Report for Algorithms & Analysis Assignment 1

Cheng Chen s3728207

Bowen Zhang s3617571

|  |
| --- |
| 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 honor code by typing "Yes": Yes. |

**Experimental Setup**

-Data Scenarios

Scenario 1: Split Node - Ratio: 10

Split Node operation has a bigger difference in the two structures so smaller ratio would suffice.

Scenario 2: Find Node - Ratio: 100

Find Node process complexity is quite similar, so we need larger ratio to see the real difference if there is any.

Scenario 3: Print Node - Ratio: 100

Print Node process complexity is quite similar, so we need larger ratio to see the real difference if there is any.

\*\*\*\*\*!!!!!!!-Sizes of data (graphs; graph density) – why did you select these sizes to test? E.g., [Sample text] We used these sizes because we this size (X) allow us to contrast the performance for Y operation for the A and B data structures.

-Generation of scenarios

We wrote code to generate data from the supplied data set for different scenarios.

-Timing

We did consecutive runs for each test to get the average time.

We did 10 for each test to generate the data.

**Evaluation**

Scenario 1: Split Node

[Sample text] We found that … (see Figure 1). We hypothesise the reason for this is that…, but…. Compare this with the incidence matrix representation … (Figure 2).

Scenario 2: Find Node

[Sample text] As we varied the densities of the, we found ...

Scenario 3: Print Node

**Recommendation**

For Split Node:

We recommend using

For Find Node:

We recommend using

For Print Node:

We recommend using

/\*For different data scenarios (pick a representative sample from your analysis), which data structures do you recommend using for which cases?\*/

**Part C**

C1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | O(1) | O(log(n)) | O(n) | O(n!) |
| Find parent | No | No | Yes | Yes |
| Find a node | No | No | Yes | Yes |
| Print all nodes | No | No | Yes | Yes |

Find parent:

Basic operation: comparison

Input size: n - 1

C(N) = = n - 1 ∈ O(n)

Find a node:

Basic operation: Compare value

Input size: n

C(N) = = n ∈ O(n)

Print all nodes:

Basic operation: Comparison

Input size: n

C(N) = = 4n ∈ Ω(n)

n! is larger than n so they also belong to O(n!)

C2.

1. In the worst case, he needs to ask N questions.

In the average case:

Cavg = 1\*1/N + 2\*1/N+…….+N\*1/N

=(1+2+3……+N)\*1/N

=(1+N)\*N/2N

=(1+N)/2

he needs to ask (1+N)/2 questions.

1 second each question:

worst case: 14 billion seconds which is 443.937 years

average case: (1+14 billion)/2 = 7000000000.5 seconds which is 221.969 years

2. In the worst case:

Assume the question is only “at most” and the answer is only yes, or no. assume the number of question is n. the worst case occurs when |n.value – (n – 1).value| == 1 and |n.value – (n – 2).value| == 1. The basic operation is comparison, aka question

C(N) = C(N/2) + 1 and C(1) = 1

Since C(N) = C(N \* 2 ^ -1) + 1 = C(N \* 2 ^ -2) + 2;

Assume N = 2 ^ k;

C(2k) = C(2k – 1) + 1 for k > 0;

C(20) = 0;

C(2k) = C(2k-1) + 1 substitute C(2k - 1) = C(2k - 2) + 1

= C(2k - 2) + 2 = C(2k - 3) + 3 = … = C(2k-i) + I = … = C(2k-k) + k

Therefore C(2k) = C(1) + k = k + 1

since n = 2k, k = log2 n, C(N) = log2 N + 1;

So in the worst case, he needs to ask (log2N + 1) questions

In the average case:

Cavg = 1\*1/(log2N + 1) + 2\*1/(log2N + 1) + …… + (log2N + 1)\*1/(log2N + 1)

=(log2N + 2)/2

He needs to ask (log2N + 2)/2 questions

If N = 14 billion:

Worst case: 34.7 which is 35 questions

Average case: 17.85 questions