

# Table of Contents

Table of Contents	iii
<b>1 Review of Terms, Proofs, and Probability</b>	<b>1</b>
1.1 Review of Proofs and Proof Techniques . . . . .	1
Induction . . . . .	2
1.2 Graphs . . . . .	3
Trees . . . . .	4
Eulerian and Hamiltonian Graphs . . . . .	5
1.3 Probability . . . . .	6
1.4 Linearity of Expectation . . . . .	8
1.5 Probability Distributions . . . . .	9
The Geometric Distribution . . . . .	10
Binomial Distributions . . . . .	11
Examples . . . . .	13
<b>2 Gale-Shapley Stable Matching</b>	<b>15</b>
2.1 Background and Intuition . . . . .	15
2.2 Formulating the Problem . . . . .	15
2.3 Examples . . . . .	17
2.4 Designing an Algorithm . . . . .	18
2.5 Runtime of the GS Algorithm . . . . .	19
2.6 Correctness of the GS Algorithm . . . . .	19
2.7 Extensions . . . . .	21
<b>3 Greatest Common Divisor</b>	<b>24</b>
3.1 Definitions . . . . .	24
3.2 Calculating the GCD . . . . .	24
3.3 Correctness of Euclid's Algorithm . . . . .	26
3.4 Runtime of Euclid's Algorithm . . . . .	27
<b>4 Insertion Sort</b>	<b>29</b>
4.1 Insertion Sort . . . . .	29
4.2 Correctness of Insertion Sort . . . . .	30
4.3 Running Time of Insertion Sort . . . . .	30
<b>5 Running Time and Growth Functions</b>	<b>32</b>
5.1 Measuring Running Time of Algorithms . . . . .	32
5.2 RAM Model of Computation . . . . .	32
5.3 Average Case and Worst Case . . . . .	32

5.4	Order of Growth . . . . .	33
	Definitions of Asymptotic Notations – $O, \Omega, \Theta, o, \omega$ . . . . .	33
5.5	Properties of Asymptotic Growth Functions . . . . .	37
<b>6</b>	<b>Analyzing Runtime of Code Snippets</b>	<b>38</b>
<b>7</b>	<b>Divide &amp; Conquer and Recurrence Relations</b>	<b>41</b>
7.1	Computing Powers of Two . . . . .	41
7.2	Linear Search and Binary Search . . . . .	43
7.3	MergeSort . . . . .	43
7.4	More Recurrence Practice . . . . .	46
7.5	Simplified Master Theorem . . . . .	48
<b>8</b>	<b>Quicksort</b>	<b>49</b>
8.1	Deterministic Quicksort . . . . .	49
8.2	Randomized Quicksort . . . . .	50
<b>9</b>	<b>Counting Inversions</b>	<b>52</b>
9.1	Introduction and Problem Description . . . . .	52
9.2	Designing an Algorithm . . . . .	53
9.3	Runtime . . . . .	55
<b>10</b>	<b>Selection Problem</b>	<b>56</b>
10.1	Introduction to Problem . . . . .	56
10.2	Selection in Worst-Case Linear Time . . . . .	56
<b>11</b>	<b>Closest Pair</b>	<b>59</b>
11.1	Closest Pair . . . . .	59
11.2	Divide and Conquer Algorithm . . . . .	60
11.3	Closest Pair Between the Sets . . . . .	61
<b>12</b>	<b>Integer Multiplication</b>	<b>63</b>
12.1	Introduction and Problem Statement . . . . .	63
12.2	Designing the Algorithm . . . . .	63
12.3	Runtime . . . . .	65
<b>13</b>	<b>Stacks and Queues</b>	<b>66</b>
13.1	The Stack ADT . . . . .	66
13.2	Queues . . . . .	69
<b>14</b>	<b>Binary Heaps and Heapsort</b>	<b>70</b>
14.1	Definitions and Implementation . . . . .	70
14.2	Maintaining the Heap Property . . . . .	71
14.3	Building a Heap . . . . .	72
	Correctness . . . . .	73

