Andrew Chiang

Exercise 3

02/19/2023

Conditions:

Table Size: 11

Double Factor: 7,

hash function: hash(Key)=key,

value: 2\*key.

**\*\*RANDOM ORDER\*\***

**5 414 138 111 25 35 67 59 37**

**Linear Probing:**

1. hash (5) = 5, 5%11 = 5 so insert in position 5.
2. hash (414) = 414, 414%11 = 7 so insert in position 7.
3. hash (138) = 138, 138%11 = 6 so insert in position 6.
4. hash (111) = 111, 111%11 = 1 so insert in position 1.
5. hash (25) = 25, 25%11 = 3 so insert in position 3.
6. hash (35) = 35, 35%11 = 2 so insert in position 2.
7. hash (67) = 67, 67%11 = 1 so insert in position 1 -> BAD.
   1. collision (1)
   2. (67+1)%11 = 2
   3. collision (2)
   4. (68+1)%11 = 3
   5. collision (3)
   6. (69+1)%11 = 4 so insert in position 4
8. hash (59) = 59, 59%11 = 4 so insert in position 4 -> BAD.
   1. collision (4)
   2. (59+1)%11 = 5
   3. collision (5)
   4. (60+1)%11 = 6
   5. collision (6)
   6. (61+1)%11 = 7
   7. collision (7)
   8. (62+1)%11 = 8 so insert in position 8
9. hash (37) = 37, 37%11 = 4 so insert in position 4 -> BAD.
   1. collision (8)
   2. (37+1)%11 = 5
   3. collision (9)
   4. (38+1)%11 = 6
   5. collision (10)
   6. (39+1)%11 = 7
   7. collision (11)
   8. (40+1)%11 = 8
   9. collision (12)
   10. (41+1)%11 = 9 so insert in position 9

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|  | 111/222 | 35/70 | 25/50 | 67/134 | 5/10 | 138/276 | 414/828 | 59/118 | 37/74 |  |

*Storing Key Value Pairs*

To put these numbers takes 12 collisions.

To retrieve these number takes the same 12 collisions.

Thus, Linear Probing takes **24 collisions.**

print table.size()=11

index=1 key=111 value=222

index=2 key=35 value=70

index=3 key=25 value=50

index=4 key=67 value=134

index=5 key=5 value=10

index=6 key=138 value=276

index=7 key=414 value=828

index=8 key=59 value=118

index=9 key=37 value=74

\*\*\* Linear probing Random Order End \*\*\*

**Quadratic Probing:**

1. hash (5) = 5, 5%11 = 5 so insert in position 5.
2. hash (414) = 414, 414%11 = 7 so insert in position 7.
3. hash (138) = 138, 138%11 = 6 so insert in position 6.
4. hash (111) = 111, 111%11 = 1 so insert in position 1.
5. hash (25) = 25, 25%11 = 3 so insert in position 3.
6. hash (35) = 35, 35%11 = 2 so insert in position 2.
7. hash (67) = 67, 67%11 = 1 so insert in position 1 -> BAD.
   1. collision (1)
   2. (1+1\*1)%11 = 2
   3. collision (2)
   4. (1+2\*2)%11 = 5
   5. collision (3)
   6. (1+3\*3)%11 = 10 so insert in position 10
8. hash (59) = 59, 59%11 = 4 so insert in position 4.
9. hash (37) = 37, 37%11 = 4 so insert in position 4 -> BAD.
   1. collision (4)
   2. (4+1\*1)%11 = 5
   3. collision (5)
   4. (4+2\*2)%11 = 8 so insert in position 8

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|  | 111/222 | 35/70 | 25/50 | 59/118 | 5/10 | 138/276 | 414/828 | 37/74 |  | 67/134 |

*Storing Key Value Pairs*

To put these numbers takes 5 collisions.

To retrieve these number takes the same 5 collisions.

