# BINARY SEARCH TREES AND ORDER-STATISTIC TREES

**School of Computer Science & Applied Mathematics**
**University of the Witwatersrand**

**Musawenkosi Gumpu**
**2326254**

A Comparative Analysis of Insertion and Deletion Methods in Binary Search Trees (BSTs)

# Contents

**Preface**
Table of Contents . . . i
List of Figures . . . ii

**1 Introduction** **2**

**2 Objective** **3**

**3 Methodology** **4**
3.1 Experimental Setup . . . 4
3.1.1 Range of Dimensions & Key Values . . . 4
3.1.2 Number of Trees . . . 4

**4 Analysis of BST** **5**
4.1 Expected Height . . . 5
4.1.1 Average Insertion Runtime . . . 5
4.1.2 Tree Destruction Runtime . . . 5

**5 Improvement to BST: The Size Attribute** **7**
5.1 Maintaining Node Sizes on Key Insertion . . . 7
5.1.1 Size Update of Inserted Node After Insert . . . 8
5.2 Maintaining Node Sizes on Key Deletion . . . 8
5.2.1 Size Update of Deleted node Before Deletion . . . 8

**6 Conclusion** **9**

**A Appendix** **10**
A.1 TREE-INSERT . . . 10
A.2 TREE-DELETE . . . 11
Tabular Results . . . 11
A.3 Tabular Results . . . 12

**References** **24**

# List of Figures


1 Comparison of input size and expected build time. . . . . . . . . . . . . ii
2 Comparison of input size and expected tree height. . . . . . . . . . . . . 1
3 Comparison of input size and expected destroy time. . . . . . . . . . . . 1


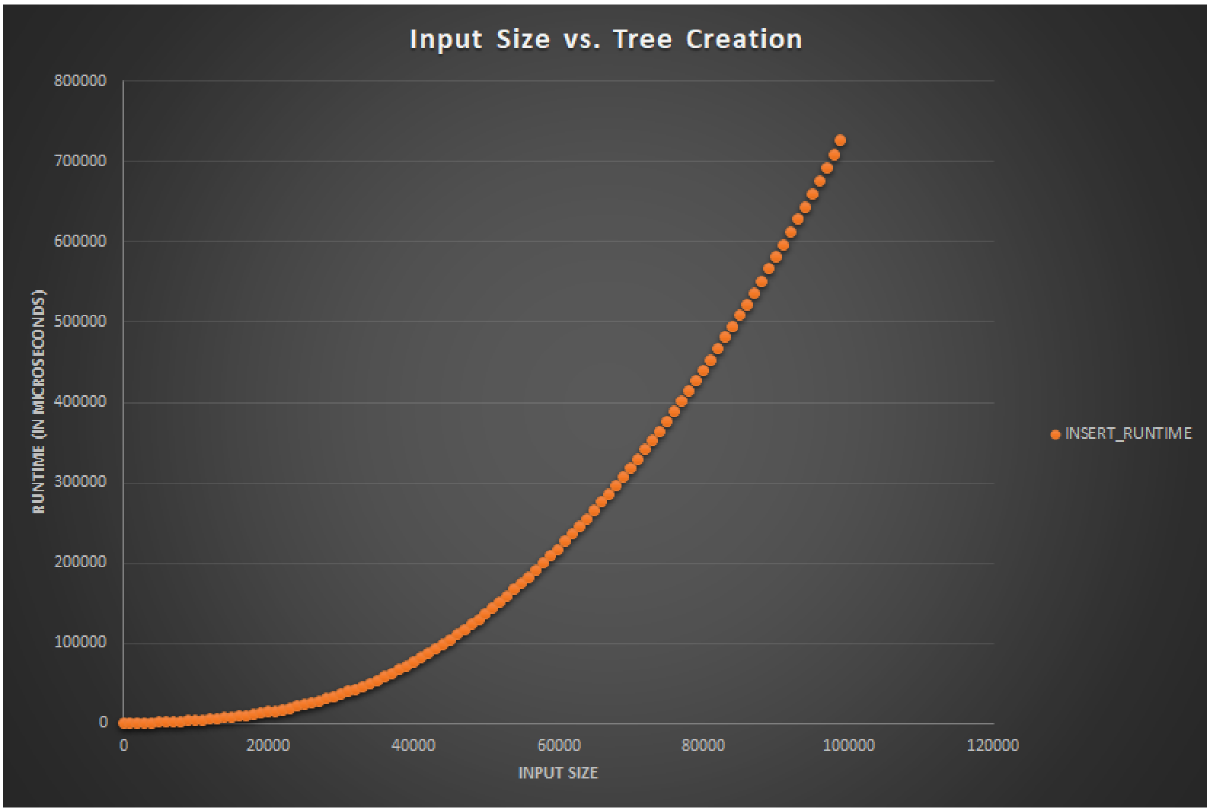


Figure 1: Comparison of input size and expected build time.

