Algorithm and analysis

Assignment 1

We (I) certify that this is all our (my) own original work. If we (I) took any parts from elsewhere, then they were non-essential parts of the assignment, and they are clearly attributed in my submission. We (I) will show we (I) agree to this honour code by typing "Yes": Yes.

Yuxin Ye | s3737548

Ching Yau Loo | s3557584

Experimental Setup

For this experiment we are testing and comparing two different binary tree representations, linked and sequential in terms of their time complexities. Three different complexities of the binary tree were generated for both representations for this experiment. Specifically, the complexity here means the size of the data.

The data size ratio of 100: 1000: 100000 (10^2:10^3:10^5) nodes were used for this experiment. The main reason for selecting this ratio is to clearly compare the performance of the two representations under exponentially different input sizes. The data generation utilised Data Generation code provided in Labotory 2. Minor edit was performed to produce the Binary Space Partitioning (BSP) format and text files of each data size. The following scenarios were decided to evaluation the performance of the two representations: 1) Grow BSP tree. 2) Finding a Node, its Parent Node and Children Node. 3) Printing the Node. These scenarios were decided based on the possible BSP usage.

The timing for each scenario was measured as the running time of methods. NanoTime() method was used to calculate run time in nanoseconds, then converted to seconds. Each scenario was ran 3 times with the average taken as analysis data.

The experiments were conducted on Titan server provided by RMIT university.

Evaluation

Scenario 1 Growing BSP Tree (node splitting)

SplitNode method was used in this scenario under the three different data size. Each run time and their averages are recorded as below.

Test Number	Linked Representation (Seconds)	Sequential Representation (Seconds)
1	0.0034957	0.0013930
2	0.0028208	0.0014611
3	0.0028158	0.0014989
Average	0.0032354	0.0013930

Table 1. 100 Nodes

Test Number	Linked Representation (Seconds)	Sequential Representation (Seconds)
1	0.0109242	0.0060153
2	0.0100521	0.0062749
3	0.0106413	0.0061588
Average	0.0109710	0.0057806

Table 2. 1000 Nodes

Test Number	Linked Representation (Seconds)	Sequential Representation (Seconds)
1	33.3157801	7.3288696
2	22.5568865	7.4058424

3	26.9916273	7.6823113
Average	27.13885812	7.5224417

Table 3. 100,000 Nodes

From the results, we found the sequential representation performed node splitting faster than linked representation. The contrasts were more distinct when the data size increased to 100,000.

We hypothesis the reason for this is that, in linked representation, three operations were performed in splitNode method: traverse the whole tree to validate if the source node exists, traverse the whole tree to find the source Node, and create new Node class for left and right children. In the case of sequential representation, three operations were also performed: traverse the array to find the source node, dynamically grow the array, and insert left and right children. However, sequential representation only had to traverse the array once with time complexity of O(n), linked representation had to traverse the tree twice.

From the theoretical and empirical analysis above, sequential representation for binary tree is more efficient in the cases of growing BSP tree.

Scenario 2 Finding a Node and Its Parent Node and Children Nodes

This scenario tested the time taken for finding a node, its parent node and its children nodes. The study used the worst-case scenario for both representations (finding the node at the end of input file). 3 input sizes were used.

Test Number	Linked Representation (Seconds)			Sequential Representation (Seconds)		
	Find Node	Find Parent	Find Children	Find Node	Find Parent	Find Children
1	0.0000130	0.0000379	0.0000134	0.0000039	0.0000255	0.0000063
2	0.0000128	0.0000456	0.0000132	0.0000039	0.0000264	0.0000063
3	0.0000096	0.0000397	0.0000126	0.0000038	0.0000252	0.0000065
Average	0.0000118	0.0000411	0.0000131	0.0000039	0.0000257	0.0000064

Table 4. 100 Nodes

Test Number	Linked Representation (Seconds)			Sequential Representation (Seconds)		
	Find Node	Find Parent	Find Children	Find Node	Find Parent	Find Children
1	0.0000161	0.0000359	0.0000258	0.0001792	0.0000639	0.0000287
2	0.0000115	0.0000373	0.0000114	0.0001683	0.0000650	0.0000278
3	0.0000123	0.0000375	0.0000114	0.0001904	0.0000646	0.0000281
Average	0.0000133	0.0000369	0.0000162	0.0001793	0.0000645	0.0000282

Table 5. 1,000 Nodes

Test Number	Linked Representation (Seconds)			Sequential Representation (Seconds)		
	Find Node	Find Parent	Find Children	Find Node	Find Parent	Find Children
1	0.0017351	0.0037897	0.0032094	0.0052419	0.0047637	0.0022394
2	0.0016534	0.0039999	0.0024743	0.0053423	0.0049229	0.0025135
3	0.0017260	0.0035905	0.0029158	0.0090770	0.0049301	0.0022050
Average	0.0017048	0.0037934	0.0028665	0.0065537	0.0048722	0.0023193

Table 6. 100,000 Nodes

		FN	FP	FC
100	Linked	0.0000118	0.0000411	0.0000131
	Sequential	0.0000039	0.0000257	0.0000064
1,000	Linked	0.0000133	0.0000369	0.0000162
	Sequential	0.0001793	0.0000645	0.0000282
100,000	Linked	0.0017048	0.0037934	0.0028665
	Sequential	0.0065537	0.0048722	0.0023193

Table 7. Comparison

Mixed results are shown between the two tree representations.

Find Node – Linked representations appears to be faster than sequential representation in the data size of 1,000 and 100,000, but slower in the data size of 100.

Find Parent – Same observation as above occurred.

