Name: Mohan REN

This file has been provided to help you structure your thoughts when discussing Lab 3, Part c with your marker. Some questions can be answered with a single sentence, some may require much longer answers. You are free to edit/rearrange this file as much as you want.

All questions should be answered for each of your chosen data structures.

Initial Expectations

>> Do you expect this data structure to be preferable to the others on all inputs, most inputs, some inputs? Why?
By choosing the hashset with open addressing, firstly, we assume that the number of the inserted value is less than the hashtable's size which means that there is no rehashing occurs or the result will be quite different. Then for giving all inputs generated randomly, compared with sorted dynamic array using binary search and the BST, the hashset should have a better performance. Because the average complexities of hashset with open addressing, dynamic array using binary search and BST are O(1), O(logn), O(logn) respectively.
(Clarify: inputs include the dict and the infile)

>> Do you expect your answer to change if the order of the words in your input dictionary is in the best/worst case? Why?
Complexity on Best case:
Dynamic array with binary search: O(1)
Hashset with open address: O(1)
BST: O(log n)

Complexity on Worst case:
Dynamic array with binary search: O(log n)
Hashset with open address: O(n)
BST: O(n)

Yes the answer will changes as the time complexity of these three data structures on difference cases is different. On best case, dynamic array and hashset has the same complexity but sometimes in the real, there are still some difference for the time really spent. Besides, the best cases for hashset and dynamic array are different: for hashset, the best case is that all value is not cluster and hash enough; for dynamic array is that all inserted values are sorted. On the worst case, although the complexity of hashset and BST are same semantically, the worst cases for them are still different: for hashset is that all values are

extremely clustered and for BST is that all values are sorted which means that all nodes are on the same side of the tree.

>> Can you phrase what you expect in terms of a one or two sentence hypothesis that you can test?
1. When the input is random generated, the hashset with open addressing data structure is preferable.
2. When the input is sorted, the dynamic array using binary search is preferable.

Experimental Design

>> How are you going to define what it means to one data structure to be preferable to another?
Now, I just define that time spent will decide which data structure is preferable. Memory cost is also important, but I will only talk about the time here.

>> Which conditions will you vary in your experiment?
So in my experiment, I will control the order of the input: sorted or not sorted (which is uniformly random)

>> How will you vary them? Why did you make these choices? Did you use theoretical complexities, best, worst and average cases to inform your decisions?
To vary them, I can write some data sets by hand or use a generator script to produce the data sets.
Here I used the theory of best case, average case and worst case: The average case is that all order of the data input is uniformly random and that is also the average case for all of three data structures. Also the time complexity of darray, hashset and BST are O(logn), O(1), O(logn) respectively.
Then I will make a sorted input and that is the best case for dynamic array (complexity O(1)) but the worst case for BST (complexity O(n)).

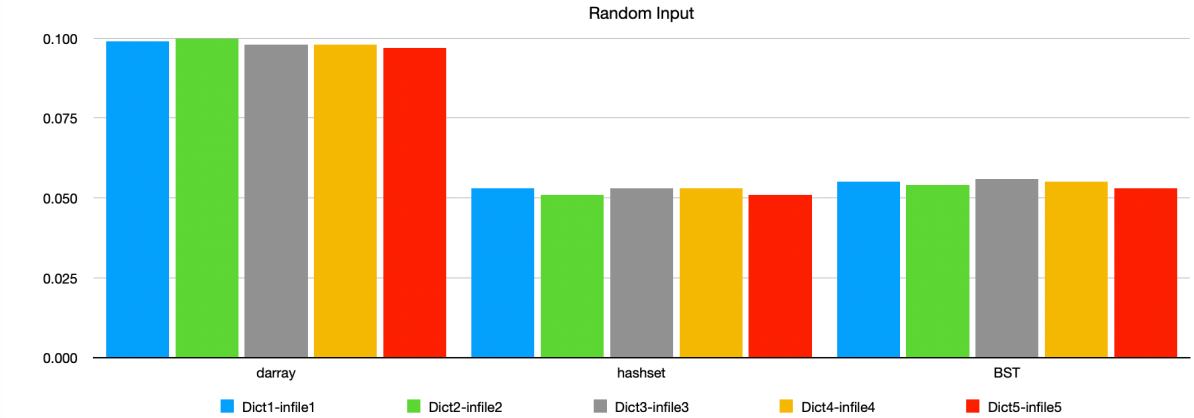
>> How will you generate the data for your experiments?
Use the generate.py given in the data repo and use command:
1. sh generate.sh large/henry/dict dict infile x y random z
2. sh generate.sh large/henry/dict dict infile x y random z

where x is the total number of the value in dict, y is the number of the value in infile and z is the percentage of infiles that should be in the dictionary.
At least 5 inputs will be generated respectively to make the results faithful.
Also, the size of the dict is 1000, 500 for infile and the percentage is 20. I hope this size is huge enough to prove my finding.