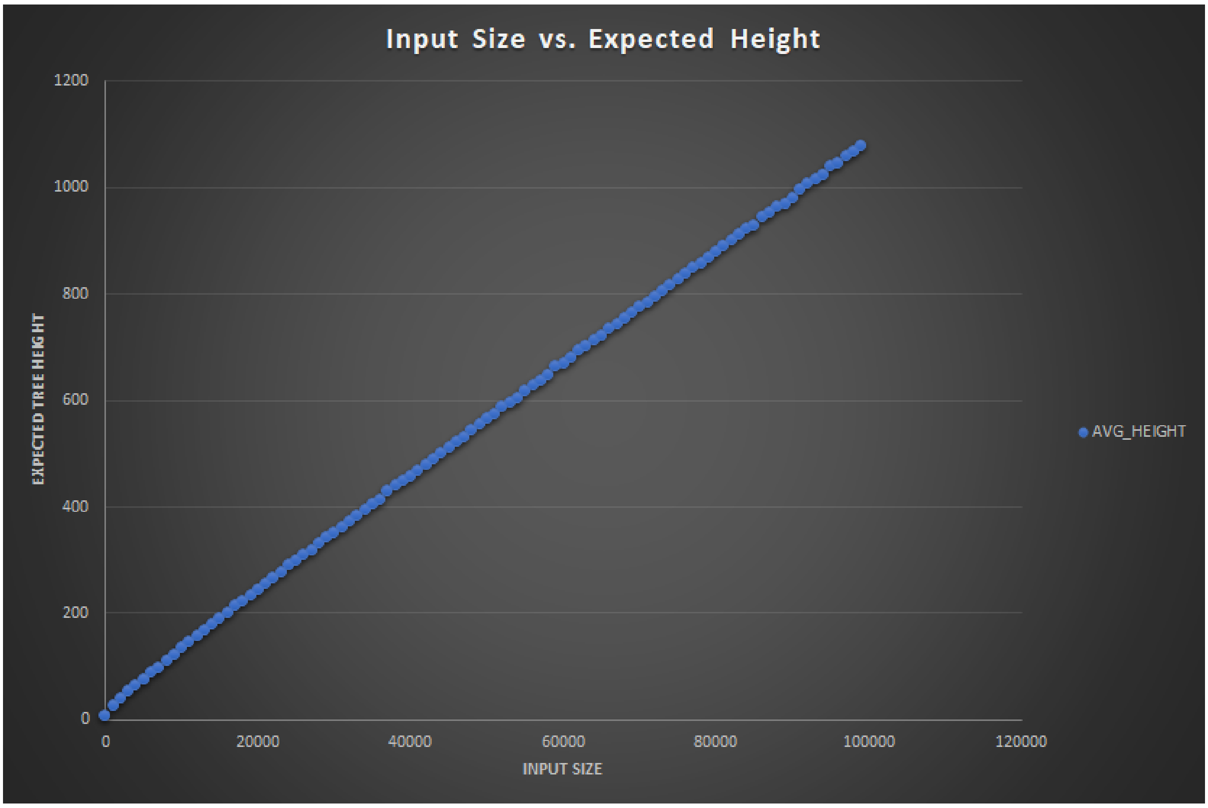


Figure 2: Comparison of input size and expected tree height.

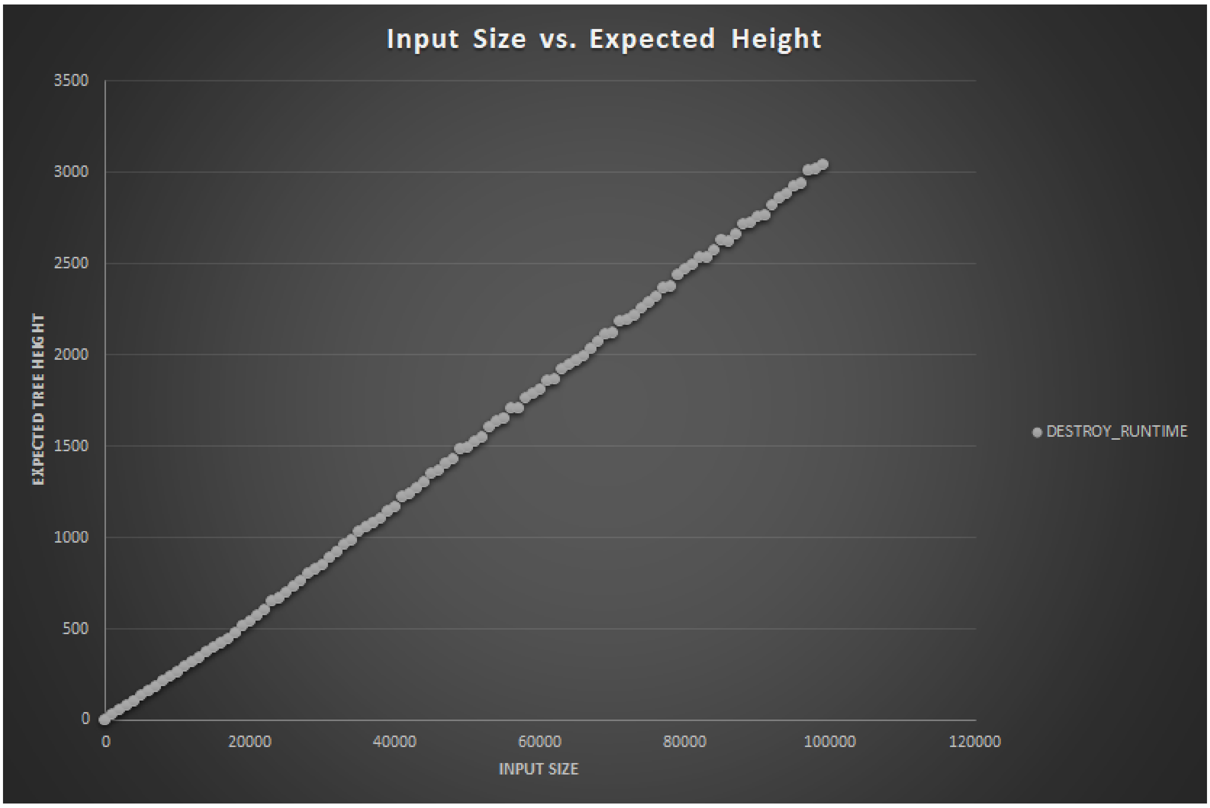


Figure 3: Comparison of input size and expected destroy time.

# Chapter 1

# Introduction

In this study, we aim to conduct a comprehensive analysis of Binary Search Trees (BSTs) by comparing the average insertion time when populating a BST with an array of randomly shuffled elements across varying input sizes. Additionally, we seek to experimentally validate the expected height of insertions in BSTs, that it is indeed of $O(\log n)$ complexity. Furthermore, we will assess the average time required for the destruction of BSTs. We also aim to determine a means to create an augmented BST which closely resembles that of a Order Statistic Tree that is not a Red-Black Tree, but has the size property used to determine the rank of a node in the tree.

