Kavan Lima

Computer Science I –Exercise
Hash Tables

4/8/25

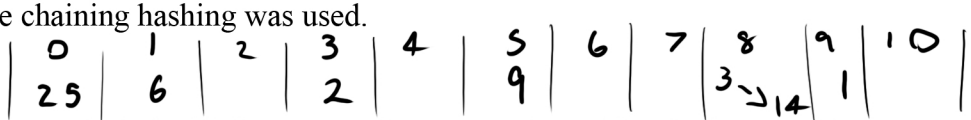
1) Consider a hash table that uses the linear probing technique with the following hash function f(x) = (5x+4)%11. (The hash table is of size 11.) If we insert the values 3, 9, 2, 1, 14, 6 and 25 into the table, in that order, show where these values would end up in the table?

index	0	1	2	3	4	5	6	7	8	9	10
value	25	6		2		9			3	1	14

2) Do the same question as above, but this time use the quadratic probing strategy.

index	0	1	2	3	4	5	6	7	8	9	10
value	25	14	6	2		9			3	1	

3) Do the question above, but draw a picture of what the hash table would look like if seperate chaining hashing was used.



4) Edit the code in htablelinear.c so that quadratic probing is the searching strategy used. You will need to modify insert function, then search and then delete. Add the code to your pdf when submitting.

```
void insertTable(struct htable *h, char word[]) {

    int hashval;
    hashval = hashvalue(word);

    // Here's the quadratic probing part.
    int originalHash = hashval;
    int i = 1;
    while (strcmp(h->entries[hashval], "") != 0)
    {
        hashval = (originalHash + (i*i))%TABLE_SIZE;
        i++;
    }

    strcpy(h->entries[hashval], word);
}
```

```
int searchTable(struct htable *h, char word[]) {

    int hashval;
    hashval = hashvalue(word);

    // See what comes first, the word or a blank spot.
    int originalHash = hashval;
    int i = 1;
    while (strcmp(h->entries[hashval], "") != 0 &&
        strcmp(h->entries[hashval], word) != 0)
    {
        hashval = (originalHash + (i*i))%TABLE_SIZE;
        i++;
    }

    // The word was in the table.
    if (strcmp(h->entries[hashval], word) == 0)
        return 1;

    // It wasn't.
    return 0;
}
```

```
void deleteTable(struct htable *h, char word[]) {

    int hashval;
    hashval = hashvalue(word);

    // See what comes first, the word or a blank spot
    int originalHash = hashval;
    int i = 1;
    while (strcmp(h->entries[hashval], "") != 0 &&
        strcmp(h->entries[hashval], word) != 0)
    {
        hashval = (originalHash + (i*i))%TABLE_SIZE;
        i++;
    }

    // Reset the word to be the empty string.
    if (strcmp(h->entries[hashval], word) == 0)
        strcpy(h->entries[hashval],"");

    // If we get here, the word wasn't in the table,
}
```