Find Children – Linked representation appears to be slightly faster than sequential representation for the data size of 1,000. For 100 and 100,000 nodes sequential representation is overall faster than linked representation even though during the second run linked representation is slightly faster than sequential representation.

We hypothesise the reason for this is that when perform findNode method, sequential representation loop through the whole array until the node was found while linked representation traverse the tree until the node was found, if the tree was in a balanced structure, the time taken to find the node is less than O(n) for Linked representation in a large data set. When perform findParent method, the same principle happens, sequential representation also needs to perform extra operations to calculate the array index of parent node. When perform findChildren method, both methods need to traverse the whole data set, therefore linked and sequential representation performs in a similar manner.

Scenario 3 Printing the nodes (Traversal)

3 types of the traversals: pre order, post order and in order were tested under the 3 different data sizes.

Test Number	Linked Representation (Seconds)			Sequential Representation (Seconds)		
	Pre order	In order	Post order	Pre order	In order	Post order
1	0.0002628	0.0001770	0.0001267	0.0009506	0.0009081	0.0011765
2	0.0002666	0.0001799	0.0001255	0.0009324	0.0009223	0.0010855
3	0.0002671	0.0001834	0.0001260	0.0009734	0.0013722	0.0010805
Avg	0.0002655	0.0002655	0.0001260	0.0009521	0.0010675	0.0011141

Table 8. 100 Nodes

Test Number	Linked Representation (Seconds)			Sequential Representation (Seconds)		
	Pre order	In order	Post order	Pre order	In order	Post order
1	0.00128226	0.00116432	0.001310476	0.0014915	0.0013838	0.0014218
2	0.00136768	0.00157728	0.001276239	0.0014761	0.0014135	0.0013555
3	0.00127494	0.00117323	0.001356931	0.0014441	0.0014108	0.0013926
Avg	0.00130830	0.00130494	0.001314549	0.00147056	0.0014027	0.00138996

Table 9. 1,000 Nodes

Test Number	Linked Representation (Seconds)			Sequential Representation (Seconds)		
	Pre order	In order	Post order	Pre order	In order	Post order
1	0.04269348	0.05006091	0.05358513	0.7370581	1.5492877	1.5534279
2	0.04409995	0.04801680	0.05205743	1.1592936	1.8754962	1.1824755
3	0.04500611	0.04873886	0.05268974	1.1736845	1.1409800	1.8766952
Avg	0.04393318	0.04893885	0.05277743	1.0233454	1.5219213	1.53753286

Table 10. 100,000 Nodes

		Pre order	In order	Post order
100	Linked	0.0002655	0.0002655	0.0001260
	Sequential	0.0009521	0.0010675	0.0011141
1,000	Linked	0.0013083	0.0013049	0.0013145
	Sequential	0.0014706	0.0014027	0.0013900
100,000	Linked	0.0439332	0.0489388	0.0527774
	Sequential	1.0233454	1.5219213	1.5375329

Table 11. Comparison

The results show that the overall performance of the three traversals for linked representation is faster than sequential representation.

We hypothesize the reason for this is because both linked and sequential representation would traverse the whole tree to print nodes, but sequential representation perform extra operations such as using the formula (2 * i + 1) and (2 * i + 2) to calculate the positions of the nodes. Hence slowing down the overall performance.

Recommendation

Both tree representations have their own merits. For growing the tree (setting and splitting nodes) sequential representation is much faster compare to linked representation with the average difference of 0.0018 seconds for 100 nodes, 0.0051 seconds for 1k nodes and 19.61 seconds for 100k nodes. For finding nodes sequential is faster than linked representation for smaller data size of 100 nodes with and average difference of 0.0000079 seconds for find node, 0.0000154 seconds for find parent and 0.0000067 seconds. But for larger data sizes of 1k nodes and 100k nodes linked representation is roughly 2 times faster than sequential representation. Lastly for traversals linked representation seems to be overall faster

comparing to sequential for all the traversals. For data size 100 nodes and 100k nodes linked representation is roughly 2 times faster than sequential but for 1k nodes both representations perform almost similar with linked representation to be slightly faster.

As the result shown each of the representation has their own merits and we recommend sequential representation for growing (setting and splitting) the tree since the time differences was very distinct between the two representations. For searching and traversing we recommend linked representation since it has the most stable performance across all data sizes compared to sequential representation.

Task C

Question C1

	O(1)	O(log(n))	O(n)	O(n!)
Find Parent	No	No	Yes	No
Find a node	No	No	Yes	No
Print all nodes	No	No	Yes	No

Linked representation has the following time complexity for the three operations (worst case):

$$C(n) = \sum_{i=0}^{n} \mathrm{i} \quad \in \quad \mathrm{O(n)}$$

Sequential representation has the same time complexity as above for finding a node, but the following time complexity for finding parent and printing all nodes operations, where one is the extra operation that calculates the array index of nodes (worst case):

$$C(n) = \sum_{i=0}^{n} \mathrm{i} + 1 \quad \in \quad \mathrm{O(n)}$$

Both representations belong to the O(n) complexity class for all three operations.

Question C2

Strategy 1:

Time complexity for worst case:

$$C(n) = \sum_{i=0}^{n} \mathrm{i}$$

On average case:

$$C(n) = \frac{(n+1)}{2}$$

The astronaut had to ask 14 billion times for the worst case before he can have food. 7 billion times on average case.

Strategy 2:

The astronaut uses binary search strategy to approach the question. The time complexity for binary search on average case is:

$$\frac{1+2+3+\cdots+34}{34} = 18$$

On worst case is $O(\log_2(n))$.

$$\log_2 14{,}000{,}000{,}000 \approx 34$$

The astronaut only need to ask about 18 times on average case and 34 times on worst case.