Thus, Quadratic Probing takes **10 collisions.**

print table.size()=11

index=1 key=111 value=222

index=2 key=35 value=70

index=3 key=25 value=50

index=4 key=59 value=118

index=5 key=5 value=10

index=6 key=138 value=276

index=7 key=414 value=828

index=8 key=37 value=74

index=10 key=67 value=134

\*\*\* Quadratic probing Random Order End \*\*\*

**Double Hashing Probing:**

1. hash (5) = 5, 5%11 = 5 so insert in position 5.
2. hash (414) = 414, 414%11 = 7 so insert in position 7.
3. hash (138) = 138, 138%11 = 6 so insert in position 6.
4. hash (111) = 111, 111%11 = 1 so insert in position 1.
5. hash (25) = 25, 25%11 = 3 so insert in position 3.
6. hash (35) = 35, 35%11 = 2 so insert in position 2.
7. hash (67) = 67, 67%11 = 1 so insert in position 1 -> BAD.
   1. collision (1)
   2. Check (f(1,67)+1)%11, where f(1,67) is 1\*hash2(67) is 1\*( 7 – (67%7)) = 1-4 is 3, so index= (3+1)%11 is 4, it is free, take the spot in position 4
8. hash (59) = 59, 59%11 = 4 so insert in position 4 -> BAD.
   1. collision (2)
   2. Check (f(1,59)+4)%11, where f(1,59) is 1\*hash2(59) is 1\*( 7 – (59%7)) = 7- 3 is 4, so index= (4+4)%11 is 3, it is free, take the spot in position 8
9. hash (37) = 37, 37%11 = 4 so insert in position 4 -> BAD.
   1. collision (3)
   2. Check (f(1,37)+4)%11, where f(1,37) is 1\*hash2(37) is 1\*( 7– (37%7)) = 7-2 is 5, so index= (5+4)%11 is 9, it is free, take the spot in position 9

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|  | 111/222 | 35/70 | 25/50 | 67/134 | 5/10 | 138/276 | 414/828 | 59/118 | 37/74 |  |

*Storing Key Value Pairs*

To put these numbers takes 3 collisions.

To retrieve these number takes the same 3 collisions.

Thus, Double Hashing Probing takes **6 collisions.**

print table.size()=11

index=1 key=111 value=222

index=2 key=35 value=70

index=3 key=25 value=50

index=4 key=67 value=134

index=5 key=5 value=10

index=6 key=138 value=276

index=7 key=414 value=828

index=8 key=59 value=118

index=10 key=37 value=74

\*\*\* Double probing Random Order End \*\*\*

**ASCENDING ORDER:**

**5 25 35 37 59 67 111 138 414**

**Linear Probing:**

1. hash (5) = 5, 5%11 = 5 so insert in position 5.
2. hash (25) = 25, 25%11 = 3 so insert in position 3.
3. hash (35) = 35, 35%11 = 2 so insert in position 2.
4. hash (37) = 37, 37%11 = 4 so insert in position 4
5. hash (59) = 59, 59%11 = 4 so insert in position 4 -> BAD.
   1. collision (1)
   2. (59+1)%11 = 5
   3. collision (2)
   4. (60+1)%11 = 6 so insert in position 6.
6. hash (67) = 67, 67%11 = 1 so insert in position 1
7. hash (111) = 111, 111%11 = 1 so insert in position 1 -> BAD.
   1. collision (3)
   2. (111+1)%11 = 2
   3. collision (4)
   4. (112+1)%11 = 3
   5. collision (5)
   6. (113+1)%11 = 4
   7. collision (6)
   8. (114+1)%11 = 5
   9. collision (7)
   10. (115+1)%11 = 6
   11. collision (8)
   12. (116+1)%11 = 7 so insert in position 7.
8. hash (138) = 138, 138%11 = 6 so insert in position 6 -> BAD.
   1. collision (9)
   2. (138+1)%11 = 7
   3. collision (10)
   4. (139+1)%11 = 8 so insert in position 8.
9. hash (414) = 414, 414%11 = 7 so insert in position 7.
   1. collision (11)
   2. (414+1)%11 = 8
   3. collision (12)
   4. (415+1)%11 = 9 so insert in position 9

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|  | 67/134 | 35/70 | 25/50 | 37/74 | 5/10 | 59/118 | 111/222 | 138/276 | 414/828 |  |

*Storing Key Value Pairs*

To put these numbers takes 12 collisions.

