Data Structures 2

Assignment 2

Implementing Perfect Hashing

Name	ID
Ranime Ahmed Elsayed Shehata.	21010531
Youssef Tarek Hussein Yousry.	21011595
Osama Hosny Hashem Elsayed.	21010238
Youssef Alaa Ahmed Haridy.	21011603
Mohamed Mohamed Mohamed Abdelmeneim Eldesouky.	21011213

Time & Space Complexity Analysis:

1. 0(N) Space Solution

Time Complexity:

- Best Case: **O(1)**

  No collisions in the first-level hash table.

- Average Case: **O(1)**

  Some collisions in the first-level hash table but they are evenly distributed across the bins.

- Worst Case: **O(n)**

  when there are many collisions in the first-level hash table, resulting in a large number of elements needing to be rehashed in the secondlevel hash tables.

Advantages:

- Low Collision Probability: By using universal hashing at the first level and rehashing with Method 1 at the second level, the probability of collisions is reduced, leading to better performance.

- Flexibility: The scheme allows for the use of different hash functions at both levels, providing flexibility in designing the hashing strategy.

Disadvantages:

The worst-case time complexity of O(N) can be a significant drawback if the hash function at the first level produces poor distribution of elements, leading to many collisions.

2. 0(N^2) Space Solution

Time Complexity:

- Best Case: **O(n)**

  when the first randomly chosen hash function results in no collisions.

- Average Case: **O(n)**

  when the first randomly chosen hash function results in some collisions, but a new hash function chosen randomly from the universal hash function family results in minimal collisions.

- Worst Case: **O(n^2)**

  when every randomly chosen hash function results in a significant number of collisions, requiring many attempts to find a suitable hash function.

Advantages:

- Adaptability: The method is flexible and can adapt to different sizes of dictionaries by adjusting the size of the hash table based on the square of the dictionary size.
- Space efficiency: This method allows for the creation of a hash table with a size that is quadratic in the size of the dictionary.

Disadvantages:

- The worst-case time complexity of O(N^2) can be a significant drawback, especially for large dictionaries. In the worst-case scenario, a large number of hash functions may need to be tried before finding one with minimal collisions.

  .

- Comparison between the 2 perfect hashing techniques w.r.t

1. **Mean Batch Insertion Time:**

No. of elements	$O(N^2)$	O(N)
10	3 ms	1 ms
100	3 ms	3 ms
1000	9 ms	13 ms
10000	67 ms	95 ms
100000	595 ms	590 ms
1000000	60 sec	57 sec 794 ms

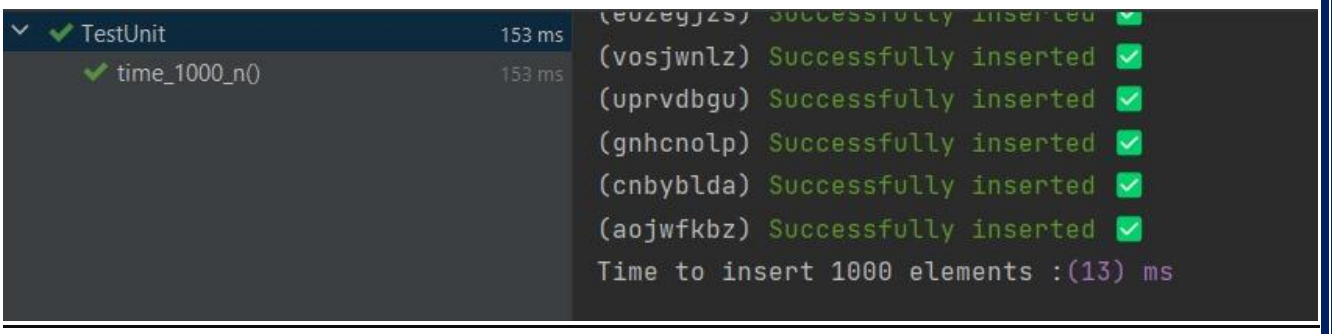
2. **Rehash Time:**

No. of elements	$O(N^2)$	O(N)
10	0 ms	0 ms
100	1 ms	0 ms
1000	83 ms	50 ms
10000	101 ms	88 ms
100000	196 ms	150 ms
1000000	254 ms	212 ms

3. **Mean Element Insert Time:**

No. of elements	$O(N^2)$	O(N)
10	1 ms	0 ms
100	6 ms	4 ms
1000	131 ms	101 ms
10000	230 ms	185 ms
100000	590 ms	502 ms
1000000	2 sec	1 sec 394 ms

JUnit test:


TestUnit 153 ms
time_1000_n() 153 ms
(vosjwnlz) Successfully inserted
(uprvdbgu) Successfully inserted
(gnhcnolp) Successfully inserted
(cnbyblda) Successfully inserted
(aojwfkbz) Successfully inserted
Time to insert 1000 elements :(13) ms