Runtime . . . . .	73
14.4 Heapsort . . . . .	75
14.5 Priority Queues . . . . .	75
<b>15 Huffman Coding</b>	<b>79</b>
15.1 From Text to Bits . . . . .	79
15.2 Variable-Length Encoding Schemes . . . . .	79
Prefix Codes . . . . .	80
Optimal Prefix Codes . . . . .	80
15.3 Huffman Encoding . . . . .	81
Binary Trees and Prefix Codes . . . . .	81
Huffman's Algorithm . . . . .	84
Designing the Algorithm . . . . .	85
Correctness of Huffman's Algorithm . . . . .	86
Running Time . . . . .	87
15.4 Extensions . . . . .	88
<b>16 Graph Traversals: BFS and DFS</b>	<b>90</b>
16.1 Graphs and Graph Representations . . . . .	90
Graph Representations . . . . .	90
16.2 Connectivity . . . . .	91
16.3 Breadth-First Search (BFS) . . . . .	91
BFS Properties . . . . .	93
Runtime of BFS . . . . .	94
16.4 Depth-First Search (DFS) . . . . .	94
Runtime of DFS . . . . .	96
DFS Properties and Extensions . . . . .	97
Classifying Edges . . . . .	99
<b>17 Application of BFS: Bipartiteness</b>	<b>101</b>
17.1 Definitions and Properties . . . . .	101
17.2 Algorithm . . . . .	101
17.3 Analysis . . . . .	101
<b>18 DAGs and Topological Sorting</b>	<b>103</b>
18.1 DAGs . . . . .	103
18.2 Topological Sorting . . . . .	103
18.3 Kahn's Algorithm . . . . .	104
18.4 Tarjan's Algorithm . . . . .	106
<b>19 Strongly Connected Components</b>	<b>107</b>
19.1 Introduction and Definitions . . . . .	107
19.2 Kosaraju's Algorithm . . . . .	109
Proof of Correctness . . . . .	109

<b>20 Shortest Path</b>	<b>112</b>
20.1 The Shortest Path Problem . . . . .	112
20.2 Dijkstra's Algorithm . . . . .	112
Analyzing the Algorithm . . . . .	113
Implementation and Running Time . . . . .	115
20.3 Shortest Path in DAGs . . . . .	115
<b>21 Minimum Spanning Trees</b>	<b>118</b>
21.1 Introduction and Background . . . . .	118
21.2 MST Algorithms . . . . .	119
Prim's Algorithm . . . . .	119
Kruskal's Algorithm . . . . .	120
Reverse-Delete . . . . .	121
21.3 Correctness of Prim's, Kruskal's, and Reverse-Delete . . . . .	122
Prim's Algorithm: Correctness . . . . .	124
Kruskal's Algorithm: Correctness . . . . .	124
Reverse-Delete Algorithm: Correctness . . . . .	124
21.4 Eliminating the Assumption that All Edge Weights are Distinct . . . . .	124
<b>22 Union Find</b>	<b>126</b>
22.1 Introduction . . . . .	126
22.2 Union by Rank . . . . .	126
22.3 Path Compression . . . . .	129
<b>23 Hashing</b>	<b>132</b>
23.1 Direct-Address Tables . . . . .	132
23.2 Hash Tables . . . . .	133
Collision Resolution by Chaining . . . . .	134
Analysis of Hashing with Chaining . . . . .	134
23.3 Hash Functions . . . . .	136
What makes a good hash function? . . . . .	136
Interpreting Keys as Natural Numbers . . . . .	136
The Division Method . . . . .	137
The Multiplication Method . . . . .	137
23.4 Open Addressing . . . . .	137
Linear Probing . . . . .	138
Quadratic Probing . . . . .	139
Double Hashing . . . . .	139
Analysis of Open-Address Hashing . . . . .	140
<b>24 Tries</b>	<b>142</b>
24.1 Introduction . . . . .	142
24.2 Standard Tries . . . . .	142
24.3 Compressed Tries . . . . .	144