To retrieve these number takes the same 12 collisions.

Thus, Linear Probing takes **24 collisions.**

print table.size()=11

index=1 key=67 value=134

index=2 key=35 value=70

index=3 key=25 value=50

index=4 key=37 value=74

index=5 key=5 value=10

index=6 key=59 value=118

index=7 key=111 value=222

index=8 key=138 value=276

index=9 key=414 value=828

\*\*\* Linear probing Ascending Order End \*\*\*

**Quadratic Probing:**

1. hash (5) = 5, 5%11 = 5 so insert in position 5.
2. hash (25) = 25, 25%11 = 3 so insert in position 3.
3. hash (35) = 35, 35%11 = 2 so insert in position 2.
4. hash (37) = 37, 37%11 = 4 so insert in position 4
5. hash (59) = 59, 59%11 = 4 so insert in position 4 -> BAD.
   1. collision (1)
   2. (4+1\*1)%11 = 5
   3. collision (2)
   4. (4+2\*2)%11 = 8 so insert in position 8
6. hash (67) = 67, 67%11 = 1 so insert in position 1.
7. hash (111) = 111, 111%11 = 1 so insert in position 1 -> BAD.
   1. collision (3)
   2. (1+1\*1)%11 = 2
   3. collision (4)
   4. (1+2\*2)%11 = 5
   5. collision (5)
   6. (1+3\*3)%11 = 10 so insert in position 10
8. hash (138) = 138, 138%11 = 6 so insert in position 6.
9. hash (414) = 414, 414%11 = 7 so insert in position 7.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|  | 67/134 | 35/70 | 25/50 | 37/74 | 5/10 | 138/276 | 414/828 | 59/118 |  | 111/222 |

*Storing Key Value Pairs*

To put these numbers takes 5 collisions.

To retrieve these number takes the same 5 collisions.

Thus, Quadratic Probing takes **10 collisions.**

print table.size()=11

index=1 key=67 value=134

index=2 key=35 value=70

index=3 key=25 value=50

index=4 key=37 value=74

index=5 key=5 value=10

index=6 key=138 value=276

index=7 key=414 value=828

index=8 key=59 value=118

index=10 key=111 value=222

\*\*\* Quadratic probing Ascending Order End \*\*\*

**Double Hashing Probing:**

1. hash (5) = 5, 5%11 = 5 so insert in position 5.
2. hash (25) = 25, 25%11 = 3 so insert in position 3.
3. hash (35) = 35, 35%11 = 2 so insert in position 2.
4. hash (37) = 37, 37%11 = 4 so insert in position 4
5. hash (59) = 59, 59%11 = 4 so insert in position 4 -> BAD.
   1. collision (1)
   2. Check (f(1,59)+4)%11, where f(1,59) is 1\*hash2(59) is 1\*( 7 – (59%7)) = 7- 3 is 4, so index= (4+4)%11 is 8, it is free, take the spot in position 8
6. hash (67) = 67, 67%11 = 1 so insert in position 1.
7. hash (111) = 111, 111%11 = 1 so insert in position 1 -> BAD.
   1. collision (2)
   2. Check (f(1,111)+1)%11, where f(1,111) is 1\*hash2(111) is 1\*( 7 – (111%7)) = 7-6 is 1, so index= (1+1)%11 is 2
   3. collision (3)
   4. Check (f(2,111)+1)%11, where f(2,111) is 2\*hash2(111) is 2\*( 7 – (111%7)) = 2\*(7-6) is 2, so index= (2+1)%11 is 3
   5. collision (4)
   6. Check (f(3,111)+1)%11, where f(3,111) is 3\*hash2(111) is 3\*( 7 – (111%7)) = 3\*(7-6) is 3, so index= (3+1)%11 is 4
   7. collision (5)
   8. Check (f(4,111)+1)%11, where f(4,111) is 4\*hash2(111) is 4\*( 7 – (111%7)) = 4\*(7-6) is 4, so index= (4+1)%11 is 5
   9. collision (6)
   10. Check (f(5,111)+1)%11, where f(5,111) is 5\*hash2(111) is 5\*( 7 – (111%7)) = 5\*(7-6) is 5, so index= (5+1)%11 is 6 so insert in position 6
8. hash (138) = 138, 138%11 = 6 so insert in position 6 -> BAD.
   1. collision (7)
   2. Check (f(1,138)+6)%11, where f(1,138) is 1\*hash2(138) is 1\*( 7 – (138%7)) = 7-5 is 2, so index= (2+6)%11 is 8
   3. collision (8)
   4. Check (f(2,138)+6)%11, where f(2,138) is 2\*hash2(138) is 2\*( 7 – (138%7)) = 2\*(7-5) is 4, so index= (4+6)%11 is 10 so insert in position 10
9. hash (414) = 414, 414%11 = 5 so insert in position 7.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|  | 67/134 | 35/70 | 25/50 | 37/74 | 5/10 | 111/222 | 414/828 | 59/118 |  | 138/276 |

