Using the skeleton app provided in the ***/Interesting Stuff/Hash Tables – Chaining/***folder, complete the implementation of a *Separate chaining with linked lists* collision resolution hash table.

Use the provided input files (100.data, 1000.data etc.) to feed your hash table.

Requirements:

* Implement all the operations as described in the skeleton
* At the end, after every element has been inserted, compute the standard deviation of all the buckets’ sizes. (see [here](http://en.wikipedia.org/wiki/Standard_deviation#Basic_examples) an example of how to compute this) – also, read up on what standard deviation is and try to figure out what a low/high value of such a measure would mean in our case.
* Try things out with a bad hash function first (a H0 – think about the worst hash function which could still somehow distribute entries in more than 1 bucket)

|  |  |  |
| --- | --- | --- |
| **Hash Function (char \* c, int i);** | **ISF \*** | **σ (standard deviation)** |
| H0 | .10 | 30.00, 70.00, 99.50, 223.38, 316.07, 499.90, 707.04, 999.95 |
| H0 | .20 | 21.79, 49.75, 70.53, 158.03, 223.55, 353.52, 499.97, 707.09 |
| H0 | .35 | 16.90, 37.80, 53.45, 119.52, 169.03, 267.26, 377.96, 534.52 |
| H0 | .50 | 14.00, 31.56, 44.68, 99.98, 141.41, 223.60, 316.22, 447.21 |
| H0 | .75 | 11.47, 25.79, 36.49, 81.64, 115.46, 182.57, 258.20, 365.15 |
| H0 | .85 | 10.78, 24.23, 34.28, 76.69, 108.46, 171.49, 242.53, 343.00 |
| H1 | .10 | 3.66, 3.32, 3.01, 3.06, 6.57, 15.47, 23.80, 34.99 |
| H1 | .20 | 2.72, 2.24, 2.14, 3.57, 6.79, 12.03, 17.55, 25.24 |
| H1 | .35 | 1.75, 1.78, 1.76, 3.64, 5.70, 9.42, 13.50, 19.24 |
| H1 | .50 | 1.51, 1.48, 1.39, 3.32, 4.95, 7.99, 11.37, 16.15 |
| H1 | .75 | 1.17, 1.17, 1.24, 2.87, 4.15, 6.59, 9.33, 13.22 |
| H1 | .85 | 1.10, 1.19, 1.20, 2.73, 3.92, 6.21, 8.78, 12.43 |
| H2 | .10 | 3.79, 2.73, 3.11, 3.04, 3.11, 3.14, 3.11, 3.16 |
| H2 | .20 | 2.65, 1.91, 2.30, 2.19, 2.21, 2.25, 2.20, 2.24 |
| H2 | .35 | 1.37, 1.68, 1.60, 1.69, 1.71, 1.71, 1.69, 1.69 |
| H2 | .50 | 1.33, 1.38, 1.38, 1.40, 1.40, 1.41, 1.41, 1.41 |
| H2 | .75 | 1.06, 1.15, 1.18, 1.14, 1.14, 1.16, 1.15, 1.15 |
| H2 | .85 | 1.04, 1.11, 1.05, 1.05, 1.10, 1.08, 1.09, 1.08 |
| H3 | .10 | 3.22, 3.64, 3.12, 3.16, 3.20, 3.10, 3.17, 3.14 |
| H3 | .20 | 2.02, 2.31, 2.39, 2.18, 2.22, 2.20, 2.25, 2.22 |
| H3 | .35 | 1.63, 1.70, 1.65, 1.69, 1.66, 1.69, 1.69, 1.69 |
| H3 | .50 | 1.39, 1.50, 1.43, 1.39, 1.41, 1.41, 1.41, 1.42 |
| H3 | .75 | 1.24, 1.15, 1.18, 1.16, 1.15, 1.15, 1.16, 1.16 |
| H3 | .85 | 1.21, 1.05, 1.08, 1.09, 1.08, 1.08, 1.09, 1.09 |

\* ISF = Initial Size Factor

Have other combinations in mind? Feel free to fill-up the table with more tries to see if any interesting results come up!

H0: bad hash function!

H1:

***int hashFunction(char \* content, int i)***

***{***

***int length = strlen(content);***

***int k, sum;***

***for (sum=0, k=0; k < length; k++)***

***{***

***sum += content[k];***

***}***

***return sum % size;***

***}***

H2 – H3 🡪 your choices!

* Try to improve each time

Deadlines:

30411 – 18.05.2015 (before 12:00 if you want review and before 23:59 if you want a grade)  
30414 – 19.05.2015 (before 12:00 if you want review and before 23:59 if you want a grade)