We also explore augmented BSTs that follow the structure of an Order Statistic Tree, without being limited to the constraints of Red-Black Trees. This augmented BST will maintain a size property that allows us to efficiently determine the rank of a node within the tree.

# Chapter 2

# Objective

The primary objective of this experiment is to empirically demonstrate that a randomly constructed Binary Search Tree (BST) using $n$ distinct keys exhibits an expected height of $O(\log n)$. By conducting a series of experiments and data analysis, we aim to confirm that the growth in height of a BST is logarithmic in nature to the input size $n$. This is useful to support the fundamental property BSTs, enabling us to better understand the practical application of the implemented insertion and destruction methods.

We also aim to develop an augmented BST that resembles the functionality of an Order Statistic Tree while avoiding the constraints of Red-Black Trees. This augmented BST will have each node incorporate a size property, will allow us to efficiently determine the rank of a node within the tree.

# Chapter 3

# Methodology

## 3.1 Experimental Setup

The experimental implementations were coded in C++. The results obtained from the experiment were recorded in a comma-separated values (CSV) file.

### 3.1.1 Range of Dimensions & Key Values

To investigate the performance of BSTs, we populate them with randomly shuffled arrays of varying input sizes. Our input sizes range from $16$ to $100,000$ elements, with keys represented as random integers falling within the range $[1, 100]$. This comprehensive range of inputs allows us to conduct an empirical analysis and compare our experimental findings with theoretical expectations. Specifically, we aim to verify that the average insertion time exhibits linearithmic behaviour, denoted as $O(n \log n)$, as each key insertion taking $\Theta(n)$ operations, may each take up to $O(\log n)$ time.

### 3.1.2 Number of Trees

To ensure the robustness and reliability of our results, we conducted 30 iterations for each input size, making use of different sets of random key values. This allowed us to accurately observe the performance of BST creation, destruction, and expected height analysis across various scenarios, aligning our conclusions with the theoretical basis of what was stated.

# Chapter 4

# Analysis of BST

It is essential to note that a standard Binary Search Tree (BST) can become unbalanced, leading to the worst-case scenario where a subtree resembles a linked list. In such situations, certain operations may have a time complexity of $O(n)$ instead of the expected $O(\log n)$ that we aim to observe.

In the following sections, we present the outcomes of our experiments (see Table A.3) and discuss the implications of our findings.

## 4.1 Expected Height

Our analysis of the expected height of a BST aligns with the behavior of a logarithmic function, as demonstrated in Figure 2.

### 4.1.1 Average Insertion Runtime

The graph representing the average insertion runtime (Figure 1) exhibits a behaviour that appears similar to that of a quadratic function, $O(n^2)$. This observation can be justified by the fact that random shuffling does not guarantee balanced tree formations during each key insertion, which inherently takes $O(n)$ time. Consequently, in the worst-case scenario, the insertion process may indeed exhibit a time complexity resembling that of a linked list, which is $O(n)$.

### 4.1.2 Tree Destruction Runtime

To efficiently destroy the BST, we decided to perform a sequential deletion of the root node of the tree. This can be motivated by the inefficiency of using the randomly shuffled array for tree destruction, as it would require searching for the keys, a task that can take $O(n)$ time in the worst-case scenario of a highly unbalanced tree with $n$ nodes. Additionally, the deletion of a node can have a time complexity of up to $O(h)$, where $h$ denotes the height of the tree. This is due to the possibility of needing to find a successor node in cases where the node being deleted has two children. One can observe this linear trend from (Figure 3). Making use of the randomly shuffled array as a reference to delete the tree would bring in the additional cost of having to search for the node to be deleted, possibly making the tree destruction runtime reach $O(n^2)$. This justifies the approach of sequentially deleting the root node of the tree.

# Chapter 5

# Improvement to BST: The Size Attribute

Augmenting a Binary Search Tree (BST) by introducing the ”size” attribute to each node allows us to determine the rank of a node within the tree and efficiently locate the i-th order statistic node, making BSTs even more versatile.

In initializing a node, it has an initial size of 0. Since we are not implementing Red-Black Tree constraints, a check is performed to determine if a node has any children before obtaining its size property. This check ensures that the size attribute remains accurate as we manipulate the tree's structure, preventing errors relating to a null pointer reference.

The size of a node, denoted as $x$, is calculated using the following formula:

$size(x) = size(leftChild) + size(rightChild) + 1$

## 5.1 Maintaining Node Sizes on Key Insertion

To maintain the sizes of all nodes after inserting a new key, we iteratively update the sizes of the nodes along the path to the root node with (5.1.1). We use the same code for key insertion as shown in A.1. It is after inserting the key that it is necessary to update the sizes of the nodes from its inserted position to the root position. (5.1.1) guarantees that after insertion, the sizes of all nodes are correct, in being the updated size of the respective subtree.

### 5.1.1 Size Update of Inserted Node After Insert

```
void insertupdate(OS_Node *currNode) {
        while (currNode != nullptr) {
                currNode.size = (currNode.left == nullptr ? 0
                    : currNode.left.size) + (currNode.right
                  == nullptr ? 0 : currNode.right.size) + 1;
                currNode = currNode.parent;
        }
}
```

## 5.2 Maintaining Node Sizes on Key Deletion

When performing key deletion using the same code as $TREE-DELETE$ in A.2, there are two fundamental cases to consider, each requiring specific adjustments to maintain the accuracy of the size attributes.

1. **Deletion with One Child:**

In the first case, when the node being deleted has only one child, the only child becomes the next successor. To ensure that the sizes of ancestors of the node to be deleted are correctly updated, we decrement their sizes by 1.

2. **Deletion with Two Children:**

In the second case, when the node being deleted has two children, the sizes of ancestors of the chosen successor - given by the minimum node of the left child's subtree - need to be decremented by 1.

We perform the 5.2.1 operation before executing the transplant operation. This simplifies the process of not needing to perform any complex calculations on which node sizes need to be updated. We also maintain the integrity of the size attributes, ensuring that after key deletion, the accuracy in the sizes of the nodes is still maintained.

