**1. Does you HashMap offer O(1) performance?**

Yes. A Graph of performance time of a combination of puts and gets at both default size and at a size of N creates a slope of 1 in log, log scale (See graph A).

**2. Does rehashing significantly slow down your HashMap's performance?**

No. At large numbers of data the time taken grows at approximately the same rate (See Graph A).

**3. How does your HashMap's performance compare to the JDK's HashMap?**

myHashMap has the same performance as the JDK HashMap both with rehashing and without (See Graphs B and C).

**A)**

**B)**

**C)**

**Raw Data : (\*NOTE: all 0’s changed to 1’s so they would show up in the graphs)**

|  |  |
| --- | --- |
| myHashMap put and get default size | |
| Number of Items | Time(ms) |
| 2048 | 16 |
| 4096 | 1 |
| 8192 | 15 |
| 16384 | 16 |
| 32768 | 16 |
| 65536 | 31 |
| 131072 | 31 |
| 262144 | 109 |
| 524288 | 187 |
| 1048576 | 422 |
| 2097152 | 873 |
| 4194304 | 4104 |
| 8388608 | 7598 |
| 16777216 | 18505 |
|  |  |
| jdk HashMap put and get default size | |
| Number of Items | Time(ms) |
| 2048 | 16 |
| 4096 | 1 |
| 8192 | 16 |
| 16384 | 16 |
| 32768 | 1 |
| 65536 | 15 |
| 131072 | 47 |
| 262144 | 94 |
| 524288 | 359 |
| 1048576 | 530 |
| 2097152 | 1031 |
| 4194304 | 2122 |
| 8388608 | 7021 |
| 16777216 | 25024 |
|  |  |
| myHashMap put and get with size MAX | |
| Number of Items | Time(ms) |
| 2048 | 1 |
| 4096 | 1 |
| 8192 | 1 |
| 16384 | 1 |
| 32768 | 1 |
| 65536 | 16 |
| 131072 | 219 |
| 262144 | 63 |
| 524288 | 141 |
| 1048576 | 484 |
| 2097152 | 1561 |
| 4194304 | 2044 |
| 8388608 | 7286 |
| 16777216 | 16646 |
|  |  |
| jdk HashMap put and get with max size(no rehash) | |
| Number of Items | Time(ms) |
| 2048 | 1 |
| 4096 | 1 |
| 8192 | 1 |
| 16384 | 1 |
| 32768 | 16 |
| 65536 | 16 |
| 131072 | 31 |
| 262144 | 62 |
| 524288 | 124 |
| 1048576 | 297 |
| 2097152 | 811 |
| 4194304 | 3308 |
| 8388608 | 10408 |
| 16777216 | 16896 |