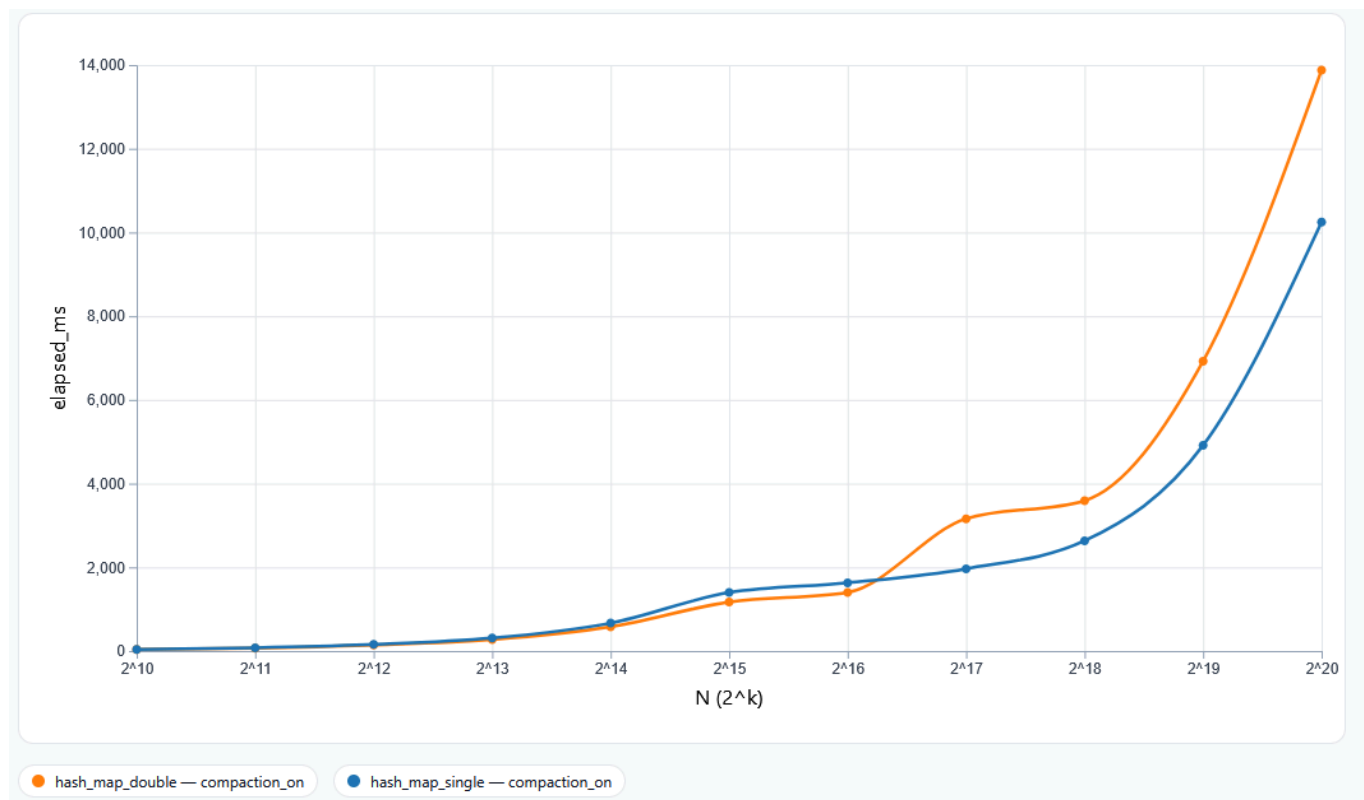


Project 5 Part 2 - Empirical Analysis - Least Recently Used Profile / Hash Table

Expectations vs. Observations

For small N , I expected double probing to be faster because single probing tends to create long runs of occupied slots in the table, which increases the number of comparisons per operation. Double probing is more likely to create shorter clusters. For large N , I expected double probing to remain faster and for the gap to widen. In a larger table, single probing causes those runs to become extremely long, making each insertion much more expensive, while double probing continues to form smaller, more scattered clusters.

