setting $X_i - X_i'$ to be another random variable Z which is valid, because these two variables are independently chosen. Its symmetry can be observed as $\mathbb{P}(Z_i) = \alpha$ is the same as $\mathbb{P}(Z_i) = -\alpha$, $\forall \alpha \in \mathbb{R}$.

Now if we include back the expectation and the exponent,

$$\begin{aligned}
\mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - \frac{k}{2}\right)^6\right] &= \mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - \mathbb{E}[X_i]\right)^6\right] \\
&= \mathbb{E}\left[\sum \sigma_i X_i\right]^6 && \text{(Eliminate constant term)} \\
&= \sum_{i_1, i_2 \dots i_6} \left(\mathbb{E}[X_{i1} \dots X_{i6}] \cdot \mathbb{E}[\sigma_{i1} \dots \sigma_{i6}]\right) && \text{(Expansion of exponent)}
\end{aligned}$$

We note that

$$\mathbb{E}[\sigma_i^j] = \begin{cases} 1, & \text{if } j \text{ is odd} \\ 0, & \text{if } j \text{ is even} \end{cases}$$

because it is a random uniform variable with magnitude 1, $\{-1, 1\}$ squared will always give 1, while the expectation of a single variable would be 0. This allows us to remove all terms of the expanded exponentiation with odd powers of σ. The remaining terms would remain only if the contributing 6 terms are in the form $\{(2, 2, 2), (4, 2), (6)\}$. Additionally, we note that we can treat any $X_{i1} \dots X_{i6}$ as independent, given that we have a 7-wise hash functions, so keys i_1, i_2, $\cdots i_6$, and z are effectively hashed to random independent locations.

To take the first possibility as an example $\{2, 2, 2\}$, it means that there are 3 distinct locations, and out of 6 terms, 3 sets of 2 terms have hashed to the same location as each other (but distinct from the other sets). The expectation of this happening for 3 specific locations is $\left(\frac{k}{m}\right)^3$, and the number of ways to choose these random variables is n^3. This gives $O\left(n^3 \cdot \left(\frac{k}{m}\right)^3\right)$. Generalizing this for the set $\{2, 4\}$ we get $O\left(n^2 \cdot \left(\frac{k}{m}\right)^2\right)$, and for the set $\{6\}$ we get $O\left(n \cdot \frac{k}{m}\right)$.

$$\sum_{i_1,i_2\ldots i_6} (\mathbb{E}[X_{i1}\ldots X_{i6}]\cdot\mathbb{E}[\sigma_{i1}\ldots\sigma_{i6}]) = O\left(n^3\cdot\left(\frac{k}{m}\right)^3\right) + O\left(n^2\cdot\left(\frac{k}{m}\right)^2\right) + O\left(n\cdot\frac{k}{m}\right)$$
$$= O\left(n^3\cdot\left(\frac{k}{m}\right)^3\right)$$
$$= O(k^3) \qquad (m = 2n)$$

Using this result, we get that

$$\left(\frac{k}{2}\right)^{-6}\cdot\mathbb{E}\left[\left(\sum_{i=1}^{n} X_i - \mathbb{E}[X_i]\right)^6\right] = \left(\frac{k}{2}\right)^{-6}\cdot O(k^3)$$
$$= O\left(\frac{1}{k^3}\right)$$

Plugging back into Eq. (1), we have

$$\mathbb{E}[\#\text{ probes to query}(z)] \le \sum_{i=1}^{\infty} k\cdot\mathbb{P}(\text{a specific length k interval containing h(x) is full})$$
$$= \sum_{i=1}^{\infty} k\cdot O\left(\frac{1}{k^3}\right)$$
$$= O(1)$$

giving us expected constant query time with a 7-wise independent hash function, as desired.

2.2 5-wise Independent Hashing

We first note that 5-independent hash functions are optimal, as it was shown by Pătraşcu and Thorup [2] that there exist random 3 and 4-independent hash functions with expected logarithmic search time for specific keys.

We now sketch the proof (by picture) that a 5-wise independent hash function is still sufficient for expected constant query time. Our goal is to construct a constant number of intervals where if I is full, at least one of these intervals is "almost full".

1. Construct a perfectly balanced binary search tree with the leaves corresponding to the entries of our array. Round up k to nearest power of 2 and consider the union of all arbitrary length k intervals that cover $h(z)$ (colored yellow in Fig. 1).

2. Go to the level of the tree that where every node has k leaves (marked in green)

3. Go 2 levels lower, where each node has $k/4$ leaves (marked in pink). The total number of pink nodes that intersect all possible k length intervals containing $h(z)$ is $O(1)$ pink nodes, as each pink node has $O(k/4)$ leaves and the yellow interval is at most $O(2k)$. In particular, at most 5 pink intervals intersect I.

