Name: Yichen Zhang
Student ID: 1299406

Introduction:
My solution involves three data structures, sorted array and radix tree. I use typedef struct method called data_t to record those data The trading name is the primary identifier used for comparisons and searches.

```
typedef struct {
   int census_year, block_id, property_id, base_property_id;
   char* building_address
   char* clue_small_area;
   char* business_address;
   char* trading_name;
   int industry_code;
   char* industry_description;
   char* seating_type;
   int number_of_seats;
   double longitude, latitude;
} data_t;
```

Stage 2:
Test files were generated for size 1, 2, 10,20 1000 rows of data, the rows of data were random. The average was calculated for the number of comparisons needed across all trials for each size.

The method I use in stage2 is first make a empty array and then insert he data into the array, resize it when it is full then use binary search and linear search to find the key

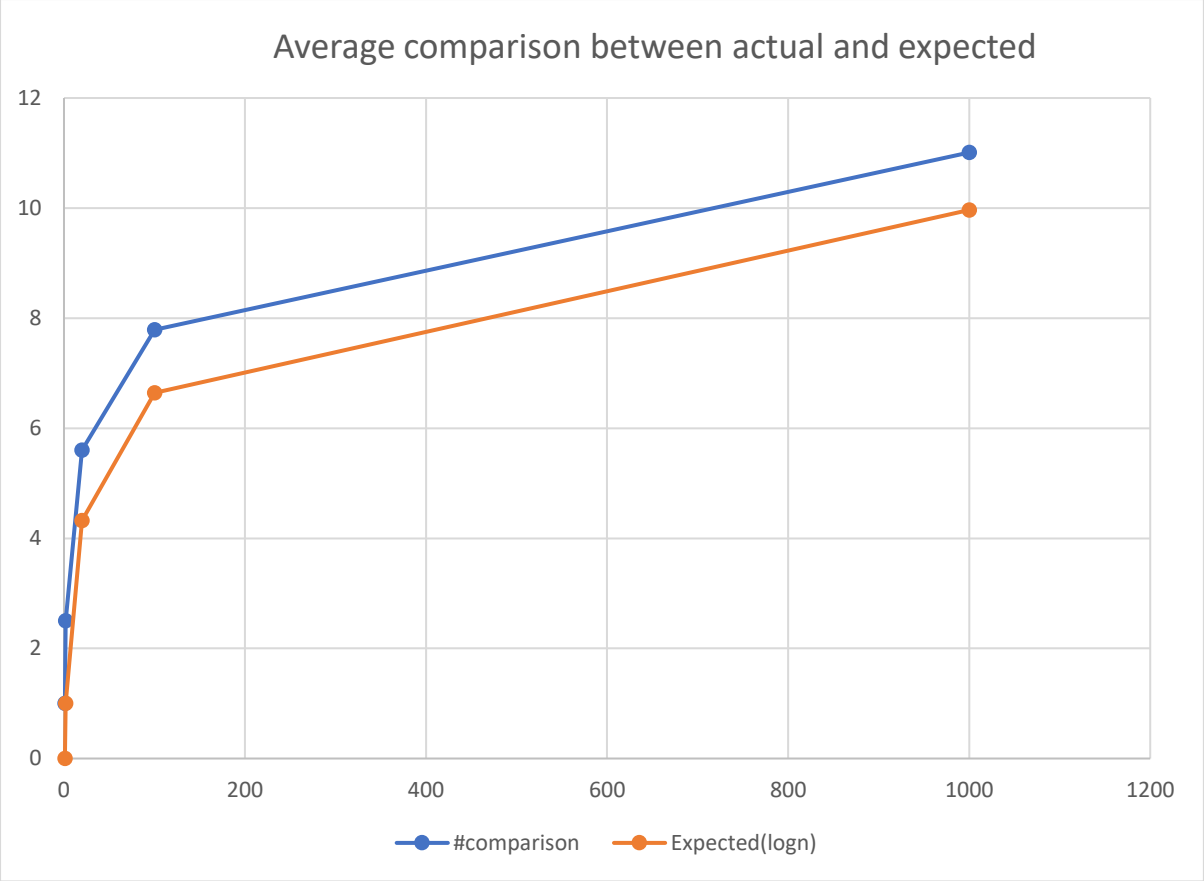
In my stage 2 I use binary search and linear search to do the comparison and find the key. The expected behaviour (Big O) is O(log n),n is the sample size. Here are the results obtained and result expected.

1/1 = 1 ;

(2+3)/2 = 2.5

…

Size	Average B	Average C	Average S	Expected
1	120	15	1	0
2	160	20	2.5	1
20	194	24.25	5.6	4.3219281
100	248.33	31.04	7.79	6.64385619
1000	351.60	43.95	11.01	9.96578429



In the graph, we can see that the difference between the actual I wrote, and the expected comparison is not too big, but we can see that when it is in small sample size, actual will cause more comparisons than expected. Also, obviously, when the input size increases the number of comparisons will increase as well.

For binary search the time complexity is O(logn), for linear search the time complexity is O(m). Also the best case of this code is O(1).

Stage 3:
Stage3 is basically on radix tree.

Size	#String comparison	Expected
1	1	112
2	1	140
20	1	122.8
100	1	135.9
1000	1	130.1

By looking at the difference between stage2 and stage3, we can see that every comparison between bit, char and str in stage 3 are all less than stage2. The time complexity for this is O(n), n is the length of length of bit, best cast, average case and worst case are all the same is O(n).
In conclusion of comparison, it is easily shown that radix tree method is better than stage 2(binary search and linear search method)

Discussion

My solution works properly and performs as expected and follows the Big O theory values. Obviously, stage3 method is better data structure for this question because it avoids repeating search.

This assignment gives me a deep learning on Big O time complexity and the RADIX TREE.