For N below 2^{16} , double probing does show a small advantage as expected: about 19% faster at 2^{15} for example, with 1172 ms for double and 1399 ms for single, but the gap remains narrow. Above that point, single probing regains the performance advantage and becomes increasingly faster, with the gap widening as N increases, which disagrees with my expectations. The absolute separation is largest for the largest N , 2^{20} , where double probing is about 40% slower, with 13880ms for double and 10249ms for single.



"Work" per operation vs. wall-clock time

Fewer probes do not consistently correlate with lower time in my data. In the range I found a performance advantage for double probing, the average probes do seem to correlate, because in that range the averages probes for single-probing is increasing with N , and staying close to constant for double probing, explaining the small performance advantage in that range. However, at larger $N > 2^{17}$, both methods showed a similar number of average probes. At $N = 2^{20}$, 3.379 for single and 3.256 for double, but the performance advantage had flipped decisively to single probing as mentioned. I did find a metric that could explain the mismatch, latency, which is milliseconds per operation. As N increased, at $N = 2^{20}$, double probing cost 0.000649 ms/op, and single probing cost 0.000479 ms/op, which is about 35% faster. This indicates that hashing overhead for double probing is expensive, and that cost becomes significant at high values of N .

Hashing cost and memory locality

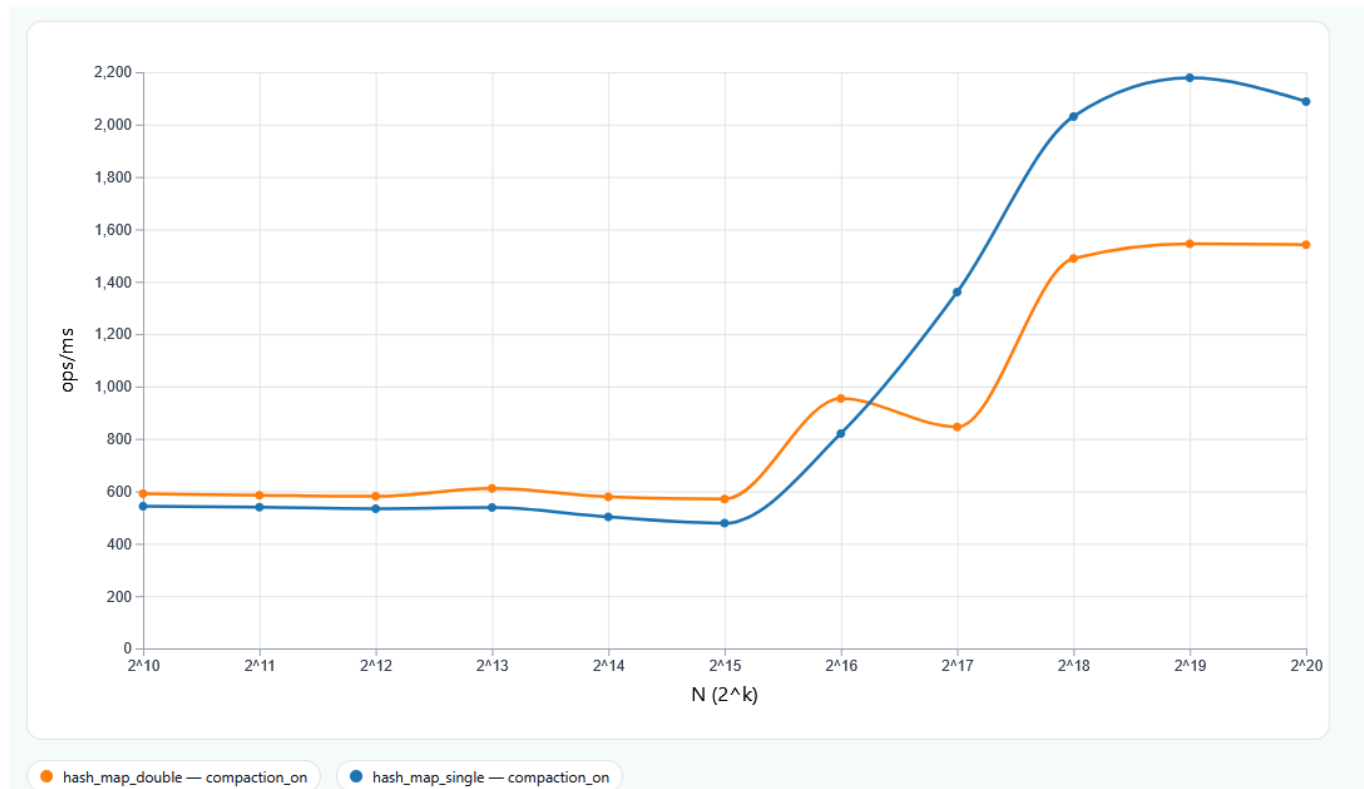
The primary hash is computed for every operation regardless, but in double probing an additional hash is computed, so the per-operation hashing costs is double the work for double probing. I expect this constant factor to matter for the largest N , because as already shown the number of probes becomes similar in that range, so the constant factor of the hashing costs will dominate. The data mentioned above regarding ms/op at $N=2^{20}$ does support this.

