FIT2004 Assignment 1 2021
Name: Karel Mackenzie Chandra
Student Id: 30373867
Tutor/Lecturer : Lim Wern Han

1 Integer Radix Sort (9 marks)

1. Input
    a. nums is a unsorted list of non-negative integers

       Ans: The data type that is used for the list in the code is integer, it is a non-negative integer and is in unsorted form.

    b. b is an integer, with value ≥ 2

       Ans: Here the data type of b is also integer it is greater then or equal to 2 because here base 2 is a binary, there is no such base 1 or base 0.If it is base 1 then the elements inside list would become 0, if base 0 then all the elements inside list would become 1.

2. Output
    a. num_rad_sort returns a list of integers. This list will contain exactly the same elements as nums, but sorted into ascending numerical order.

       Ans: Here the data type of the return list would be integer same non-negative but this time the list is already sorted from smallest to largest

3. Big-O complexity
    a. num_rad_sort should run in $O((n + b) * \log_b M)$ time where
        i. n is the length of nums
        ii. b is the value of b
        iii. M is the numerical value of the maximum element of nums

       Ans: After running through the code it can be sure that the overall time complexity of radix sort is $O((n + b) * \log_b M)$, where $\log_b M$ is use in the radix sort to find the max element, b is the base that is going to be use for calculation and to loop it based on the nums of col. And O(n+b) is when calling the counting sort and n here is the length of num and b

is the base use for the calculation.

2 Timing bases (9 marks)

1. Input
    a. num_list is a list of non-negative integers

        Ans: The data type of num_list is integer and non-negative

    b. base_list is a list of integers, all with values ≥ 2, sorted ascending

        Ans: The data type of base is integer and it is greater than or equal to 2, as there is no such base 1 or base 0.If it is base 1 then the elements inside list would become 0, if base 0 then all the elements inside list would become 1

2. Output
    a. base_timer a list of numbers. Element i in this list is the time taken to run your radix sort from Task 1 on num_list using element i from base_list as the base.

        Since the actual runtimes will vary between students due to differences in implementation and hardware, there are no marks for the exact values of the times you obtain as output. In other words, just because two students obtain different outputs for the same input does not mean that one student has an error.

        The marks for this task come from the nature of the output, and your explanation of it (details in section 2.3)

3. Explanation
    a. In the explanation.pdf document, include 2 graphs. One comparing y1 and y2, with bases1 as the horizontal axis. The other compares y3 and y4, with bases 2 as the horizontal axis.

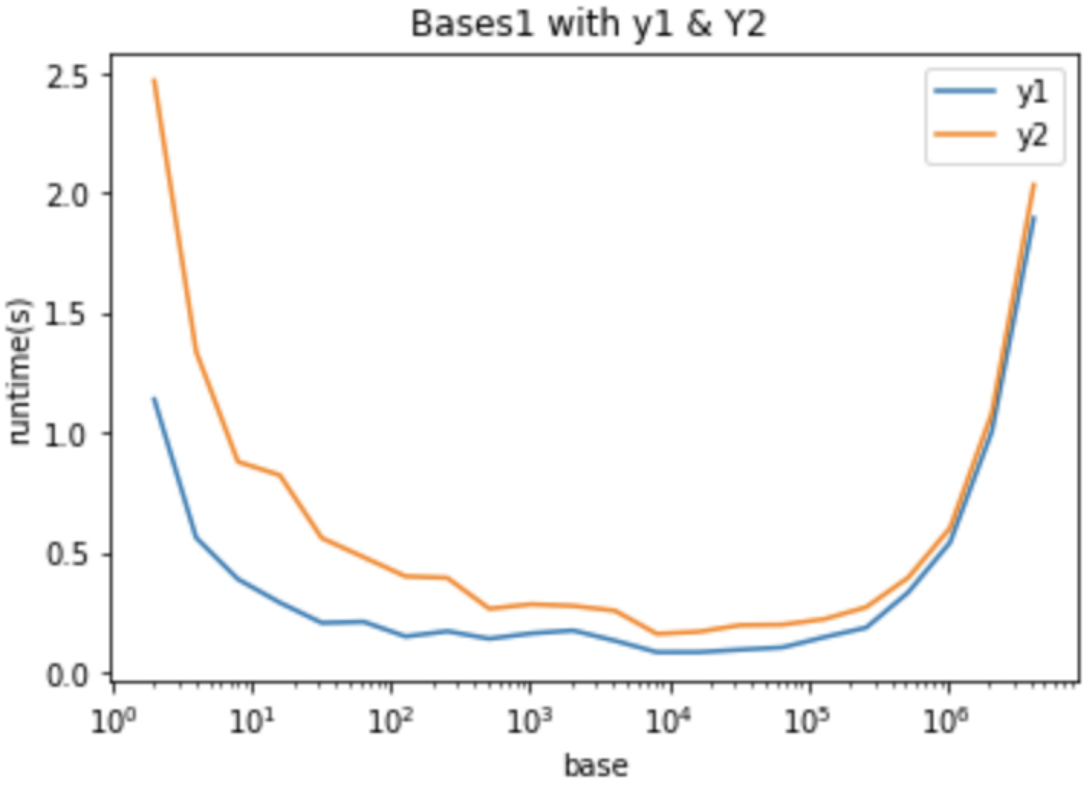
        For the first graph, you should use a logarithmic scale on your horizontal axis, but for the second graph you should use a linear scale.

