Data Structures PA3

Task 5 write-up

To implement my cache I used a hash table with linear probing.

I used linear probing because it ensures that the value, if it exists is in close proximity in the array to the initial hashed index, and since arrays are moved to memory-cache in chunks based on proximity (Data prefetching/cache prefetching). Neither double hashing nor linked list hashing can use this feature of data prefetching to their advantage.

Each node contains a key, a value and a frequency count. The frequency count keeps track of the number of times this item has been accessed from cache (very time an item is successful looked up in cache the item's frequency is incremented by one).

I used they hash function that gave the best performance, polynomial hash with a=41, and div compression.

Every time an item is not found in cache, I search it in the dictionary file and simultaneously insert that word in to cache. The insert in cache works like this:

If the location in cache is already occupied, it checks the frequency of access

If frequency of access of greater than one, then it moves forward linearly

However, since the load factor of the cache is not maintained below 0.5, I ensure O(1) in lookup and insert by only searching/inserting up to a threshold value in the cache. For example if the threshold is 3, then I only insert or lookup in cache to within 3 items from the original hash value. If this was not ensured, my cache performance would degrade to O(size of cache)/O(1000).

The time taken by secret_1.txt using cache was 8603 seconds whereas without cache it took 10819 seconds.