>> How will you validate your findings?
I generated 5 pairs of input (dict and infile) based on the large dict in henry for random proving, and same thing for sorted proving. Then I time the process for each of three data structures on each pair for 3 times. Then I average the three results, and it will be the time cost for each data structure on each kind of inputs. Then, for each data structure, I average the time spent on total 5 inputs to compare which one is the fastest in different cases.

Results and Analysis

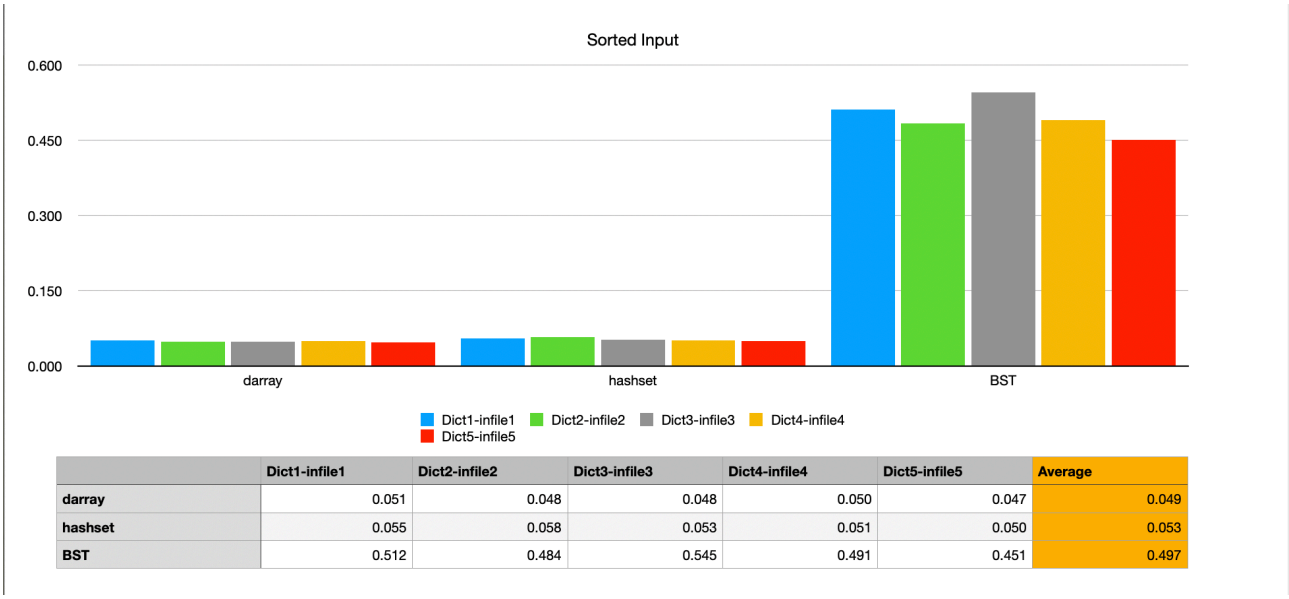
>> What results did you record?
Time spent for different data structure when input is random



	Dict1-infile1	Dict2-infile2	Dict3-infile3	Dict4-infile4	Dict5-infile5	Average
darray	0.099	0.100	0.098	0.098	0.097	0.098
hashset	0.053	0.051	0.053	0.053	0.051	0.052
BST	0.055	0.054	0.056	0.055	0.053	0.055

For this case, we see that the average time spent by darray, hashset and BST are 0.098s, 0.052s and 0.055s respectively.

Time spent for different data structure when input is sorted



	Dict1-infile1	Dict2-infile2	Dict3-infile3	Dict4-infile4	Dict5-infile5	Average
darray	0.051	0.048	0.048	0.050	0.047	0.049
hashset	0.055	0.058	0.053	0.051	0.050	0.053
BST	0.512	0.484	0.545	0.491	0.451	0.497

For this case, we can see that the average time spent by darray, hashset and BST are 0.049s, 0.053s and 0.497s respectively.

>> What does this tell you about the performance of the data structure?
By viewing the result given above, it shows that the average time spent by three data structures only have slight difference. For the input generated randomly, the hashset with open addressing is the fastest although only 0.003 second faster than the BST but that is because the size of the input is not huge.
Then for the sorted input which is the best case for dynamic array with binary searching but the worst case for the BST. In the results given above for sorted input, we see that array has the best performance as the average time spent by it is the shortest as in this case, the time complexity is O(1). Hashset has a similar time spent as the random input and this is the same as my expectation for it. One thing deserve our notice is that the average time spent by BST is much longer than others as all value were added on the same side which also cause the worst case for it.

>> What is the answer to the question "Under what conditions is it preferable to use this data structure?”
As the results are good for my expectation, I can say that:
1. When the inputs are random, hashset with open addressing is preferable;
2. When the inputs are sorted, dynamic array with binary searching is preferable.

However, I haven't strongly validated these conclusion but this is what I extrapolate from the data.