Single probing has better locality due to the clustering, so whenever its probe count is similar to double, it tends to perform well because it traverses short, contiguous clusters that are cache friendly. This is obvious at $N = 2^{20}$, where the probe counts are nearly identical, but single probing is ~35% faster. The broken, short cluster pattern of double probing helps most in the mid-range N . For $N = 2^{15}$, double probing has average probes of 6.613, where single probing average probes is 20.457, a huge relative difference. In that region we saw double probing pull slightly ahead because reducing probe length still outweighs the extra hashing cost and weaker locality with regards to memory.

Compaction Effects

Compactions only occur for $N < 2^{17}$. In that range, double probing triggers more compactions at each N , for example at $N = 2^{12}$ double probing triggers 24 compactions where single probing triggers only 19. It doesn't align precisely with tombstone%, which is 15% for double and 8% for single at $N = 2^{13}$. The compactions tend to align with effective load factor%, which remains higher for double probing at all $N > 2^{12}$. Runs with more compactions do not reliably show higher elapsed times, so the correlation is weak. For example, at $N=2^{14}$, double probing has 23 compactions and 577.6ms elapsed, but single probing has 18 compactions and 666.2ms elapsed; single is slower, even with fewer compactions. Although compaction does have a cost, it seems to be outweighed by other factors, specifically hashing

overhead at high N and probe count at low N. Before compaction, the map shows long runs of active slots with almost no available gaps, leading to very long probe chains. After compaction, these long runs fracture and more slots are available more often, which is to say the clusters become smaller. This structural change directly reduces average probe lengths, because insertion operations encounter available slots much sooner.



Throughput and Latency

Comparing throughput and latency, which are inverses of each other, to elapsed time, they all tell a consistent story; the crossover point at $N=2^{17}$ where throughput of single probing surpasses double is precisely the point where the elapsed time shifts in favor of single probing. At $N = 2^{20}$, single probing achieves 2088 ops/ms, while double probing achieves only 1541 ops/ms, a gap of ~35%, even though their average probe counts are nearly identical (3.379 vs 3.256). This lines up perfectly with the previously observed gap in elapsed times, but as a metric throughput stands out because it directly shows how much useful work each method produces per millisecond.

Occupancy

There is not much change in the occupancy measures like load factor, effective load factor, and tombstone% across N, they are roughly the same for both single and double at each N. There is a significant spike in tombstone% at $N= 2^{17}$, which was previously noted as the crossover point, where single tombstone% spikes to 35% and double to 46%. The time gap is greatest at $N = 2^{20}$, where all three measures don't really differ at all: effective load factor for single is 10% and for double it is 11%. The explanation must be that occupancy factors are not a large

influence on elapsed time. Since the average probes are also similar there, this is more evidence for the conclusion that the per-operation cost dominates at large N .

Before / After Compaction Structure

Before compaction, the run-length distribution had a long right tail, with many medium and large run lengths, indicating significant clustering and accumulated tombstones. After compaction, the histogram shifted sharply left: long runs disappeared, the maximum run length dropped significantly, and most runs became short. This structural cleanup greatly reduces the length of the average probe.