### 5.2.1 Size Update of Deleted node Before Deletion

```
void deleteUpdateSize(OS_Node *currNode) {
        while (currNode != nullptr) {
                currNode.size -= 1;
                currNode = currNode.parent;
        }
}
```

# Chapter 6

# Conclusion

In conclusion, the Binary Search Tree (BST) and its augmented version, which includes the size property, have produced the following results.

When randomly constructing the BST, its expected height has consistently remained close to an average of $O(\log n)$. This ensures that efficient search operations are possible without incurring excessive overhead in setting up the data structure.

Insertion, although not strictly linearithmic and exhibiting a quadratic trend, can be improved by incorporating the AVL property or adopting the Red-Black constraints. However, this improvement comes at the cost of a more complex implementation.

Destroying the tree by iteratively deleting the root node results in a linear trend, as observed. This approach was well-justified when compared to using each key in the randomly shuffled list used to construct the tree.

Including the size property in nodes is a simple implementation that enables the ability to determine the rank of a node and retrieve order statistic nodes. Maintaining this property is fairly straightforward and opens up opportunities for various applications that can contribute significantly to solving complex computational challenges.

# Appendix A

# Appendix

## A.1 TREE-INSERT

As adapted to C++ from the algorithm provided by Cormen *et al.* [2009]

```
void TREE-INSERT(BST& T, Node* z) {
        y = nullptr;
        x = T.root;
        while (x != nullptr)
                y = x;
                if (z.key < x.key)
                        x = x.left;
                else
                        x = x.right;
        z.p = y;
        if (y == nullptr)
                T.root = z;
        else if (z.key < y.key)
                y.left = z;
        else
                y.right = z;

        \\ For maintaining OS_BST sizes, insertUpdate(z) is
           called
}
```

## A.2 TREE-DELETE

As adapted to C++ from the algorithm provided by Cormen *et al.* [2009]

```
void TREE-DELETE(BST& T, Node* z) {
        if (z.left == nullptr) {
                \\ For maintaining OS_BST node sizes, deleteUpdateSize(z) is called here
                TRANSPLANT(T, z, z.right);
        }
        else if (z.right == nullptr) {
                \\ For maintaining OS_BST node sizes, deleteUpdateSize(z) is called here
                TRANSPLANT(T, z, z.left);
        }
        else {
                y = TREE-MINIMUM(z.right);

                \\ For maintaining OS_BST node sizes, deleteUpdateSize(y) is called here

                if (y.p != z) {
                        TRANSPLANT(T, y, y.right);
                        y.right = z.right;
                        y.right.p = y;
                }
                TRANSPLANT(T, z, y);
                y.left = z.left;
                y.left.p = y;
        }

}
```

# List of Tables

## A.3 Tabular Results

| SIZE | AVG_HEIGHT | INSERT_RUNTIME | DESTROY_RUNTIME |
|---|---|---|---|
| 16 | 7 | 2 | 0 |
| 144 | 14 | 50 | 14 |
| 272 | 18 | 80 | 23 |
| 400 | 20 | 112 | 33 |
| 528 | 21 | 156 | 40 |
| 656 | 23 | 143 | 39 |
| 784 | 26 | 154 | 45 |
| 912 | 27 | 180 | 56 |
| 1040 | 30 | 189 | 58 |
| 1168 | 30 | 236 | 73 |
| 1296 | 32 | 287 | 86 |
| 1424 | 34 | 316 | 90 |
| 1552 | 35 | 331 | 93 |
| 1680 | 37 | 342 | 101 |
| 1808 | 39 | 418 | 120 |
| 1936 | 40 | 434 | 122 |
| 2064 | 41 | 501 | 149 |
| 2192 | 44 | 491 | 132 |
| 2320 | 46 | 528 | 143 |
| 2448 | 47 | 606 | 158 |
| 2576 | 48 | 599 | 155 |
| 2704 | 51 | 614 | 155 |
| 2832 | 51 | 649 | 166 |
| 2960 | 53 | 712 | 183 |
| 3088 | 54 | 745 | 179 |
| 3216 | 56 | 783 | 187 |
| 3344 | 57 | 837 | 190 |
| 3472 | 60 | 974 | 220 |
| 3600 | 62 | 903 | 204 |
| 3728 | 62 | 988 | 224 |

| | | | |
|---|---|---|---|
| 3856 | 64 | 1015 | 220 |
| 3984 | 66 | 1084 | 230 |
| 4112 | 67 | 1128 | 238 |
| 4240 | 68 | 1188 | 239 |
| 4368 | 70 | 1246 | 245 |
| 4496 | 71 | 1298 | 247 |
| 4624 | 74 | 1363 | 260 |
| 4752 | 74 | 1424 | 266 |
| 4880 | 75 | 1479 | 270 |
| 5008 | 77 | 1584 | 277 |
| 5136 | 79 | 1627 | 279 |
| 5264 | 80 | 1799 | 310 |
| 5392 | 82 | 1773 | 304 |
| 5520 | 83 | 1865 | 305 |
| 5648 | 85 | 1950 | 313 |
| 5776 | 85 | 2032 | 320 |
| 5904 | 87 | 2142 | 344 |
| 6032 | 90 | 2232 | 358 |
| 6160 | 91 | 2323 | 364 |
| 6288 | 93 | 2432 | 365 |
| 6416 | 94 | 2505 | 366 |
| 6544 | 95 | 2583 | 378 |
| 6672 | 96 | 2702 | 388 |
| 6800 | 98 | 2792 | 390 |
| 6928 | 100 | 2940 | 405 |
| 7056 | 100 | 2957 | 395 |
| 7184 | 103 | 3210 | 442 |
| 7312 | 104 | 3310 | 429 |
| 7440 | 105 | 3332 | 422 |
| 7568 | 107 | 3379 | 430 |
| 7696 | 107 | 3520 | 431 |
| 7824 | 110 | 3694 | 452 |
| 7952 | 111 | 4237 | 487 |
| 8080 | 113 | 4171 | 495 |
| 8208 | 114 | 4340 | 507 |
| 8336 | 117 | 4559 | 519 |
| 8464 | 117 | 4507 | 500 |
| 8592 | 118 | 4709 | 529 |
| 8720 | 121 | 4612 | 494 |
| 8848 | 119 | 4759 | 503 |
| 8976 | 123 | 5095 | 523 |
| 9104 | 125 | 5037 | 532 |
| 9232 | 126 | 5289 | 522 |
| 9360 | 127 | 5641 | 550 |

