COSC 242 Assignment Report

Patrick Murrow

September 11th, 2014

1 Section I

The main advantage of hash tables is speed. Hash tables use keys to find a position to insert the value. Searching uses the same mechanism as inserting, which means that the key that is being searched for will often take few iterations to find. Rather than searching through a list sequentially O(n) or even in a Binary Search Tree, Olog(n), hashtables can simply use the unique identifier to start the search exactly where it would have been inserted, if there were no collisions. A good hashing function means searching is relatively quick, potentially O(1). Linear probing starts at h(k) (or h(k%length)) If unoccupied, the object is inserted, otherwise insert at the next position h(k+1) then h(k+2), this strategy works well for small amounts of data that dont have very similar keys. But is prone to clustering, making searching slower as the key to be found will not be in a position close to its home key. Double hashing minimises clustering by changing index with a calculated step rather than a step of 1. The initial posistion is the same as linear probing but if there is a collision, then add (collision * g(1)) to that position, where g is some function such as: 1 + k%(length 1). This means that if two keys had the same start position initally, that there is a high chance that the next position for each would be different. Because of this, there would be much lower clustering and faster searching.These strategies were implemented in our program by having enumerable types which allowed the method to be chosen. First trying to insert at the home cell, calling a step method in event of a collision that steps the index to insert by 1 for linear probing or by 1 + k%(length 1) for double hashing.The asgn.c file used a next_prime method which calculated the next prime to use for the size of the hashtable. The operation handling used flags which toggled the spell check, double hashing/linear probing and whether stats were printed. After selecting an option the program would read words from input and perform tasks on the words depending on which flags were set to TRUE (has value 1 but is more clear to use TRUE). When implementing the double hashing strategy we had issues figuring out when the table was full because we initially thought that if the key was trying to be inserted in to the same place as it's original home the hash table was full, however, it wasn't. We realised that if num_keys was equal to capacity then the hash table was full. Otherwise, most of the other code and implementation was sound.

2 Section II

I think our group worked really well. I think it was really helpful working in a team, when there was a bug, three eyes debugging was so much more efficient. Communicating to other group members was challenging and I can see how it would work really well in a firm, where programmers are consistently working beside one another. Effort: Sam - 30%, Daniel - 40%, Patrick - 30%. Usefulness: Sam - 40%, Daniel - 30%, Patrick - 30%.