Maps + Histograms

Snapshots of 0/1 Maps:

```
double_probing 4096
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]


```
1110101111101111111011101101100010010111011111111011101010101010010110111
1111110001111111110111
1111111110111010011110111011111010111110001110111111101111101111101101101111
0111110111111111101100
0100111011011111011011111101010011101111100111111111111101010010111101111010
0111101110111100111110
1111101111101111101001111110101111110110101011110100111010110111010101111111
111111111011110111111
11111111111111111011110011111110101110010101101111011110011111001001100
10111111111111111001110
11111110111111111011111010110110011111110111111001111110110010011001110111011
1101110111111110101110
1111101110111110011110111011111111101110111111011111110111111011111111
1111011001101001111111
11010101101100101110111111110110111011011001101100111110101011011101101110
1100111010110110111110
110101111011111011111111101101111000111101001110111011111110101111110111011110
1110110111100111111111
101111101101010001111111111110111111101111111111100111011101011011101001111
11110111111111111111101
11111110101001101110101011001111011011111101111011110001111111110111111111
1110011011010000110110
1101111111011111000111110100111101111111001011111101101111111011111010101111
1101011101101011101111
1101111111101111010110111111110111111111111111111110111011111111011111111
1111001111001011100111
111100110110111111111111100111111111111111111110111111110111110101111001
01111011111111111111101
1001011111011100001111001111011111110011111101101011111111111101111101001111101
1000111011011111111111
01100110100011011111111111101111111101111111111110111111111111101111111
0111111011110101110101
1111111110110111001111111111111111101101111110111011111101111101101111111
11111011011111111111101
1101011110100010001111011101111111011110101101111100101111101111111111111
1111110101110110001111
111111111110111001110111110100010111111011110011010110111111011111111111110
1100001111011110110111
101110100111001111111111111111111111100111101100111001111111101101101111111011
1011111101110111011100
101111011111111101111011101011100101011111001110111111111111111111101110101
0111011110101010111111
11111101111111111111111111111101001111010111111101111011111011101101000
1111111110101110110111
111101111110010111111110110101111110111011111111111101111111111111111111
1111000011111010111111
```

00111110111111111011110111111111101111101111111111011111111111111011
0111100111111111110101
1111111011010111110000110110111110110110111111111011110111111111111111
11111111111101111110
11010111111011001101111110011111110111101111111111101111111111110111111
111101011101110111110
1111111011101111111111101111110100111100111100110101111101001011111011001101
1010111011111011110100
1101110011110101110010111111111011100101110111111001111111111000111010111
111111111111110110111
1101001111100111001111111011110011010101110111111101110111110110011101111
111111111101111110111
101111011111111111111111111100111111011010001110111011111110111110111011101
1101010111111111110001
10111011011111000110111111111001001111111111101111011111110011010111111011
1111111101011111011111
00011111111011011111111111111011010111110101111111111110111110101111001111010
1111111111101111110111
1011111111111111101011111111010011111110111110111111111111011111101100101111
1101111000010111010011
010001111111011111011011110111001111101111110110001101111111011011101111
0111100111110111001110
1111111111111010101101011110111110011111111011101111111111110001111110101011
0111111010111011111011
11101111111111111111101111111001110110111100111101111101011111110111011111
1010111101110110011101
0111110011111111111010101101111001111110111011010111111011111100100110101111
1101011011101111111011
1110011111110110111110111110111100001011111101111110110110111100101111101111
1111011111111100111111
1111111111101111111111101010101111111011110111111111110001111010100011110100111
1100101100111111010111
00110110111111111011101111101001111000011111011111101110111111110011101111111111
11111111111001101110110
100111110110001110111111101111101111011110111111101111110111111001111010010
0101110101111001101111
0001100111101011111011111110111111110111001011111110111111111111011111
1110110111111010110110
01111111111110010111111110111111000111101001001111111111101111111111111111
11100111111101011111011
11010111110101011111111010111111111101111111111101111001010011111111101111
11111101111111110011101
011011111110111000101111111101111111110111101101101111101111111111111111
1101111111100011111111
11011110111111111111111111111011111111001111111001011011111111111101
11111111111010111011111

[illegible]