| | | | |
|---|---|---|---|
| 9488 | 129 | 5453 | 515 |
| 9616 | 130 | 5548 | 521 |
| 9744 | 131 | 5683 | 526 |
| 9872 | 134 | 5835 | 540 |
| 10000 | 135 | 15555 | 1333 |
| 10128 | 135 | 19721 | 1710 |
| 10256 | 137 | 20326 | 1703 |
| 10384 | 139 | 21444 | 1735 |
| 10512 | 140 | 21903 | 1802 |
| 10640 | 142 | 22126 | 1775 |
| 10768 | 141 | 28920 | 2427 |
| 10896 | 146 | 27215 | 2212 |
| 11024 | 144 | 27440 | 2281 |
| 11152 | 148 | 26593 | 2024 |
| 11280 | 149 | 26091 | 1918 |
| 11408 | 149 | 26555 | 1944 |
| 11536 | 152 | 12505 | 916 |
| 11664 | 154 | 7232 | 531 |
| 11792 | 154 | 7305 | 532 |
| 11920 | 156 | 8532 | 625 |
| 12048 | 157 | 7812 | 562 |
| 12176 | 159 | 8001 | 554 |
| 12304 | 160 | 8122 | 561 |
| 12432 | 162 | 8471 | 594 |
| 12560 | 164 | 8946 | 593 |
| 12688 | 164 | 9062 | 601 |
| 12816 | 166 | 9399 | 619 |
| 12944 | 168 | 10033 | 630 |
| 13072 | 168 | 10323 | 677 |
| 13200 | 170 | 10377 | 665 |
| 13328 | 172 | 10388 | 629 |
| 13456 | 172 | 10543 | 653 |
| 13584 | 175 | 11159 | 674 |
| 13712 | 176 | 10898 | 665 |
| 13840 | 178 | 11533 | 686 |
| 13968 | 179 | 11902 | 703 |
| 14096 | 179 | 13366 | 780 |
| 14224 | 182 | 14475 | 921 |
| 14352 | 182 | 14613 | 824 |
| 14480 | 187 | 13117 | 728 |
| 14608 | 185 | 12837 | 706 |
| 14736 | 187 | 13145 | 722 |
| 14864 | 189 | 13338 | 723 |
| 14992 | 191 | 13707 | 735 |

| | | | |
|---|---|---|---|
| 15120 | 191 | 14090 | 744 |
| 15248 | 193 | 14330 | 736 |
| 15376 | 195 | 14580 | 774 |
| 15504 | 194 | 14884 | 788 |
| 15632 | 198 | 14990 | 748 |
| 15760 | 198 | 14920 | 731 |
| 15888 | 199 | 15417 | 796 |
| 16016 | 201 | 15785 | 771 |
| 16144 | 202 | 16432 | 809 |
| 16272 | 203 | 16430 | 822 |
| 16400 | 207 | 16453 | 788 |
| 16528 | 206 | 16778 | 791 |
| 16656 | 209 | 17066 | 788 |
| 16784 | 211 | 17510 | 810 |
| 16912 | 210 | 17546 | 829 |
| 17040 | 214 | 18521 | 839 |
| 17168 | 214 | 18140 | 827 |
| 17296 | 216 | 18578 | 842 |
| 17424 | 215 | 19774 | 1033 |
| 17552 | 219 | 19173 | 891 |
| 17680 | 221 | 18498 | 820 |
| 17808 | 220 | 18640 | 803 |
| 17936 | 222 | 19086 | 828 |
| 18064 | 225 | 19112 | 807 |
| 18192 | 226 | 19484 | 850 |
| 18320 | 228 | 20091 | 830 |
| 18448 | 226 | 20090 | 841 |
| 18576 | 229 | 20364 | 864 |
| 18704 | 229 | 20349 | 819 |
| 18832 | 232 | 20719 | 844 |
| 18960 | 234 | 21170 | 850 |
| 19088 | 234 | 21718 | 867 |
| 19216 | 235 | 21987 | 882 |
| 19344 | 240 | 21889 | 860 |
| 19472 | 238 | 22288 | 862 |
| 19600 | 241 | 22637 | 900 |
| 19728 | 243 | 32033 | 1283 |
| 19856 | 244 | 23317 | 883 |
| 19984 | 245 | 23760 | 886 |
| 20112 | 246 | 23876 | 924 |
| 20240 | 249 | 24562 | 902 |
| 20368 | 248 | 25024 | 906 |
| 20496 | 247 | 24738 | 903 |
| 20624 | 251 | 25317 | 922 |

