1. Recurrence

Recurrence Relation:

Repeated Substitution:

1st expansion

1st Substitution:

2nd expansion

2nd Substitution:

Order of substitutions

From observing the pattern

Solving the geometric series using

Solving with Master Theorem:

Because ,

Since ,

1. Master Theorem:

Formula:

,

< 2, so

,

Since f(n) has a power of 1, and ,

,

Since f(n) is constant, is compared to 0.

Since ,

,

Since ,

So

,

Since

So,

1. Radix Sort

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | C | A | P |
| 1 | C | O | L |
| 2 | U | S | D |
| 3 | S | U | N |
| 4 | J | P | Y |
| 5 | V | E | E |
| 6 | R | O | W |
| 7 | J | O | B |
| 8 | C | O | X |
| 9 | L | O | L |
| 10 | R | A | T |
| 11 | W | O | W |
| 12 | D | O | D |
| 13 | C | A | R |
| 14 | F | I | G |
| 15 | P | I | G |
| 16 | V | I | S |
| 17 | L | O | W |
| 18 | L | O | X |
| 19 | V | E | A |
| 20 | C | A | D |
| 21 | D | O | G |
| 22 | T | S | L |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | V | E | A |
| 1 | J | O | B |
| 2 | U | S | D |
| 3 | D | O | D |
| 4 | C | A | D |
| 5 | V | E | E |
| 6 | F | I | G |
| 7 | P | I | G |
| 8 | D | O | G |
| 9 | C | O | L |
| 10 | L | O | L |
| 11 | T | S | L |
| 12 | S | U | N |
| 13 | C | A | P |
| 14 | C | A | R |
| 15 | V | I | S |
| 16 | R | A | T |
| 17 | R | O | W |
| 18 | W | O | W |
| 19 | L | O | W |
| 20 | C | O | X |
| 21 | L | O | X |
| 22 | J | P | Y |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | C | A | D |
| 1 | C | A | P |
| 2 | C | A | R |
| 3 | C | O | L |
| 4 | C | O | X |
| 5 | D | O | D |
| 6 | D | O | G |
| 7 | F | I | G |
| 8 | J | O | B |
| 9 | J | P | Y |
| 10 | L | O | L |
| 11 | L | O | W |
| 12 | L | O | X |
| 13 | P | I | G |
| 14 | R | A | T |
| 15 | R | O | W |
| 16 | S | U | N |
| 17 | T | S | L |
| 18 | U | S | D |
| 29 | V | E | A |
| 20 | V | E | E |
| 21 | V | I | S |
| 22 | W | O | W |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | C | A | D |
| 1 | C | A | P |
| 2 | C | A | R |
| 3 | R | A | T |
| 4 | V | E | A |
| 5 | V | E | E |
| 6 | F | I | G |
| 7 | P | I | G |
| 8 | V | I | S |
| 9 | J | O | B |
| 10 | D | O | D |
| 11 | D | O | G |
| 12 | C | O | L |
| 13 | L | O | L |
| 14 | R | O | W |
| 15 | W | O | W |
| 16 | L | O | W |
| 17 | C | O | X |
| 18 | L | O | X |
| 19 | J | P | Y |
| 20 | U | S | D |
| 21 | T | S | L |
| 22 | S | U | N |

1. Double Hashing

H1(25):

Home slot – 0.

H1(14):

Home slot – 4.

H1(9):

Home slot – 7.

H1(7):

Home slot – 12.

H1(5):

Home slot – 4 (1 collision).

H1(3):

Home slot – 10

H1(0):

Home slot – 0 (1 collision).

H1(21):

Home slot – 2.

H1(6):

Home slot – 8.

H1(33):

Home slot – 3.

H1(25):

Home slot – 0 (2 collisions).

H1(42):

Home slot – 10.

H1(24):

Home slot – 7.

H1(107):

Home slot – 6.

7. Complexity:  
  
My Radix Sort uses three methods: getMaxLength(), SortByIndex, and the actual Radix Sort.

The GetMaxLength method has a time complexity O(n), with n representing the number of elements in the array input. It’s space complexity is O(1), as it only stores constant space for the maximum length value of the array.  
  
For My SortByIndex method, The time complexity is O(N), as the number of iterations through some of my loops are constant regardless of input, such as the hashing an ArrayList to each letter and their case. It has a space complexity of O(N), as the buckets used in the sort have to store the values of all elements in the array.

For the Radix Sort, the time Complexity is O(N) for the best case, for if the maximum word length is 1, it will only iterate the SortByIndex method (O(N)) once. However, the worst case is O(N\*M) with M representing the length of the longest number, for if that length is more than one, the SortByIndex method will be iterated M times. The space complexity is O(N), as it only stores memory for the buckets (O(N)) and the maximum length.

So, overall, since all methods are being iterated in Radix Sort, the time complexity of my program is O(N) for the best case, O(N\*M) for the worst case. The space complexity is O(N).  
  
For my Word Pattern, the time and space complexity overall is O(N), with N representing the number of characters in the string. To separate all the words in the string, the algorithm had to iterate through each character, separating at the delimiter. This process uses O(N) time and will always iterate through each character. For the space complextity, the best case is that each letter in s is stored as a word, which would be in (O(N/2)).