CSE 347 Analysis of Algorithms March 25, 2019

Lecture 17 Summary

Instructor: Brendan Juba

1. The promise of obtaining dictionaries with $O(1)$ insert, lookup, and delete operations using hashing seems too good to be true in a worst case model
   (a) By the Pigeonhole principle, if the universe has size $N \geq km$, then any hash function for a table of size m has some bucket with at least k collisions. If all of our elements are from these k, then we are not getting $O(1)$ time operations.
   (b) We saw that when we insert up to $O(\sqrt{m\delta})$ *uniform random* elements into the table using $h(x) = x \bmod m$, we had no collisions with probability $1 - \delta$, but we can't replicate our strategy for treaps, i.e., assigning a random key to each element: we need something like a dictionary data structure to remember which keys we assigned to which elements.
2. The key observation is that it's enough to control the probability that pairs of elements collide, which is obtained by *universal hash functions*
   (a) Precisely, we now consider a *family* of hash functions $\mathcal{H}$, where we choose an actual hash function h (uniformly) at random from $\mathcal{H}$. So now, if H denotes our random member of $\mathcal{H}$, for each x $H(x)$ is a random element of $\{0, \ldots, m-1\}$. (We only need to remember the choice of $h \in \mathcal{H}$, not a distinct $H(x)$ for each x.)
   (b) We say that this random H is a *universal hash function* if for every pair of distinct x and y, $\Pr[H(x) = H(y)] = 1/m$.
   (c) If we insert up to $\sqrt{2m\delta}$ elements into a hash table using a universal hash function, there are no collisions with probability $1 - \delta$.
      i. We first write the expected number of collisions in terms of indicator random variables for collisions between a fixed pair; since the hash function is universal, the pair collides with probability $1/m$, so in expectation we have at most δ colliding pairs.
      ii. Now we use *Markov's inequality*: for any nonnegative random variable, the probability it is more than k times its expected value is less than $1/k$.
      iii. That is: in order for the number of colliding pairs to be at least 1, this quantity must be more than $1/\delta$ times its expected value. This occurs with probability at most δ.
   (d) Furthermore, if we insert up to $\sqrt{Cm\delta}$ elements, each x collides with at most C other elements with probability $1-\delta$, so as long as we're inserting any $O(\sqrt{m})$ elements, there are still $O(1)$ collisions per bucket, and we have $O(1)$-time operations with probability $1 - \delta$.
      i. We just observe that there are, in expectation, $\sqrt{C\delta/m}$ elements that collide with any fixed x.
      ii. In order to exceed C, this must be more than $\sqrt{Cm/\delta}$ times its expected value, which by Markov's inequality occurs with probability at most $\sqrt{\delta/Cm}$.

iii. Now we just take a union bound over the $\sqrt{Cm\delta}$ elements: in total, the probability that any of them collides with more than C elements is at most δ, as needed.

3. The standard linear hash functions $h_{\mathbf{a}}(x) = \sum_{i=1}^{\ell} a_i x_i \bmod m$ for m prime ($\mathbf{a}$ chosen uniformly at random from $\{0, 1, \ldots, m-1\}^{\ell}$) give a universal hash function. (I.e., we have broken x into ℓ blocks of bits of size $\lfloor \log_2 m \rfloor$ each, which we interpret as numbers in the range $\{0, \ldots, m-1\}$.)

   (a) For any distinct pair x and y, there is some ith index where they differ.

   (b) Letting $r = \sum_{j \neq i} a_j x_j \bmod m$ and $s = \sum_{j \neq i} a_j y_j \bmod m$, there is a collision if and only if $a_i x_i + r = a_i y_i + s \bmod m$. There is a unique choice of a_i satisfying this equation.

   (c) Since, for any choice of a_j, $j \neq i$, a_i is an independent and uniform-random element of $\{0, 1, \ldots, m-1\}$, therefore, $h_{\mathbf{a}}(x) = h_{\mathbf{a}}(y)$ with probability exactly $1/m$ as needed.

   (d) Note that if we add a constant shift b, i.e., extend our hash function to $h_{\mathbf{a},b}(x) = \sum_{i=1}^{\ell} a_i x_i + b \bmod m$, then we have the stronger property that for any choice of x, y, w, z, $\Pr[h_{\mathbf{a},b}(x) = w \text{ and } h_{\mathbf{a},b}(y) = z] = 1/m^2$, since there is similarly a unique choice of the *pair* a_i and b that satisfies this system of two linear equalities. We say that this family of hash functions is *pairwise independent*, and this stronger property is sometimes useful for algorithms.

4. Next time: we will start studying randomized algorithms that relax their *correctness* guarantee, i.e., they only return the *correct answer* with high probability (which we can control).

   (a) Specifically, we will see an algorithm for the Global Min-Cut problem: given an unweighted, undirected graph G, find a cut (A, B) (both nonempty) that minimizes the number of edges crossing.

   (b) Note that the Global Min-Cut problem is distinct from the Minimum s-t Cut problem

      i. The graph is undirected and unweighted, whereas the Minimum s-t Cut problem uses a weighted, directed graph

      ii. The value of the cut is just the number of edges, not the total weight of edges crossing *from A to B*.

      iii. We are not required to return a cut with $s \in A$ and $t \in B$.

   (c) Nevertheless, we can solve the Global Min-Cut problem by creating directed edges of weight 1 in both directions, enumerating over all pairs of vertices, and taking the best s-t cut. We will see that it is possible to design a much simpler *randomized* algorithm for Global Min-Cut.