```
0111111011111110111111100111011110111110011011111101100111001111111100101101
1101011111101101111111
10110101101111111101011111111101010100010111011101010111111011101011011110101
1110111110111111111111
101111111011101111111111111110111111011101111010111110110110110110110010111
111110101111011111110
11001111111111111111111110011111101111111111110010101111001111111101111111111
1111011011111111111100
11110111101011110011111101111101011011101111111001101111110111110110111110101
1110110011111111101111
1110101111101101101111111111011011111101110111101011111110100111111001110
1111111111001101110111
1111110111011111111011111111100111011010110111111111111111111111111111111111
1101111110111111111110
111111111111111111101111111111110101111110101101101011111010111101111
111111110111011011111
10111111111101110101101111111111110111111010011111101111111110111111111101
1000110001011111111000
11110011001011111101111100101100111111111101010111111111011111111111011011010
1110111111111111111101
11111110001110111010111111110100111011101111000111111111111111111111111111
1111011111110111111011
11111111110111101100111110111100001110011111100111110111111010011011101101111
0111101101111111111101
11011110110110011111010010111110111111110110111011111001111111110100101111111
1111101111111110000111
01111101100110010111101111011101001110010110111101111111111010101011111111111
1111111110011110110111
0111110111101111110011011100101111111010001001111011110111011101110111111111
1111011111110111110111
111101111110111101111111111101111010100011111010101101111011101110111111111
1110011111111010111110
1111110111100100111111111100011111101011111011011111100110011101011010100011
1111111111110111110111
010001101111111111110111110111011011101111111111111111111111001100101110
1101011111101011110110
11011111111111101111011101110111011100111101011111111100111100111111111101
1110101110101111111111
10011011101111110111010011111111111111011111111101001101010111101101111111111
1111011111111011111111
1111110111111111001111111110011110101111110011111111110110111110111011
1111111110111111001001
111111101011110001111111111010111011111111111111111111111111111111011111101110
1111111110000010011101
1010111101111111111101111111010011011111110111111111111111111111111111110
1111011111011111111111
```

[illegible]

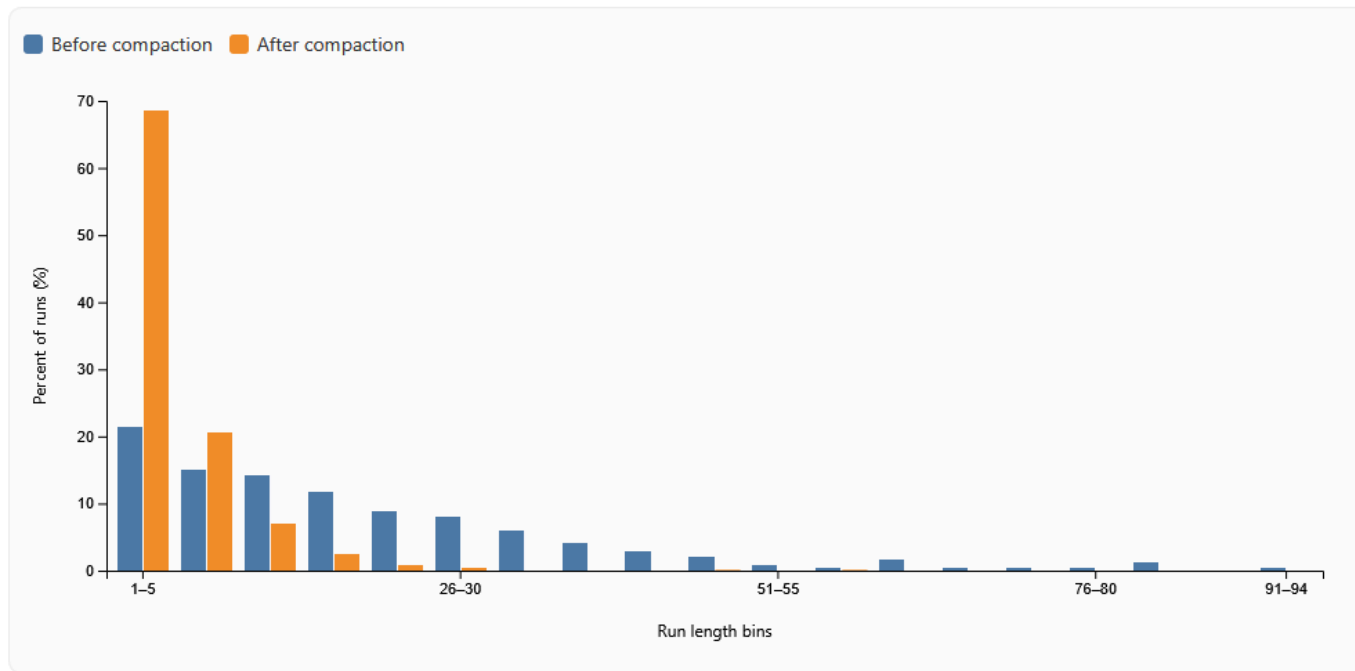