        The vertical axis corresponds to the runtimes. This axis should not use a logarithmic scale. In the same pdf, answer the following questions:

        1. Why do the base/time curves for the first two graphs show a U shape? In other words, why are the times high when the base is low and when the base is high, but low when the base is in between? Justify your answer using the complexity of radix sort.

Ans: The time is high when the base is low because even though the count_array has less elements but the bucket per element is so large that iterating it through will take some amount of time. $O((n + b) * \log_b M)$ if we put it in the complexity of radix sort because of the b being so small it will not have a big effect on n and also since $\log_b M$ use b as it base then the log will not be very effective if the b is small.

The time is high also for a big base, is when the amount of count_array would be so large that it overcomes n which the length on list, the count will only filled not even half of the base and here $\log_b M$ will also become smaller meaning the overall value will be more then having a smaller base. If b is so large, one b value overcomes the whole complexity if it is so large that is why we can see that both edges are high as b is very small and b is very large.



2. Why are the times for y2 about twice as long as for y1 when the base is low? Include a mathematical argument based on the complexity of radix sort in your answer.

   Ans: y2 data is larger then y1 data by twice, and even though the base is small it won't affect the number of data with the

complexity of $O((n + b) * \log_b M)$ here we can see the n for y2 is equal to 2n of y1.

$Y2 = O((2n + b) * \log_b M)$

$Y1 = O((n + b) * \log_b M)$

$Y2 > Y1$ (when $b < 5$)

3. Why are the times for y2 not twice as long as for y1 (and in fact are very close) when the base is high? Include a mathematical argument based on the complexity of radix sort in your answer.

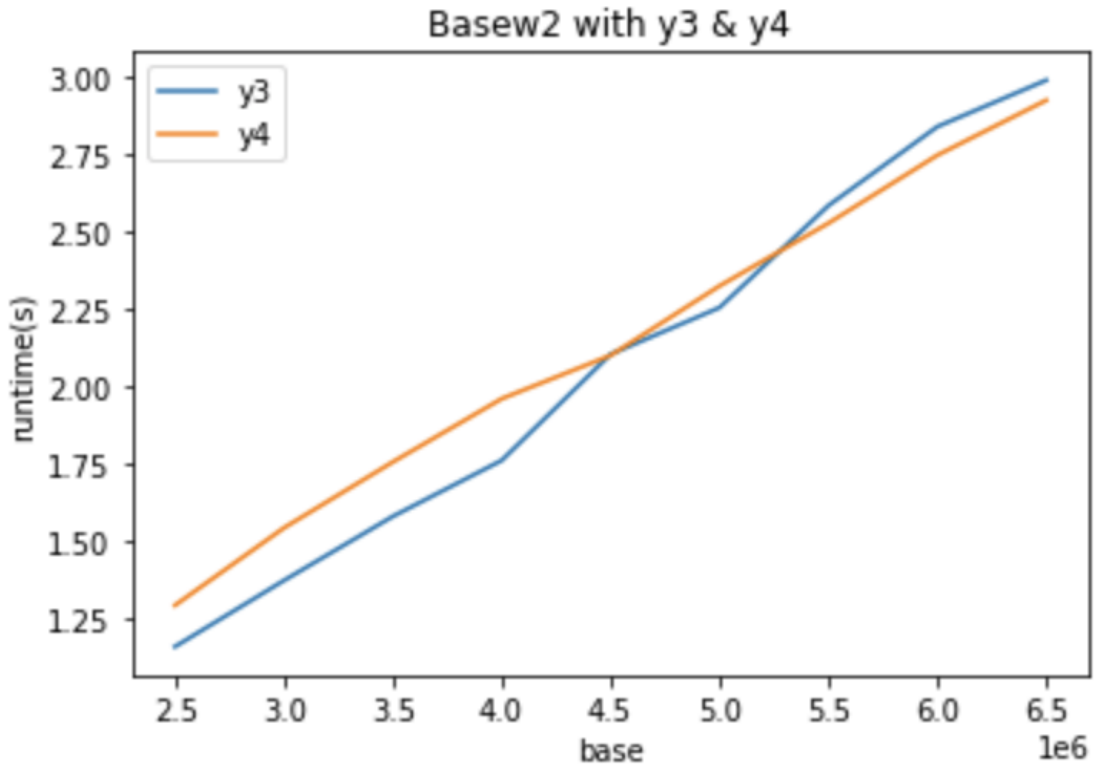
   Ans: In high bases we can see that b plays a role on the overall complexity of $O((n + b) * \log_b M)$ even though y2 = n is twice the amount of y1 = n. It will be overcome by the large base in (n+b) it will be mainly b and for the log of high bases it will be smaller in corresponding to M. So we have:

