CSC C78F 2000
Assignment 3

due: Wednesday, November 8, 2000

1. Let $|U| = N$ and $m \geq 2$. Let $\mathcal{H} = \{h_1, h_2, \ldots, h_t\}$ be a 1-universal family of hash-functions $h_i : U \to \{0, 1, \ldots, m-1\}$.

   (a) Prove that for every $1 \leq i \leq t$ there is a set $U_i \subset U$ of size $|U_i| \geq N/m^i$ with the property that the functions $h_1, h_2, \ldots, h_i$ are constant on U_i.

   (b) Prove that $|\mathcal{H}| \geq m(\lceil \log_m N \rceil - 1)$.

   (c) Assume that m is a prime and let $U = \{(x, y) : 0 \leq x, y < m\}$. For every $a \in U$ we define the hash function h_a by $h_a(x, y) = (a_1 x + a_2 y \bmod m)$. Find a 1-universal family of hash functions $\mathcal{H} \subset \{h_a : a \in U\}$ for which the bound in (b) holds with equality. You must prove that this family has the desired properties.

2. Recall that a hash function is perfect, for a given set of n keys, if it hashes the keys without collisions.

   (a) Show that if we hash to a table of size $m > n(n-1)$ then a randomly chosen hash function from a universal family of hash functions is perfect (for a given set of n keys) with probability $> 1/2$.

   For the remaining parts of this problem we let $U = \{0, 1\}^6$ and for every binary 3×6 matrix B let $h_B : U \to \{0, 1\}^3$ be the linear transformation given by $h_B(x) = Bx$. We consider the family of hash functions $\mathcal{H} = \{h_B : B \text{ is a binary 3 x 6 matrix }\}$.

   (b) Give a fast randomized method for finding a hash function in $\mathcal{H}$ that perfectly hashes the set $S = \{010100, 110110, 100101\}$. Use this method to find a perfect hash function.

   (c) Can we always find a perfect hash function from $\mathcal{H}$ for a given set of 3 keys?

   (d) Find a perfect hash function from $\mathcal{H}$ for

   $$S = \{x \in U : 0 = x_1 + x_2 = x_3 + x_4 + x_5 + 1 = x_6\}.$$

   (e) Is there a perfect hash function from $\mathcal{H}$ for every set S of size 8? Explain your answer.

3. Recall the ADT CACHE, discussed in Assignment 1, which consists of a subset $C \subseteq \{1, \ldots, m\}$ of size at most k and one operation *access(i)*, which adds i to the set C and, if this causes the size of C to become bigger than k, removes the element from C that was least recently the parameter of an access operation. Initially, $C = \emptyset$.

   Another ADT, MAGICCACHE is the same as CACHE, except that its set will be called C' and, if the size of C' becomes bigger than k, MAGICCACHE removes the element from C' that will next be accessed furthest in the future.

   For example, if $k = 2$ and the sequence

   $$access(1),\ access(2),\ access(3),\ access(1)$$

   is performed, then CACHE removes element 1 from C during the third access and removes element 2 from C during the fourth access, whereas MAGICCACHE removes element 2 from C' during the third access and does not have to remove any element from C' during the fourth access.

   Although MAGICCACHE has magical knowledge of what accesses will be requested in the future (and behaves accordingly), this question shows that CACHE, which does not have this knowledge, removes at most a factor of k more elements than MAGICCACHE when processing any sequence of accesses.

   For any sequence of accesses σ, let $R(\sigma)$ denote the number of elements removed from C by CACHE and let $R'(\sigma)$ denote the number of elements removed from C' by MAGICCACHE. Note: elements that are removed more than once are counted each time they are removed.

(a) Let σ be a sequence of accesses during which CACHE removes at least one element. Prove that MAGICCACHE removes at least one element during σ.

(b) Let $\rho\sigma\tau$ be a sequence of accesses such that, during ρ, CACHE removes at least 1 element and, during σ, CACHE removes exactly k elements. Prove that MAGICCACHE removes at least 1 element during σ.

(c) Prove that, for any sequence σ, $R(\sigma) \leq k \cdot R'(\sigma)$.

(d) For any positive integer n, construct a sequence of accesses σ for which $R(\sigma) = kn$ and $R'(\sigma) = n$. Justify why your sequence has the required properties.

4. Consider the following data structure for representing a set. The elements of the set are stored in a singly linked list of sorted arrays, where the number of elements in each array is a power of 2 and the sizes of all the arrays in the list are different. Each element in the set occurs in exactly one array. The arrays in the linked list are kept in order of increasing size.

   To insert a new element x into the set (given the precondition that x is not already in the set),

```
create a new array of size 1 containing x
insert this new array at the beginning of the linked list
while the linked list contains 2 arrays of the same size
   merge the 2 arrays into one array of twice the size
```

   (a) Draw the data structure that results after inserting the following sequence of elements into an initially empty set:

   $$2, 63, 1, 32, 77, 8$$

   (b) What is the worst case time for performing SEARCH, when the set has size n? Briefly justify your answer.

   (c) What is the worst case time for performing INSERT, when the set has size n? Briefly justify your answer.

   (d) Use the aggregate method to prove that the amortized insertion time in a sequence of n insertions, starting with an initially empty set, is $O(\log n)$.

   (e) Use the accounting method to prove that the amortized insertion time in a sequence of n insertions, starting with an initially empty set, is $O(\log n)$.