```
11101111011101111101100110011110110111111011111111011100111001111111111111011
0001111110110111010111
0111111011101111101100111111011111111100011111111101111101111101111110101001
1111010011101111101111
11011110111110101011111101110110110101111011101111101011011110011011101010110
1111111011110011011111
1111111111110111111101101101111011110111001010111110111111011111101111101011
1111111111011111111001
11111111111100111111111111111101011101111110011011111110110111111111111110
010111111111110111111
1101011110111111111011111111011111101011101110110111111101110110111010111111
1011001110011111111111
011111111110111001111111111101101101011111101110111111001111010111110101101011
0111111110010111111011
111111001111111111010111100100110011111111111111111111111111111011111110101001
11111111111111111111
011101011111100101111011111111111100111111101111111101111110111111111111111
1111111101111110111011
10110111000110111111011111110111011101011001010111111101011011011011111011111
111011111111011111110
1111100101101111111111011111111011011101111110111111011111101011011111110110
010111111101011111110
01111011111111111111111111111110100111111111011111101111111111111111101111110011
1011111111111111101101
11111011111011111101111101011110111110110101011011111111111101111111111111011
11111011011111111011111
1101101011101111110100111001111010111100100010011001101110111111111110111101111
1111111111001110110110
111111111111111010111111011111110101110111111101111111111111111111111011111
1111111111101111111111
110011111011111111110111111010111011111011111111111111111111111111101011011
1111111110111110111111
1111111111111101111111110011111111011111001100111111111101100111111111110101101
1110110101111111011110
1101111010111101101101110011111111101101111111101111111011111110111110111
1111111111111011111111
111111001111111111110110111101101111111111111111111111111111111011111110111
1110101011111101111111
1110111111110100101100111110101111101111110010111101101111111111110111111111
1111111111111111111111
11100111101111101111111111101101110111111101111111111111111111011110111111
1101111101111011101111
101111101111101111111111111111111111111111100111101111011111011101101111101
0111100111111110110111
1111010101111111001111110111110111101111111111111111111010011111111111111111
10010111111011101110111
```