| | | | |
|---|---|---|---|
| 20752 | 252 | 25107 | 915 |
| 20880 | 254 | 25842 | 926 |
| 21008 | 256 | 26165 | 941 |
| 21136 | 256 | 26537 | 950 |
| 21264 | 257 | 27058 | 956 |
| 21392 | 259 | 27477 | 949 |
| 21520 | 261 | 27600 | 946 |
| 21648 | 264 | 28095 | 981 |
| 21776 | 266 | 28512 | 965 |
| 21904 | 267 | 28894 | 1004 |
| 22032 | 267 | 28921 | 981 |
| 22160 | 266 | 29418 | 982 |
| 22288 | 270 | 29539 | 999 |
| 22416 | 270 | 30680 | 1009 |
| 22544 | 273 | 30841 | 1014 |
| 22672 | 274 | 31017 | 1031 |
| 22800 | 274 | 31620 | 1033 |
| 22928 | 275 | 31708 | 1032 |
| 23056 | 277 | 32152 | 1025 |
| 23184 | 282 | 32139 | 1043 |
| 23312 | 280 | 33001 | 1042 |
| 23440 | 283 | 33191 | 1066 |
| 23568 | 283 | 33511 | 1073 |
| 23696 | 285 | 34287 | 1069 |
| 23824 | 288 | 34686 | 1071 |
| 23952 | 286 | 35005 | 1071 |
| 24080 | 289 | 35150 | 1100 |
| 24208 | 292 | 35450 | 1106 |
| 24336 | 291 | 36245 | 1138 |
| 24464 | 292 | 37570 | 1123 |
| 24592 | 294 | 39117 | 1135 |
| 24720 | 294 | 41226 | 1279 |
| 24848 | 296 | 41985 | 1199 |
| 24976 | 299 | 39894 | 1195 |
| 25104 | 302 | 40061 | 1167 |
| 25232 | 301 | 41547 | 1321 |
| 25360 | 302 | 42154 | 1240 |
| 25488 | 307 | 42277 | 1265 |
| 25616 | 305 | 42627 | 1274 |
| 25744 | 307 | 40939 | 1213 |
| 25872 | 306 | 42144 | 1185 |
| 26000 | 307 | 41825 | 1182 |
| 26128 | 310 | 42607 | 1193 |
| 26256 | 312 | 42931 | 1159 |

| | | | |
|---|---|---|---|
| 26384 | 314 | 43109 | 1195 |
| 26512 | 315 | 43304 | 1222 |
| 26640 | 318 | 44202 | 1205 |
| 26768 | 320 | 44213 | 1212 |
| 26896 | 319 | 45252 | 1258 |
| 27024 | 318 | 45790 | 1224 |
| 27152 | 319 | 46054 | 1228 |
| 27280 | 322 | 46462 | 1227 |
| 27408 | 325 | 46456 | 1220 |
| 27536 | 326 | 47685 | 1240 |
| 27664 | 328 | 48003 | 1250 |
| 27792 | 327 | 48399 | 1245 |
| 27920 | 329 | 48616 | 1263 |
| 28048 | 331 | 49424 | 1265 |
| 28176 | 332 | 49631 | 1258 |
| 28304 | 334 | 50688 | 1277 |
| 28432 | 333 | 50855 | 1299 |
| 28560 | 336 | 51407 | 1308 |
| 28688 | 341 | 52106 | 1310 |
| 28816 | 341 | 52148 | 1348 |
| 28944 | 343 | 53079 | 1295 |
| 29072 | 343 | 53377 | 1319 |
| 29200 | 342 | 53221 | 1349 |
| 29328 | 342 | 54454 | 1304 |
| 29456 | 347 | 53432 | 1322 |
| 29584 | 347 | 54980 | 1361 |
| 29712 | 348 | 55466 | 1333 |
| 29840 | 352 | 56230 | 1350 |
| 29968 | 352 | 56782 | 1374 |
| 30096 | 353 | 57268 | 1357 |
| 30224 | 355 | 57230 | 1412 |
| 30352 | 359 | 58241 | 1350 |
| 30480 | 359 | 58393 | 1380 |
| 30608 | 356 | 59115 | 1357 |
| 30736 | 361 | 59494 | 1375 |
| 30864 | 361 | 60355 | 1371 |
| 30992 | 363 | 61280 | 1392 |
| 31120 | 365 | 61789 | 1405 |
| 31248 | 366 | 61705 | 1443 |
| 31376 | 367 | 62942 | 1408 |
| 31504 | 369 | 62127 | 1419 |
| 31632 | 371 | 68727 | 1772 |
| 31760 | 373 | 68037 | 1615 |
| 31888 | 373 | 70395 | 1698 |

| | | | |
|---|---|---|---|
| 32016 | 375 | 71459 | 1654 |
| 32144 | 374 | 66226 | 1446 |
| 32272 | 377 | 65818 | 1459 |
| 32400 | 380 | 67579 | 1496 |
| 32528 | 379 | 66585 | 1438 |
| 32656 | 381 | 67927 | 1442 |
| 32784 | 382 | 68874 | 1486 |
| 32912 | 384 | 68784 | 1461 |
| 33040 | 386 | 69697 | 1513 |
| 33168 | 384 | 70599 | 1540 |
| 33296 | 384 | 71704 | 1514 |
| 33424 | 390 | 71272 | 1528 |
| 33552 | 389 | 72819 | 1614 |
| 33680 | 390 | 72835 | 1508 |
| 33808 | 392 | 72392 | 1516 |
| 33936 | 394 | 71935 | 1521 |
| 34064 | 396 | 74547 | 1533 |
| 34192 | 400 | 75049 | 1593 |
| 34320 | 401 | 76458 | 1559 |
| 34448 | 399 | 76388 | 1554 |
| 34576 | 402 | 76893 | 1606 |
| 34704 | 405 | 78037 | 1612 |
| 34832 | 408 | 78959 | 1566 |
| 34960 | 406 | 79302 | 1567 |
| 35088 | 405 | 79818 | 1560 |
| 35216 | 408 | 78921 | 1573 |
| 35344 | 409 | 80307 | 1593 |
| 35472 | 414 | 80540 | 1592 |
| 35600 | 414 | 80646 | 1632 |
| 35728 | 411 | 81866 | 1615 |
| 35856 | 417 | 83284 | 1624 |
| 35984 | 416 | 84341 | 1631 |
| 36112 | 419 | 83663 | 1610 |
| 36240 | 418 | 84823 | 1646 |
| 36368 | 422 | 84502 | 1623 |
| 36496 | 423 | 85052 | 1634 |
| 36624 | 423 | 87290 | 1663 |
| 36752 | 424 | 86924 | 1672 |
| 36880 | 426 | 87211 | 1696 |
| 37008 | 426 | 88872 | 1679 |
| 37136 | 427 | 90042 | 1680 |
| 37264 | 430 | 89930 | 1711 |
| 37392 | 431 | 90124 | 1712 |
| 37520 | 432 | 90757 | 1726 |