   $Y2 = O((2n + b) * \log_b M)$

   $Y1 = O((n + b) * \log_b M)$

   $Y2 = Y1$ (when $b > 5$)

   The equation will mainly be affected by the high base that is why It is closer to each other in high bases



.

4. Why are the times for y3 and y4 almost the same, despite data2 having twice as many elements as data1? Include a mathematical argument based on the complexity of radix sort in your answer.

   Ans: This is because the graph only shows the relation of time and base meaning even though the data is different, the differences is not really significant the relationship shows that as the base grows the runtime grows following the mathematical equation of the graph we can see that it is Y = mx + c. Meaning in this case As y grows x grows too m in this case is the difference c is the data set. Hence from Y = mx + c we can see that the runtime and base is direct proportion

5. Why do the graphs for y3 and y4 show an almost linear shape? Include a mathematical argument based on the complexity of radix sort in your answer.

   Ans: A linear shape meaning the equation is Y = mx+c here the x axis is the base and y axis is the run time, the correlation of

both is, as the base increases the time taken is also increased meaning the time in this graph is greatly dependent on the base. The number of data here won't really affect as time is dependent on the base. We can conclude that even though y3 and y4 have different amounts of data, it does really affect a linear graph as it is for sure the time is greatly dependent on the base. Hence from Y = mx + c we can see that the runtime and base is direct proportion

4. Complexity
    a. The overall complexity for this task is not important.

3 Interest Groups (9 marks)

Consider a database consisting of people, each of whom likes a set of things. We want to create groups of people with identical interests.

To do this, you will write a function interest_groups(data).

3.1 Input

data is a list, where each element is a 2-element tuple representing a person. The first element is their name, which is a nonempty string of lowercase a-z with no spaces or punctuation. Every name in the list is unique.

The second element is a nonempty list of nonempty strings, which represents the things this person likes. The strings consist of lowercase a-z and also spaces (i.e. they can be multiple words) but no other characters. This list is in no particular order.

Ans: The data type of both list, list of names and like things is a str, to use radix sort we must find the corresponding ascii value. Then once found we can sort through the tuple.

3.2 Output

interest_groups returns a list of lists. For each distinct set of liked things, there is a list which contains all the names of the people who like exactly those things. Within each list, the names should appear in ascending alphabetical order. The lists may appear in any order.

Example:

In = data = [("nuka", ["birds", "napping"]),

("hadley", ["napping birds", "nash equilibria"]),

("yaffe", ["rainy evenings", "the colour red", "birds"]),

("laurie", ["napping", "birds"]),

("kamalani", ["birds", "rainy evenings", "the colour red"])]

>>> interest_groups(data)

Out = [["laurie", "nuka"], ["hadley"], ["kamalani", "yaffe"]]

Ans: The output is also a tuple but only the names that is group together based on what interest they have. The data type is str.

3.3 Complexity

Input constraint: If we call the number of strings in any list of liked things X and the length of the longest string in that list Y, then XY is O(M)

interest_groups must run in O(NM)

• N is the number of elements in data

• M is the maximum number of characters among all sets of liked things. You may assume that all names are also shorter than M characters.

In the example above, N = 5 (there are 5 people), and M = 33, since there are 33 characters in the sets belonging to yaffe and kamalani, and they are the longest sets.

Ans: After checking through the function we can say that the Big-O is OINM), where n is the len of data given, M is a multiplication of the strings inside the like list with the longest string value. Note that there might be sum nested loops to access the element. The complexity of ord(_) is O(1) so it keeps