```
11110011110111110001111101011111101101111111100011101011111111001011101
10010111111011101110010
110111110101110011011111111111111111111111111101011110011111110101011101111111010101
0111111111111111001110
10011111111110011111010101111101111011110101001011101111111001111110111011101
1110101111011111110111
1101011111111011011101111100111110111101110111101011111011111011111111111101
1111110101111010101011
111101111111111111111101010111111010111011101111001111111111100110111111111
0001111011111110111101
111111011111110111101111111111111111111110111111000110111111010111111111111101
1111110011110111110001
111111111111011111111101111110101110111011111011111011111011111111010011101111
1101111010111111101111
11011111111010111111001101010011100011101111011111111001111110110101010100111
1111111101111100111011
11101110111111111011111111111111111111111111111111111111010110111
1111011110111111110111
1111111101110110101110001111011010111111111100111111111010111111111111110111
1111111011110111011111
1111111100111101111111111110111111111110111111101111111111110011111111110011
1111110111011110111111
01110111111111000101111001111001101001111111111110111111111111011111111111011111
1111011111111011101101
11111111111111111101111101111111101111111011111101001011011001111100011111111
1001110110101111110111
11111110110101111111111110110101110010111111011100011101100100111111111010101101
1011111100101111111101
11101011111111111111111100010110111111101111110101110011111111001111111110110011
10010111011111111010111
11011111110111111111111111111111111111110111110111111011111111111101110
1101111101111111111101
011011111111101111111010111111011111111111111011100111110100111111011111111111111
0101101101111001101111
111111011111111101111111111111111111111111101101111111111111111111111111
00111110111111110111111
111111111111011110111111111111111111111111011111111111110100111111000101
1110100101111111110111
111110001111011111011111111111111111111111100111111111111101111111111111011111
11111111110101111101101
1101111001110111111010111011100101110111111111111111111111111011111110111
01111111111111111011111
1101011111111111001010011011111111111111100111111111111101010111111
1111111111111010111110
110011011111111111111111111111111111111111101001011011111111111111111111
1111011010101111111100
```

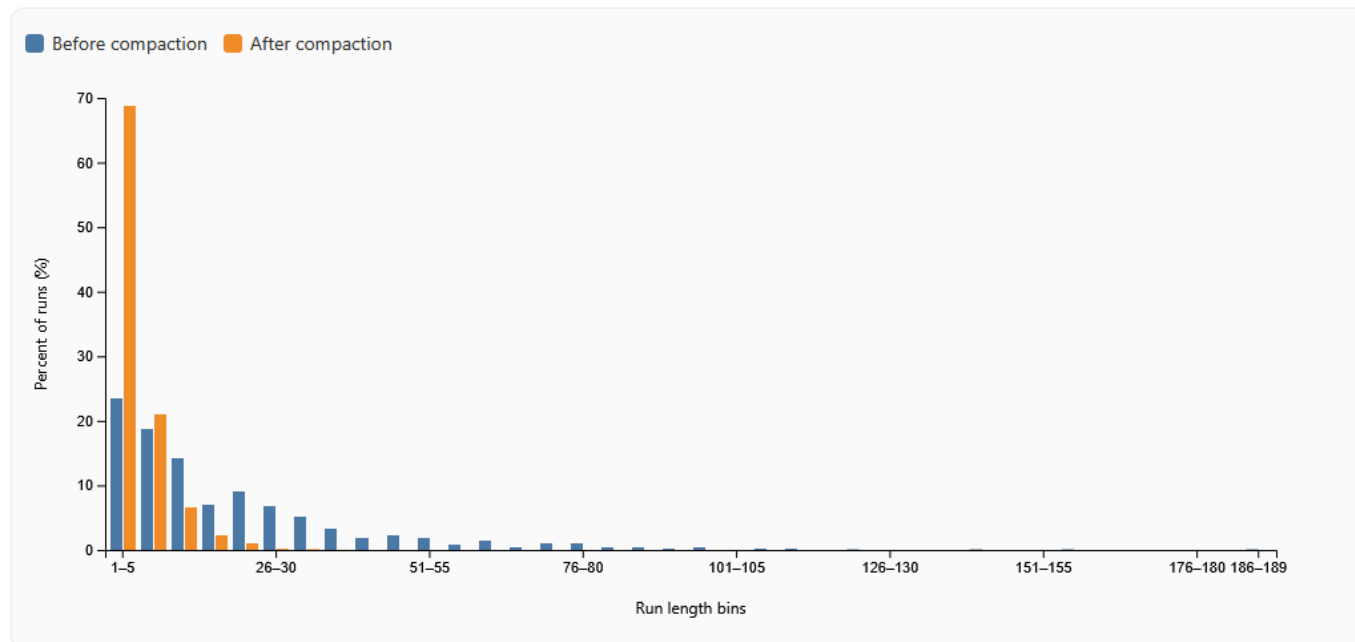

[illegible]

0

Header: double_probing 4096 — parsed lengths: before=5101 bits, after=5101 bits



Header: double_probing 16384 — parsed lengths: before=20479 bits, after=20479 bits



Before compaction, the longest runs appear as a few very large blocks with few gaps, shown by the tail of the histogram. Double and single probing show visibly different clustering. Double probing produces more scattered medium-length clusters, and single probing produces longer runs. Compaction tends to break up these long runs, fracturing them into small clusters and removing the long stretches. After compaction, the long runs shrink. The max run length decreases and most runs fall into the smallest buckets. Long runs directly correspond to higher probe counts. Removing tombstones and breaking the clusters lowers average probes, which

reduces elapsed ms as discusses previously. After compaction, tombstone_pct drops, effective load decreases, and the histogram shows shorter runs.