*Storing Key Value Pairs*

To put these numbers takes 8 collisions.

To retrieve these number takes the same 8 collisions.

Thus, Double Hashing Probing takes 1**6 collisions.**

print table.size()=11

index=1 key=67 value=134

index=2 key=35 value=70

index=3 key=25 value=50

index=4 key=37 value=74

index=5 key=5 value=10

index=6 key=111 value=222

index=7 key=414 value=828

index=8 key=59 value=118

index=10 key=138 value=276

\*\*\* Double probing Ascending Order End \*\*\*

**DESCENDING ORDER:**

**414 138 111 67 59 37 35 25 5**

**Linear Probing:**

1. hash (414) = 414, 414%11 = 7 so insert in position 7.
2. hash (138) = 138, 138%11 = 6 so insert in position 6.
3. hash (111) = 111, 111%11 = 1 so insert in position 1.
4. hash (67) = 67, 67%11 = 1 so insert in position 1 -> BAD.
   1. collision (1)
   2. (67+1)%11 = 2 so insert in position 2.
5. hash (59) = 59, 59%11 = 4 so insert in position 4.
6. hash (37) = 37, 37%11 = 4 so insert in position 4 -> BAD.
   1. collision (2)
   2. (37+1)%11 = 5 so insert in position 5.
7. hash (35) = 35, 35%11 = 2 so insert in position 2 -> BAD.
   1. collision (3)
   2. (35+1)%11 = 3 so insert in position 3.
8. hash (25) = 25, 25%11 = 3 so insert in position 3 -> BAD.
   1. collision (4)
   2. (25+1)%11 = 4 so insert in position 4.
   3. collision (5)
   4. (26+1)%11 = 5 so insert in position 5.
   5. collision (6)
   6. (27+1)%11 = 6 so insert in position 6.
   7. collision (7)
   8. (28+1)%11 = 7 so insert in position 7.
   9. collision (8)
   10. (29+1)%11 = 8 so insert in position 8 so insert in position 8.
9. hash (5) = 5, 5%11 = 5 so insert in position 5.
   1. collision (9)
   2. (5+1)%11 = 6 so insert in position 6.
   3. collision (10)
   4. (6+1)%11 = 7 so insert in position 7.
   5. collision (11)
   6. (7+1)%11 = 8 so insert in position 8.
   7. collision (12)
   8. (8+1)%11 = 9 so insert in position 9 so insert in position 9.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|  | 111/222 | 67/134 | 35/70 | 59/118 | 37/74 | 138/276 | 414/828 | 25/50 | 5/10 |  |

*Storing Key Value Pairs*

To put these numbers takes 12 collisions.

To retrieve these number takes the same 12 collisions.

