COMS 4995-8: Advanced Algorithms (Spring'21) Jan 19, 2021

Lecture 3: Hashing

Instructor: *Alex Andoni* Scribes: *Conor Sweeney (cjs2201), Erica Wei(cw3137)*

1 Course Info

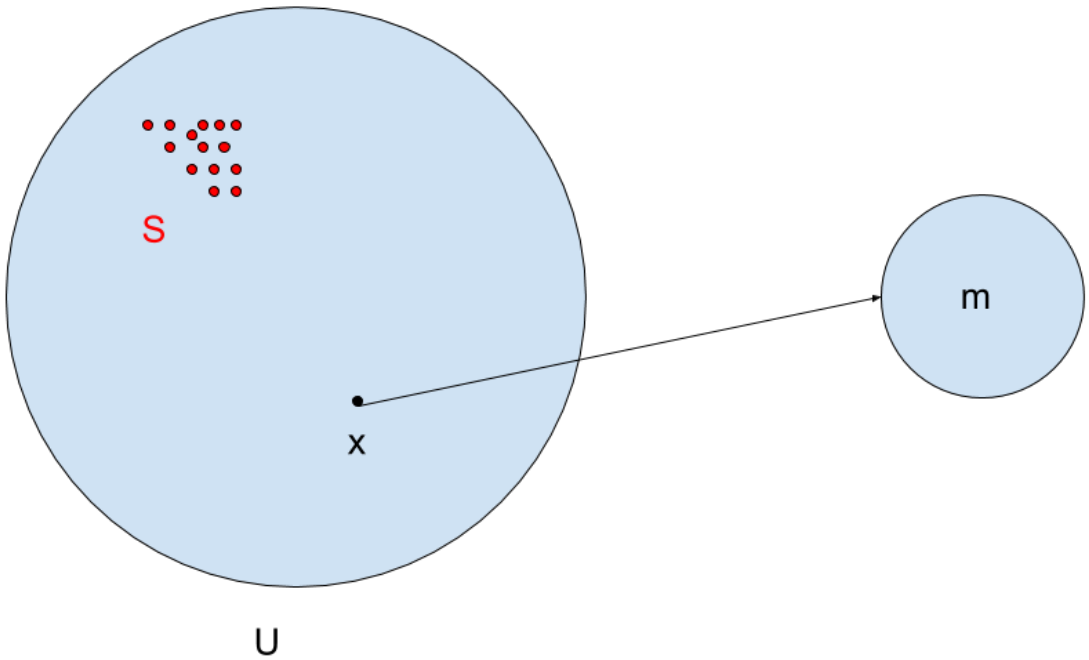
- Homework 1 has been posted (in courseworks under Files). It is due next Thursday Jan 28 at 4pm.
- OH Calendar has been posted as well (see courseworks/course info for link).
- Please sign up for to scribe (see courseworks announcement).

2 Class Topics

- Dictionary & Hashing

3 Problem: Dictionary

In this problem, a dictionary, a data structure problem. We want to pre-process a subset S $\in$ U, U is a large universe, into the dictionary so that it can quickly answer if "Is x $\in$ S?" Solution is to use hashing.



Above is a visualization of how a dictionary maps a large universe U to a smaller easily search-able dictionary m.

3.1 Collision

When $x \neq y$, but hash function maps them into the same cell s.t $h(x) = h(y)$.

3.2 Ideal

Enough for hash function h such that $\forall y \in S, \forall x \in U, x \neq y \rightarrow h(x) \neq h(y)$.

3.3 Issue

It is very hard to construct such a hash function because it depends on S a lot. And it is hard to compute the hash function. Here we will construct a hash function h s.t we have a better trade-off between computation/evaluation time and how "good" it is for distribution in terms of collisions.

4 Solutions

4.1 Random Function

Randomly choose hash function h and allow some collisions.