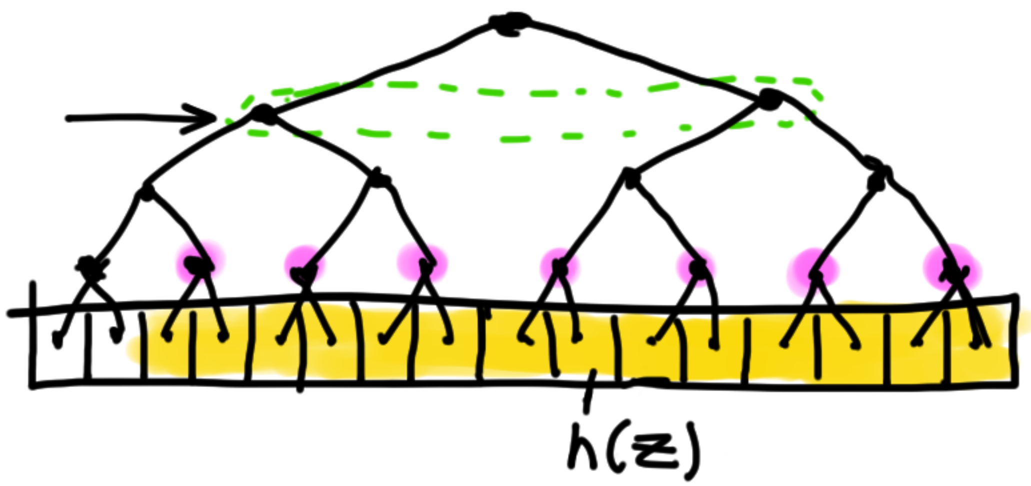

Figure 1: Diagram of 5-wise independent hash function proof, $m = 16$, $k = 8$

To reuse the equation used for the proof of 7-wise independent hashing

$$
\begin{aligned}
\mathbb{E}[\text{Runtime}] &\leq \sum_{k=1}^{\infty} E_k \\
&= \sum_{k=1}^{\infty} P(E_k = 1) && (E_k \text{ is an indicator variable}) \\
&= \sum_{k=1}^{\infty} \mathbb{P}\left(\exists \text{ a full length } k\text{-interval containing } h(z)\right) \\
&= \sum_{k=1}^{\infty} \mathbb{P}\left(\exists \text{ a pink interval that is almost full}\right) \\
&\leq \sum_{k=1}^{\infty} O(1) \cdot P(\exists \text{ a particular pink interval almost full}) && (\text{symmetry between intervals})
\end{aligned}
$$

Claim 2.5. If I is full, at least one pink interval must be at least 3/5 full.

Proof. The length k interval T is fully contained in the union of ≤ 5 pink nodes; call them $b_1, .., b_5$. If T is full, then k elements hash to T, which by pigeonhole means one of $b_1, ..., b_5$ must have at least $k/5$ elements hashed to it. But that b_i is an interval of length $k/4$, and $\frac{k/5}{k/4} = 80\%$, and thus it is at least 80% full, and thus greater than 3/5 full.

This assumes though that k is a power of 2. In reality we pick the pink nodes by rounding up to the nearest power of 2, then going down 2 levels which corresponds to dividing by 4 (so if we care about $k = 14$, we would have pink nodes covering intervals of length 4). If the pink node interval length is t, then basically we know $1/4 \leq t/k < 1/2$ (close to 1/2 if k is 1 more than a power of 2).

The "worst case" is t/k is close to $1/2$, in which case we really have b_i's being intervals of length $k/2$ and T covered by ≤ 3 pink nodes, so the pigeonhole argument gives $\frac{k/3}{k/2} = 66.7\% > 3/5$. $\square$

Now we see that the intervals represented by pink nodes satisfy our desired property. From this, we see that we can actually modify Eq. (1) as

$$\mathbb{E}[\#\text{ probes to query}(z)] \leq \sum_{i=1}^{\infty} \mathbb{P}(\text{there exists a pink interval that is "almost" full}) \tag{10}$$

Using the symmetrization trick with a 5-wise independent hash function instead of 7-wise, we can bound this as

$$\mathbb{P}(\text{there exists a pink interval that is "almost" full}) \leq \sum_{i=1}^{\infty} O(1) \cdot O(\frac{1}{k^2}) = O(1) \tag{11}$$

giving us expected constant query time with a 5-wise independent hash function, as desired.

3 Approximate Membership and Dictionary

Solutions to the dictionary problem typically take $O(nW)$ bits, where W is the word size. For data structures with smaller space complexity, we settle for *approximate* solutions.

3.1 Approximate Membership Problem

The following is the approximate membership problem: Store a database of keys subject to

1. insert(x): adds x to the database
2. query(x): returns whether x is in the database. If x is actually present, it returns YES with probability 1. If x is not actually present, it returns NO with probability $\geq 1 - \epsilon$.

We wish to solve this problem with $O(n \log \frac{1}{\epsilon})$. In the next lecture, we will see how to do this with Bloom filters.

3.2 Approximate Dictionary Problem

Approximate dictionary has a very similar setup as approximate membership. Assuming that a key is not in the database, querying it outputs an arbitrary value with a probability $\leq \epsilon$. If the key actually is in the database, it will return the correct value with probability 1. In the next lecture, we will see how to implement this data structure using bloomier filters and cuckoo hashing.

References

[1] Anna Pagh, Rasmus Pagh, Milan Ružić. Linear Probing with 5-wise Independence. *SIAM Review*, 53(3):547–558, 2011.

[2] Mihai Pătraşcu, Mikkel Thorup. On the k-Independence Required by Linear Probing and Minwise Independence. ACM Transactions on Algorithms (TALG) 12(1): 1-27, 2015.