Analysis of Hash Table Collision Strategies

TCSS 342: Data Structures
May 21st, 2015
Jesse Bannon

Summary

When rewriting my Huffman Encoding algorithm to use English words rather than ASCII characters, a hash table was the only appropriate solution.

The assignment recommendation for dealing with hash collisions suggested using linear probing. The moment I read that I knew it would cause a performance hit. So instead I implemented quadratic probing my first try.

The difference in operations between the two functions are minimal. Quadratic probing has one additional multiplication operation over linear probing.

Below is the code that represents both functions. The highlighted piece is included in the quadratic probing function, **not** the linear probing function.

```
unsigned int get(t_node * table,
                 unsigned int key,
                 char * word)
{
    unsigned int idx, i = 0;
    while (1) {
        idx = (i*i++) + key % HASH_TABLE_SIZE;
        if (table[idx].key == NULL || strcmp(table[idx].key, word) == 0) {
            ++(stats.probes[i-1]);
            return idx;
        }
    }
}
```

Probing Analysis

In my analysis I used the recommended assignment text file WarAndPeace.txt, which is also included in the repository, as my test file.

Statistics that hold true for both probing tests:

```
WarAndPeace.txt size:        3291642 bits
Encoded Binary file size:    1384949 bits
Compression ratio:           0.4207

Number of Entries:     22691
Number of Buckets:     32768
Fill Percentage:       69.25%
```

Statistics that differ between probing tests:

	Linear Probing	Quadratic Probing
Run-time:	545 ms	541 ms
Max Probe:	328	25
Average Probe:	0.46	0.27

The performance impact of 4 milliseconds is not crucial. Though it is still obvious that quadratic probing is the better solution based on the max and average probes.

For further statistics, use the `-cs` argument when encoding any text file to see the stats displayed above in addition to a histogram of all probes.

The default get function uses quadratic probing, but I have included a commented version of linear probing within *HuffmanTree.c* you care to play around with it.