$H = \{\text{all functions } h : U \to \lfloor m \rfloor\}$

$|H| = m^{|u|}$

$h \in H$ is chosen at random

4.1.1 So what is the size of each bucket?

Let C_x := the collision count for x then:

$C_x \triangleq$ the number of elements in S s.t. $h(x) = h(y)$

Then the expectation for collisions can be considered as:

$$\begin{aligned}
\mathbb{E}_n[\text{size of the bucket}] &= \mathbb{E}[C_x] \\
&= \mathbb{E}_n[\sum_{i \in S} \mathbb{1}[h(i) = h(x)]] \\
&= \sum_{i \in S} \mathbb{E}_n[\mathbb{1}[h(i) = h(x)]] \\
&= |s| \cdot \frac{1}{m} \\
&= \frac{n}{m}
\end{aligned} \tag{1}$$

It is OK to set $m = \Theta(n)$ (e.g. m = n) which would mean that $\mathbb{E}_n$[size of the buckets] = 1.

Query Time = $O(1)$ + time to compute h (in expectations)

4.1.2 How large is the biggest bucket?

$\Theta(\frac{log(n)}{log(log(n))})$ with probability at least 99%.

4.1.3 How do we choose/store the random hash function h?

It is actually okay to use a hash function $h \in H$ with "less" randomness. See section 5.3 for more details.

4.2 Knuth's Solution

Knuth suggests a concrete hash function to use:

$$h(x) = \lfloor \{ \frac{\sqrt{5} - 1}{2} x \} * m \rfloor \tag{2}$$

m is the integer where U maps to [m]. This is not a good hash function because it's deterministic and in some cases it will have a large number of collisions.

4.3 Less Random

Definition: H is $\alpha\cdot$ almost -universal if

$$\forall x \neq y, Pr_h(h(x) = h(y)) \leq \alpha/m \tag{3}$$

In this case we relax from randomness and the probability of collisions is upper bounded.

Claim 1. $\mathbb{E}[\textit{size of the bucket}] \leq \alpha \cdot n/m.$

The proof is similar as above.

Example [Dietzfelbinger et al., '97] :

$$\begin{aligned} \forall a \in [U] \text{ a is randomly chosen with odd number,} \\ h_a(x) = \lfloor (a \cdot x)\%|U| \cdot \frac{m}{|U|} \rfloor \\ H = \{h_a, a \in [U]odd\} \end{aligned} \tag{4}$$

Fact: H is 2-almost universal.

Lemma: Dictionary problem can be solved by using O(n) space and O(1) expected query time.

Proof. Set m = n, table takes space O(m+n)= O(n), E[size of bucket] $\leq 1 + \frac{n}{m} = 2$. Hash function description is $O(lg|U|)$. □

4.4 Perfect Hashing

Goal O(1) run time deterministically.

Ideally Let $C \triangleq \sum_{x\in S} C_x$, we want C = 0 $\Rightarrow$ size of bucket ≤ 1.

$$\begin{aligned} \mathbb{E}[c] &= \mathbb{E}_h[\sum_{x\in S} C_x] \\ &= \sum_{x\in S} \mathbb{E}_h[C_x] \\ &= \sum_{x\in S} n/m \\ &= \frac{n^2}{m} \end{aligned} \tag{5}$$

Suppose set m = $4n^2$, then $\mathbb{E}[c] = 1/4$.

By Markov Bound:

$$Pr[C \geq 4\mathbb{E}[C]] \leq \frac{\mathbb{E}[C]}{4\mathbb{E}[C]} = \frac{1}{4} \tag{6}$$

With probability at least 3/4, we have $C < 4\mathbb{E}[C] = 1$. Since C is an integer, here we can say C = 0, no collisions.

Corollary: can solve Dictionary problem with $O(n^2)$ space and O(1) query time.

Algorithm is following:
1) Set m = $4n^2$
2) Build a hash table using a random hash function $h \in H$.
3) Compute C.
4) If C $\geq$ 1, then try again.
This will eventually finish because the probability you have to try again is 1/4. The idea here is the size of table is large enough so there's not many collisions.
Issue: How many times we need to try again?

$$\begin{aligned} \mathbb{E}[\#\text{ of tries in pre-processing algorithm}] &= 1 \cdot 1 + 1/4 + (1/4)^2 + (1/4)^3 + ... \\ \text{Another way to think about it is } &= 1 \cdot 3/4 + 1/4 \cdot (3/4) * 2 + (1/4)^2 \cdot (3/4) * 3 + ... \\ &= \frac{1}{1 - 1/4} = 4/3 \end{aligned} \tag{7}$$

Expectation to try is 4/3 times.