| | | | |
|---|---|---|---|
| 37648 | 434 | 92253 | 1870 |
| 37776 | 439 | 93179 | 1732 |
| 37904 | 438 | 93773 | 1754 |
| 38032 | 441 | 95026 | 1746 |
| 38160 | 439 | 93265 | 1739 |
| 38288 | 445 | 94072 | 1697 |
| 38416 | 440 | 97807 | 1785 |
| 38544 | 446 | 95913 | 1769 |
| 38672 | 446 | 96909 | 1741 |
| 38800 | 445 | 97483 | 1773 |
| 38928 | 448 | 97918 | 1757 |
| 39056 | 447 | 98276 | 1781 |
| 39184 | 450 | 100113 | 1817 |
| 39312 | 451 | 100917 | 1772 |
| 39440 | 456 | 101217 | 1751 |
| 39568 | 456 | 101578 | 1763 |
| 39696 | 455 | 102933 | 1800 |
| 39824 | 458 | 103172 | 1800 |
| 39952 | 459 | 103262 | 1783 |
| 40080 | 461 | 102678 | 1820 |
| 40208 | 461 | 105614 | 1817 |
| 40336 | 465 | 107191 | 1810 |
| 40464 | 464 | 108199 | 1805 |
| 40592 | 466 | 108649 | 1867 |
| 40720 | 466 | 108684 | 1821 |
| 40848 | 465 | 109482 | 1879 |
| 40976 | 471 | 109803 | 1864 |
| 41104 | 469 | 110158 | 1838 |
| 41232 | 472 | 111264 | 1853 |
| 41360 | 473 | 111152 | 1903 |
| 41488 | 477 | 112455 | 1867 |
| 41616 | 478 | 112325 | 1920 |
| 41744 | 479 | 115300 | 1920 |
| 41872 | 475 | 113849 | 1914 |
| 42000 | 480 | 115052 | 1884 |
| 42128 | 482 | 115777 | 1892 |
| 42256 | 482 | 117777 | 1921 |
| 42384 | 485 | 117523 | 1925 |
| 42512 | 487 | 117562 | 1909 |
| 42640 | 487 | 120324 | 1954 |
| 42768 | 489 | 119851 | 1932 |
| 42896 | 489 | 120850 | 1929 |
| 43024 | 491 | 120202 | 1986 |
| 43152 | 492 | 121119 | 1954 |

| | | | |
|---|---|---|---|
| 43280 | 496 | 121685 | 1906 |
| 43408 | 494 | 124469 | 1983 |
| 43536 | 497 | 123543 | 1992 |
| 43664 | 498 | 124239 | 1947 |
| 43792 | 498 | 125112 | 1989 |
| 43920 | 500 | 127794 | 1989 |
| 44048 | 503 | 126129 | 1984 |
| 44176 | 507 | 130050 | 2012 |
| 44304 | 505 | 129814 | 2040 |
| 44432 | 507 | 128762 | 2024 |
| 44560 | 506 | 130674 | 2013 |
| 44688 | 509 | 132023 | 2040 |
| 44816 | 507 | 132581 | 2015 |
| 44944 | 512 | 133249 | 2076 |
| 45072 | 509 | 133260 | 2034 |
| 45200 | 514 | 133101 | 2054 |
| 45328 | 515 | 135371 | 2063 |
| 45456 | 517 | 136498 | 2067 |
| 45584 | 522 | 137179 | 2157 |
| 45712 | 519 | 137798 | 2168 |
| 45840 | 520 | 138170 | 2107 |
| 45968 | 523 | 138957 | 2171 |
| 46096 | 525 | 141162 | 2167 |
| 46224 | 522 | 139647 | 2060 |
| 46352 | 529 | 140733 | 2094 |
| 46480 | 529 | 142750 | 2130 |
| 46608 | 525 | 141415 | 2081 |
| 46736 | 533 | 144141 | 2134 |
| 46864 | 531 | 144545 | 2103 |
| 46992 | 533 | 146182 | 2111 |
| 47120 | 534 | 146253 | 2164 |
| 47248 | 537 | 148652 | 2150 |
| 47376 | 539 | 147806 | 2158 |
| 47504 | 536 | 148011 | 2209 |
| 47632 | 542 | 149983 | 2191 |
| 47760 | 541 | 151748 | 2178 |
| 47888 | 543 | 153763 | 2175 |
| 48016 | 541 | 153102 | 2277 |
| 48144 | 547 | 153161 | 2254 |
| 48272 | 544 | 154451 | 2224 |
| 48400 | 547 | 153369 | 2192 |
| 48528 | 550 | 156991 | 2220 |
| 48656 | 554 | 157428 | 2241 |
| 48784 | 552 | 157160 | 2197 |