Thus, Linear Probing takes **24 collisions.**

print table.size()=11

index=1 key=111 value=222

index=2 key=67 value=134

index=3 key=35 value=70

index=4 key=59 value=118

index=5 key=37 value=74

index=6 key=138 value=276

index=7 key=414 value=828

index=8 key=25 value=50

index=9 key=5 value=10

\*\*\* Linear probing Descending Order End \*\*\*

**Quadratic Probing:**

1. hash (414) = 414, 414%11 = 7 so insert in position 7.
2. hash (138) = 138, 138%11 = 6 so insert in position 6.
3. hash (111) = 111, 111%11 = 1 so insert in position 1.
4. hash (67) = 67, 67%11 = 1 so insert in position 1 -> BAD.
   1. collision (1)
   2. (1+1\*1)%11 = 2 so insert in position 2.
5. hash (59) = 59, 59%11 = 4 so insert in position 4.
6. hash (37) = 37, 37%11 = 4 so insert in position 4 -> BAD.
   1. collision (2)
   2. (4+1\*1)%11 = 5 so insert in position 5.
7. hash (35) = 35, 35%11 = 2 so insert in position 2 -> BAD.
   1. collision (3)
   2. (2+1\*1)%11 = 3 so insert in position 3.
8. hash (25) = 25, 25%11 = 3 so insert in position 3 -> BAD.
   1. collision (4)
   2. (3+1\*1)%11 = 4.
   3. collision (5)
   4. (3+2\*2)%11 = 7.
   5. collision (6)
   6. (3+3\*3)%11 = 1.
   7. collision (7)
   8. (3+4\*4)%11 = 8 so insert in position 8.
9. hash (5) = 5, 5%11 = 5 so insert in position 5 -> BAD.
   1. collision (8)
   2. (5+1\*1)%11 = 6 so insert in position 6.
   3. collision (9)
   4. (5+2\*2)%11 = 9 so insert in position 9.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|  | 111/222 | 67/134 | 35/70 | 59/118 | 37/74 | 138/276 | 414/828 | 25/50 | 5/10 |  |

*Storing Key Value Pairs*

To put these numbers takes 9 collisions.

To retrieve these number takes the same 9 collisions.

Thus, Quadratic Probing takes **18 collisions.**

print table.size()=11

index=1 key=111 value=222

index=2 key=67 value=134

index=3 key=35 value=70

index=4 key=59 value=118

index=5 key=37 value=74

index=6 key=138 value=276

index=7 key=414 value=828

index=8 key=25 value=50

index=9 key=5 value=10

\*\*\* Quadratic probing Descending Order End \*\*\*

**Double Hashing Probing:**

1. hash (414) = 414, 414%11 = 5 so insert in position 7.
2. hash (138) = 138, 138%11 = 6 so insert in position 6.
3. hash (111) = 111, 111%11 = 1 so insert in position 1.
4. hash (67) = 67, 67%11 = 1 so insert in position 1 -> BAD.
   1. collision (1)
   2. Check (f(1,67)+1)%11, where f(1,67) is 1\*hash2(67) is 1\*( 7 – (67%7)) = 7- 4 is 3, so index= (3+1)%11 is 4, it is free, take the spot in position 4.
5. hash (59) = 59, 59%11 = 4 so insert in position 4 -> BAD.
   1. collision (2)
   2. Check (f(1,59)+4)%11, where f(1,59) is 1\*hash2(59) is 1\*( 7 – (59%7)) = 7- 3 is 4, so index= (4+4)%11 is 9, it is free, take the spot in position 8
6. hash (37) = 37, 37%11 = 4 so insert in position 4 -> BAD.
   1. collision (3)
   2. Check (f(1,37)+4)%11, where f(1,37) is 1\*hash2(37) is 1\*( 7 – (37%7)) = 7- 2 is 5, so index= (5+4)%11 is 9, it is free, take the spot in position 9
7. hash (35) = 35, 35%11 = 2 so insert in position 2.
8. hash (25) = 25, 25%11 = 3 so insert in position 3.
9. hash (5) = 5, 5%11 = 5 so insert in position 5.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|  | 111/222 | 35/70 | 25/50 | 67/134 | 5/10 | 138/276 | 414/828 | 59/118 | 37/74 |  |

*Storing Key Value Pairs*

To put these numbers takes 3 collisions.

To retrieve these number takes the same 3 collisions.

Thus, Double Hashing Probing takes **6 collisions.**

print table.size()=11

index=1 key=111 value=222

index=2 key=35 value=70

index=3 key=25 value=50

index=4 key=67 value=134

index=5 key=5 value=10

index=6 key=138 value=276

index=7 key=414 value=828

index=8 key=59 value=118

index=9 key=37 value=74

\*\*\* Double probing Descending Order End \*\*\*