Applied Algorithms

Jaya Sandeep Ketha

December 2023

Q1.
Yes, I was able to insert all n words for each of (a) through (d). Text considered is 'Frog Prince' by Brothers Grimm, and n is considered as 1024.

Q2.
In (a), my maximum chain length is 6. 804 slots in the array were unused.

Q3.
In (b), 9.61 nodes had to visited for each insertion on average.
In (c), 2.36 nodes had to visited for each insertion on average.
In (d), 1.90 nodes had to visited for each insertion on average.

Q4.
In (b), I was able to insert 340 words.
In (c), I was able to insert 340 words.
In (d), I was able to insert 340 words.

Q5.
For (a), I was able to insert 4693 words.
For (b), I was able to insert 1024 words.
For (c), I was able to insert 1024 words.
For (d), I was able to insert 921 words.

Q6.
My maximum chain length is 31.

Words are:
these, while, might, below, given, sleep, wiped, least, aught, kinds, tales, boats, small, elbow, crumb, bless, steal, birth, pails, fewer, human, clung, crawl, boast, sands, gazer, roads, bloom, shady, prick, fires.

My minimum chain length is 0, as there are few slots unoccupied. If only occupied slots are considered, my minimum slot length is 1. Example: [sorrowful].

Q7.
In (b), 206.87 nodes had to visited for each insertion on average.
In (c), 18.64 nodes had to visited for each insertion on average.
In (d), 9.38 nodes had to visited for each insertion on average.

Q8.
In (a), there are 311 empty slots available.
In (b), there are 0 empty slots available.
In (c), there are 0 empty slots available.
In (d), there are 103 empty slots available.

Q9.
Each method (linear probing, quadratic probing, chaining) for handling collisions in hash tables has its own set of advantages and limitations. The appropriateness of each method depends on various factors such as the specific application requirements, the nature of data being stored, and the hash table imple-

mentation itself. Here's an overview of each method:

1. Linear Probing:

- **Appropriateness:** Linear probing is simple to implement and requires less memory overhead compared to chaining. It can be more cache-friendly as it tends to access contiguous memory locations.

- **Limitations:** It suffers from clustering, causing more collisions and degradation in performance when the table becomes densely populated. The clustering effect can lead to a degradation in performance as the table fills up.

2. Quadratic Probing:

- **Appropriateness:** It tries to address the clustering issue of linear probing by using a different probing sequence. It can distribute keys more uniformly across the hash table compared to linear probing.

- **Limitations:** Quadratic probing still encounters clustering, though it tends to distribute elements more evenly. However, it may suffer from secondary clustering issues.

3. Chaining (Separate Chaining):

- **Appropriateness:** Chaining provides a solution by using linked lists at each slot to handle collisions. It offers a more consistent time complexity for insertions and lookups even with a higher load factor.

- **Limitations:** It might have higher memory overhead due to the pointers or additional data structures needed to manage linked lists. It might also suffer from cache inefficiency if nodes in the linked lists are not contiguous in memory.

General:

- **Linear/Quadratic Probing:** Suitable for smaller datasets with a more predictable distribution of keys and when memory usage needs to be optimized.

- **Chaining:** Suitable for larger datasets, unknown or non-uniform key distributions, and when maintaining a consistent performance under higher load factors is crucial.

The appropriateness of each method depends on trade-offs between time complexity, space complexity, expected load factor, and the distribution of keys in the specific use case or application. Often, empirical testing and profiling are needed to determine the most suitable collision resolution method for a particular scenario.