Universal Minimum Perfect Hash Functions

Jacob Denson

October 13, 2014

A hash is a function from some set to an interval in the natural numbers $[0, N]$. We typically call elements of domain the keys of the hash. The main application of hashes is if we have an array of values corresponding to each key, such that the position of the value corresponds to the mapped integer of the hash function, we can lookup elements in the array in the time it takes to compute the hash function, hopefully in constant time.

If two keys map to the same value, we call this a collision. This either causes obvious memory problems in an array problem, or lookup time to increase in order to fix these collisions. We say a hash function is perfect if it is injective – in the context of array mapping, this means the array will be collision free. A hash function is minimal if it is surjective. This means arrays will be as space efficient as possible, as every position in the array will correspond to some key. It is of course desirable for a hash function to be both minimal and perfect. In fact, this is the best a hash function could be given that every key in the domain will be used.

This leads us to the problem of this document. Given any set of keys, construct a perfect, minimal constant time lookup hash function using linear memory in the number of keys.

The solution that has been developed needs a family of hash functions, and a compact static-dictionary data structure with contant time lookup. The family of hash functions should run in constant time, and map indices uniformly in the range. We need a large enough family so that some subcollection will work with the keys given.

The algorithm to do this was invented by Michael Fredman, János Komlós and Endre Szemerédi, and is named after them: FKS. It takes a set of N keys and returns a constant time lookup structure. The trick is to hash the keys into m groups $M_1, M_2, \ldots, M_m$ such that $\sum_{k=1}^{m} |M_k|^2 < 3m$, and then to hash each of group M_k perfectly into $[0, |M_k|^2]$. The space required for this is linear by the requirement that $\sum_{k=1}^{m} |M_k|^2 < 3m$. Finally, we number the elements based on their position in the mapped array. We construct this in time asymptotic to $\Theta(N^3 lgM)$, where M is the range mapped to. The problem with this algorithm is M can become very large, so we have an additional algorithm which reduces M to a constant factor of N.

This additional algorithm, designed by Fabiano C. Botelho, Djamal Belazzougui, Rasmus Pagh, and Nivio Ziviani, is known as BDZ. BDZ chooses

$M^* \geq 1.23N$, and 3 hashes mapped into $[0, M^*]$ very carefully and cleverly. We then store the function in a way we can lookup very quickly.

I will expand this more when I have time.