HASH FUNCTIONS

1. Hash Function 1 (Polynomial Rollback) :

This is a common hash function used in string pattern matching. It treats the string as a polynomial and evaluates its modular arithmetic. So,

Hash = s[0]*p^0 + s[1]*p^1 + ...+ s[n-1] * p^(n-1)

p = 31 : A small prime that minimizes collisions for alphabetic strings .

Modular arithmetic (m=1e9 + 9) prevents overflow .

2. Hash Function 2 (djb2) :

```
for (char c : key)
{
     hashValue = ((hashValue << 5) + hashValue) + c;
}
```

Here, hashValue = for all characters (hash*33+c)

So here 5381(initial) and 33(multiplier) reduce collisions.

TABLES FOR EACH LOAD FACTOR

1. Load Factor = 0.4

Method	Hash1 Function					Hash2 Function				
	# of collisions during insertion	Before Deletion		After Deletion		# of collisions during insertion	Before Deletion		After Deletion	
		Avg Search Time	Avg Probes	Avg Search Time	Avg Probes		Avg Search Time	Avg Probes	Avg Search Time	Avg Probes
Separate Chaining with balanced BST	1617	0 ns	N/A	0 ns	N/A	1645	0 ns	N/A	0 ns	N/A
Linear Probing with Step Adjustment	1834	317.2 ns	1.32	0 ns	1.772	1856	0 ns	1.308	0 ns	1.789
Double Hashing	1814	0 ns	1.225	0 ns	1.518	1849	0 ns	1.267	0 ns	1.525

2. Load Factor = 0.5

Method	Hash1 Function					Hash2 Function				
	# of collisions during insertion	Before Deletion		After Deletion		# of collisions during insertion	Before Deletion		After Deletion	
		Avg Search Time	Avg Probes	Avg Search Time	Avg Probes		Avg Search Time	Avg Probes	Avg Search Time	Avg Probes
Separate Chaining with balanced BST	1617	0 ns	N/A	0 ns	N/A	1645	0 ns	N/A	0 ns	N/A
Linear Probing with Step Adjustment	1839	0 ns	1.317	0 ns	1.826	1866	1151.7 ns	1.316	0 ns	1.826
Double Hashing	1810	0 ns	1.233	0 ns	1.532	1852	0 ns	1.237	1110.6 ns	1.484

3. Load Factor = 0.6

Method	Hash1 Function					Hash2 Function				
	# of collisions during insertion	Before Deletion		After Deletion		# of collisions during insertion	Before Deletion		After Deletion	
		Avg Search Time	Avg Probes	Avg Search Time	Avg Probes		Avg Search Time	Avg Probes	Avg Search Time	Avg Probes
Separate Chaining with balanced BST	1617	0 ns	N/A	0 ns	N/A	1645	0 ns	N/A	0 ns	N/A
Linear Probing with Step Adjustment	1847	0 ns	1.291	0 ns	1.805	1865	0 ns	1.317	0 ns	1.896
Double Hashing	1806	0 ns	1.264	0 ns	1.542	1870	0 ns	1.245	0 ns	1.533

4. Load Factor = 0.7

Method	Hash1 Function					Hash2 Function				
	# of collisions during insertion	Before Deletion		After Deletion		# of collisions during insertion	Before Deletion		After Deletion	
		Avg Search Time	Avg Probes	Avg Search Time	Avg Probes		Avg Search Time	Avg Probes	Avg Search Time	Avg Probes
Separate Chaining with balanced BST	1617	0 ns	N/A	1109.2 ns	N/A	1645	0 ns	N/A	0 ns	N/A
Linear Probing with Step Adjustment	1854	0 ns	1.269	0 ns	1.744	1865	0 ns	1.331	0 ns	1.901
Double Hashing	1817	0 ns	1.218	0 ns	1.523	1866	0 ns	1.235	0 ns	1.507

5. Load Factor = 0.8

Method	Hash1 Function					Hash2 Function				
	# of collisions during insertion	Before Deletion		After Deletion		# of collisions during insertion	Before Deletion		After Deletion	
		Avg Search Time	Avg Probes	Avg Search Time	Avg Probes		Avg Search Time	Avg Probes	Avg Search Time	Avg Probes
Separate Chaining with balanced BST	2848	0 ns	N/A	0 ns	N/A	2910	0 ns	N/A	0 ns	N/A
Linear Probing with Step Adjustment	3566	0 ns	2.384	0 ns	5.562	3686	0 ns	2.53	1093.8 ns	7.289
Double Hashing	3580	13618.8 ns	1.686	0 ns	3.016	3652	0 ns	1.744	0 ns	2.976

6. Load Factor = 0.9

Method	Hash1 Function					Hash2 Function				
	# of collisions during insertion	Before Deletion		After Deletion		# of collisions during insertion	Before Deletion		After Deletion	
		Avg Search Time	Avg Probes	Avg Search Time	Avg Probes		Avg Search Time	Avg Probes	Avg Search Time	Avg Probes
Separate Chaining with balanced BST	2848	0 ns	N/A	0 ns	N/A	2910	378.3 ns	N/A	0 ns	N/A
Linear Probing with Step Adjustment	3558	0 ns	2.18	0 ns	5.562	3687	0 ns	2.386	1015.5 ns	6.741
Double Hashing	3569	0 ns	1.837	0 ns	3.194	3636	0 ns	1.815	0 ns	3.14

Here , for all we are generating 10000 strings and then inserting them on Hash Table .

For all tables , their initial capacity is given 100 .

IMPACT OF LOAD FACTORS ON HASH TABLE PERFORMANCE

1. Separate Chaining :

When load factor increases

=> The number of elements in each chain grows .

Thus, chains become longer. So search time increases slightly . Here

the tables above cannot visualize it because of less amount of search.

=> Collisions increase. But as chaining allows multiple element per bucket

Performance degrades gradually.

2 . Open Addressing (Both for Linear probing and Double hashing) :

=> Higher load factor : More collisions and longer probe sequences.

So, as load factor increases performance decreases.

More precisely,

A) Linear Probing : Here groups of filled slots slow down searchs and insertions .

B) Double Hashing : Performs better than linear probing .