24.4 Suffix Tries . . . . .	146
Saving Space . . . . .	147
Construction . . . . .	147
Using a Suffix Trie . . . . .	147
<b>25 Balanced BSTs: AVL Trees</b>	<b>148</b>
25.1 Review: Binary Search Tree . . . . .	148
25.2 Definition of an AVL Tree . . . . .	148
25.3 Update Operations: Insertion and Deletion . . . . .	150
Insertion . . . . .	150
Deletion . . . . .	153
 <b>ADVANCED TOPICS</b>	 <b>155</b>
<b>26 Skip Lists</b>	<b>156</b>
26.1 Skip Lists . . . . .	156
26.2 Analysis . . . . .	158
<b>27 Bloom Filters</b>	<b>162</b>
27.1 Bloom Filters . . . . .	162
<b>28 Balanced BSTs: Red-Black Trees</b>	<b>164</b>
28.1 Properties of Red-Black Trees . . . . .	164
<b>29 Minimum Cut</b>	<b>166</b>
29.1 The Minimum Cut Problem . . . . .	166
29.2 A Randomized Approach . . . . .	166
29.3 Probabilistic Analysis . . . . .	167
<b>30 2-SAT</b>	<b>169</b>
30.1 Introduction to the 2-SAT Problem . . . . .	169
30.2 Randomized 2-SAT Algorithm . . . . .	169
30.3 Probabilistic Analysis . . . . .	170
 <b>APPENDIX</b>	 <b>173</b>
<b>A Common Running Times</b>	<b>174</b>

# Review of Terms, Proofs, and Probability

# 1

## 1.1 Review of Proofs and Proof Techniques

The *unique factorization theorem* states that every positive number can be uniquely represented as a product of primes. More formally, it can be stated as follows.

Given any integer  $n > 1$ , there exist a positive integer  $k$ , distinct prime numbers  $p_1, p_2, \dots, p_k$ , and positive integers  $e_1, e_2, \dots, e_k$  such that

$$n = p_1^{e_1} p_2^{e_2} p_3^{e_3} \cdots p_k^{e_k}$$

and any other expression of  $n$  as a product of primes is identical to this except, perhaps, for the order in which the factors are written.

**Example.** Prove that  $\sqrt{2}$  is irrational using the unique factorization theorem.

**Solution.** Assume for the purpose of contradiction that  $\sqrt{2}$  is rational. Then there are numbers  $a$  and  $b$  ( $b \neq 0$ ) such that

$$\sqrt{2} = \frac{a}{b}$$

Squaring both sides of the above equation gives

$$\begin{aligned} 2 &= \frac{a^2}{b^2} \\ a^2 &= 2b^2 \end{aligned}$$

Let  $S(m)$  be the sum of the number of times each prime factor occurs in the unique factorization of  $m$ . Note that  $S(a^2)$  and  $S(b^2)$  is even. Why? Because the number of times that each prime factor appears in the prime factorization of  $a^2$  and  $b^2$  is exactly twice the number of times that it appears in the prime factorization of  $a$  and  $b$ . Then,  $S(2b^2)$  must be odd. This is a contradiction as  $S(a^2)$  is even and the prime factorization of a positive integer is unique.

**Example.** Prove or disprove that the sum of two irrational numbers is irrational.

**Solution.** The above statement is false. Consider the two irrational numbers,  $\sqrt{2}$  and  $-\sqrt{2}$ . Their sum is  $0 = 0/1$ , a rational number.

**Example.** Show that there exist irrational numbers  $x$  and  $y$  such that  $x^y$  is rational.

**Solution.** We know that  $\sqrt{2}$  is an irrational number. Consider  $\sqrt{2}^{\sqrt{2}}$ .

Case I:  $\sqrt{2}^{\sqrt{2}}$  is rational.

In this case we are done by setting  $x = y = \sqrt{2}$ .