| | | | |
|---|---|---|---|
| 48912 | 555 | 157825 | 2204 |
| 49040 | 556 | 159922 | 2207 |
| 49168 | 560 | 158444 | 2235 |
| 49296 | 556 | 161062 | 2229 |
| 49424 | 557 | 162597 | 2228 |
| 49552 | 556 | 164354 | 2219 |
| 49680 | 564 | 164020 | 2264 |
| 49808 | 562 | 165510 | 2210 |
| 49936 | 564 | 165264 | 2243 |
| 50064 | 566 | 166582 | 2307 |
| 50192 | 566 | 167158 | 2316 |
| 50320 | 566 | 169417 | 2287 |
| 50448 | 571 | 171481 | 2314 |
| 50576 | 574 | 169023 | 2281 |
| 50704 | 571 | 170475 | 2322 |
| 50832 | 574 | 172416 | 2394 |
| 50960 | 576 | 172810 | 2366 |
| 51088 | 577 | 174358 | 2359 |
| 51216 | 578 | 174510 | 2355 |
| 51344 | 580 | 175233 | 2352 |
| 51472 | 587 | 175407 | 2401 |
| 51600 | 580 | 177762 | 2365 |
| 51728 | 579 | 176102 | 2401 |
| 51856 | 584 | 179310 | 2360 |
| 51984 | 585 | 177228 | 2335 |
| 52112 | 588 | 179949 | 2369 |
| 52240 | 587 | 180616 | 2363 |
| 52368 | 588 | 182715 | 2368 |
| 52496 | 592 | 205968 | 2780 |
| 52624 | 593 | 182790 | 2395 |
| 52752 | 593 | 186480 | 2378 |
| 52880 | 594 | 186698 | 2428 |
| 53008 | 597 | 187779 | 2438 |
| 53136 | 596 | 189586 | 2552 |
| 53264 | 598 | 188808 | 2460 |
| 53392 | 602 | 191028 | 2469 |
| 53520 | 600 | 191537 | 2415 |
| 53648 | 602 | 192291 | 2456 |
| 53776 | 605 | 192465 | 2450 |
| 53904 | 604 | 195437 | 2476 |
| 54032 | 606 | 195434 | 2523 |
| 54160 | 610 | 197505 | 2496 |
| 54288 | 609 | 196138 | 2494 |
| 54416 | 612 | 197156 | 2511 |

| | | | |
|---|---|---|---|
| 54544 | 611 | 200165 | 2543 |
| 54672 | 614 | 199705 | 2504 |
| 54800 | 612 | 201317 | 2514 |
| 54928 | 621 | 205013 | 2498 |
| 55056 | 616 | 203142 | 2547 |
| 55184 | 625 | 204548 | 2547 |
| 55312 | 624 | 205019 | 2666 |
| 55440 | 627 | 206630 | 2533 |
| 55568 | 622 | 206126 | 2597 |
| 55696 | 625 | 206658 | 2578 |
| 55824 | 622 | 209344 | 2578 |
| 55952 | 629 | 211509 | 2625 |
| 56080 | 629 | 209203 | 2625 |
| 56208 | 631 | 211636 | 2652 |
| 56336 | 628 | 212049 | 2555 |
| 56464 | 632 | 215015 | 2574 |
| 56592 | 634 | 213753 | 2567 |
| 56720 | 635 | 218410 | 2646 |
| 56848 | 636 | 215722 | 2591 |
| 56976 | 635 | 221262 | 2615 |
| 57104 | 641 | 220277 | 2665 |
| 57232 | 642 | 221443 | 2817 |
| 57360 | 642 | 227251 | 2628 |
| 57488 | 647 | 223062 | 2697 |
| 57616 | 647 | 227636 | 2768 |
| 57744 | 647 | 225410 | 2718 |
| 57872 | 650 | 226579 | 2673 |
| 58000 | 648 | 227650 | 2853 |
| 58128 | 650 | 230398 | 2799 |
| 58256 | 649 | 229457 | 2808 |
| 58384 | 653 | 228909 | 2625 |
| 58512 | 654 | 230120 | 2670 |
| 58640 | 658 | 230544 | 2670 |
| 58768 | 657 | 230849 | 2723 |
| 58896 | 659 | 230498 | 2690 |
| 59024 | 662 | 234719 | 2705 |
| 59152 | 665 | 237551 | 2728 |
| 59280 | 664 | 235929 | 2718 |
| 59408 | 663 | 234655 | 2689 |
| 59536 | 665 | 237013 | 2708 |
| 59664 | 669 | 238595 | 2728 |
| 59792 | 671 | 240337 | 2710 |
| 59920 | 668 | 242228 | 2738 |
| 60048 | 669 | 240750 | 2735 |

| | | | |
|---|---|---|---|
| 60176 | 674 | 240903 | 2688 |
| 60304 | 671 | 244706 | 2777 |
| 60432 | 674 | 246157 | 2768 |
| 60560 | 676 | 244190 | 2779 |
| 60688 | 677 | 247122 | 2777 |
| 60816 | 677 | 249880 | 2916 |
| 60944 | 678 | 251653 | 2933 |
| 61072 | 681 | 250025 | 2902 |
| 61200 | 681 | 252658 | 2924 |
| 61328 | 685 | 255493 | 2973 |
| 61456 | 687 | 253200 | 2955 |
| 61584 | 688 | 252725 | 2990 |
| 61712 | 686 | 257158 | 2969 |
| 61840 | 686 | 259155 | 3005 |
| 61968 | 691 | 258984 | 2887 |
| 62096 | 691 | 263566 | 2969 |
| 62224 | 696 | 262781 | 2869 |
| 62352 | 699 | 266447 | 3041 |
| 62480 | 696 | 269258 | 2921 |
| 62608 | 695 | 265346 | 2970 |
| 62736 | 700 | 268876 | 2903 |
| 62864 | 699 | 267018 | 2936 |
| 62992 | 701 | 270871 | 2923 |
| 63120 | 700 | 269418 | 2991 |
| 63248 | 704 | 273081 | 2932 |
| 63376 | 707 | 272650 | 2920 |
| 63504 | 708 | 273542 | 3030 |
| 63632 | 714 | 275290 | 3038 |
| 63760 | 712 | 276485 | 3039 |
| 63888 | 712 | 278638 | 3042 |
| 64016 | 713 | 276616 | 2929 |
| 64144 | 712 | 280540 | 2966 |
| 64272 | 712 | 280782 | 2942 |
| 64400 | 718 | 279163 | 2912 |
| 64528 | 719 | 282160 | 3053 |
| 64656 | 716 | 312760 | 3017 |
| 64784 | 720 | 284863 | 3005 |
| 64912 | 723 | 282833 | 2938 |
| 65040 | 723 | 284668 | 2983 |
| 65168 | 725 | 287142 | 2960 |
| 65296 | 728 | 289324 | 3020 |
| 65424 | 728 | 289319 | 3008 |

# References

[Cormen *et al.* 2009] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*, 2009.