Author: Jacob MacDermaid

Date:8/24/2017

Program: DfsSort.java

Theoretical worst-case running time is Θ(n2). The reason for this is the algorithm would need to scan each edge for every vertex in the worst case, so it would perform n2 checks.

The following are average run times and count for 5,6,7, and 10 nodes.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nodes | ns | ns | ns | ns | ns | avg(ns) |
| 5 | 24803 | 24804 | 23948 | 24804 | 24804 | 24632.6 |
| 6 | 28653 | 26086 | 42338 | 26514 | 26942 | 30106.6 |
| 7 | 46614 | 30791 | 29508 | 28653 | 28225 | 32758.2 |
| 10 | 42337 | 41482 | 64148 | 41055 | 41482 | 46100.8 |

**Table 1: Run time for DFS sort**

|  |  |
| --- | --- |
| Nodes | Count |
| 5 | 35 |
| 6 | 47 |
| 7 | 70 |
| 10 | 130 |

**Table 2: Count for DFS**

Looking at figure 1, which is the count results, it proves that the class of efficiency is Θ(n2), using the line of best fit.

**Figure 1: DFS (counts)**

--------------------------------------------------------------------------------------------

Program: SourceRemoval.java

Theoretical worst-case running time is Θ(n3). The reason for this is in the worst case the algorithm finds the source at the end of the matrix. This requires n2 checks. Once this row and column are removed you are left with (n-1)2 and then (n-2)2 checks and so on. This leads to 12+22+32+...+n2 = n(n+1)(2n+1)/6. From that it is found this algorithm runs in Θ(n3).

The following are average run times and count for 5,6,7, and 10 nodes.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nodes | ns | ns | ns | ns | ns | avg(ns) |
| 5 | 13258 | 20527 | 16679 | 15823 | 12830 | 15823.4 |
| 6 | 11974 | 19672 | 20954 | 16679 | 11974 | 16250.6 |
| 7 | 17106 | 17106 | 20099 | 19244 | 16251 | 17961.2 |
| 10 | 26087 | 28226 | 32074 | 41482 | 47042 | 34982.2 |

**Table 1: Run time for Source removal sort**

|  |  |
| --- | --- |
| Nodes | Count |
| 5 | 25 |
| 6 | 36 |
| 7 | 55 |
| 10 | 132 |

**Table 2: Count for source removal**

Looking at the figure 2, which the graph of the count results it proves that the class of efficiency is Θ(n3), using the line of best fit.

**Figure 2: Source Removal (counts)**

The faster algorithm is source removal when number of nodes is less than 10. If you look at figure 3 of the counts for these two algorithms you can see that source removal starts off with a smaller count and takes less time, if you compare run times from table 1 and table 3. However, as number of nodes increases the difference becomes less and less until number of nodes hits 10. Then the count becomes greater for source removal. So, dfs sort becomes the faster algorithm. This is backed up by the theoretical worst-case because dfs has a better class of efficiency than source removal.

**Figure 3: DFS vs Source Removal (counts)**