each level. For example, once we have found the values of the three children of the root, which are 1, -1, and -1, we find the value of the root by computing $\max(1, -1, -1) = 1$. Because the value of the root is 1, it follows that the first player wins when both players follow a minmax strategy. ◀

Chess programs on smartphones can now play at the grandmaster level.

Game trees for some well-known games can be extraordinarily large, because these games have many different possible moves. For example, the game tree for chess has been estimated to have as many as 10^{100} vertices! It may be impossible to use Theorem 3 directly to study a game because of the size of the game tree. Therefore, various approaches have been devised to help determine good strategies and to determine the outcome of such games. One useful technique, called *alpha-beta pruning*, eliminates much computation by pruning portions of the game tree that cannot affect the values of ancestor vertices. (For information about alpha-beta pruning, consult [Gr90].) Another useful approach is to use *evaluation functions*, which estimate the value of internal vertices in the game tree when it is not feasible to compute these values exactly. For example, in the game of tic-tac-toe, as an evaluation function for a position, we may use the number of files (rows, columns, and diagonals) containing no Os (used to indicate moves of the second player) minus the number of files containing no Xs (used to indicate moves of the first player). This evaluation function provides some indication of which player has the advantage in the game. Once the values of an evaluation function are inserted, the value of the game can be computed following the rules used for the minmax strategy. Computer programs created to play chess, such as the famous Deep Blue program, are based on sophisticated evaluation functions. For more information about how computers play chess see [Le91].

Links

Exercises

1. Build a binary search tree for the words *banana, peach, apple, pear, coconut, mango*, and *papaya* using alphabetical order.

2. Build a binary search tree for the words *oenology, phrenology, campanology, ornithology, ichthyology, limnology, alchemy*, and *astrology* using alphabetical order.

3. How many comparisons are needed to locate or to add each of these words in the search tree for Exercise 1, starting fresh each time?

a) *pear* **b)** *banana*
c) *kumquat* **d)** *orange*

4. How many comparisons are needed to locate or to add each of the words in the search tree for Exercise 2, starting fresh each time?

a) *palmistry* **b)** *etymology*
c) *paleontology* **d)** *glaciology*

5. Using alphabetical order, construct a binary search tree for the words in the sentence "*The quick brown fox jumps over the lazy dog.*"

6. How many weighings of a balance scale are needed to find a lighter counterfeit coin among four coins? Describe an algorithm to find the lighter coin using this number of weighings.

7. How many weighings of a balance scale are needed to find a counterfeit coin among four coins if the counterfeit coin may be either heavier or lighter than the others? Describe an algorithm to find the counterfeit coin using this number of weighings.

***8.** How many weighings of a balance scale are needed to find a counterfeit coin among eight coins if the counterfeit coin is either heavier or lighter than the others? Describe an algorithm to find the counterfeit coin using this number of weighings.

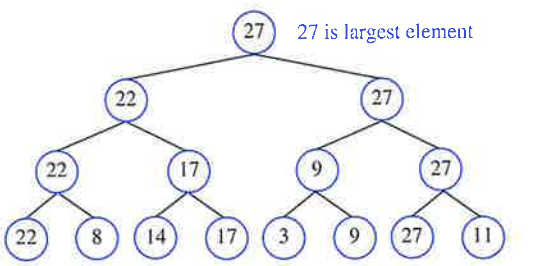
***9.** How many weighings of a balance scale are needed to find a counterfeit coin among 12 coins if the counterfeit coin is lighter than the others? Describe an algorithm to find the lighter coin using this number of weighings.

***10.** One of four coins may be counterfeit. If it is counterfeit, it may be lighter or heavier than the others. How many weighings are needed, using a balance scale, to determine whether there is a counterfeit coin, and if there is, whether it is lighter or heavier than the others? Describe an algorithm to find the counterfeit coin and determine whether it is lighter or heavier using this number of weighings.

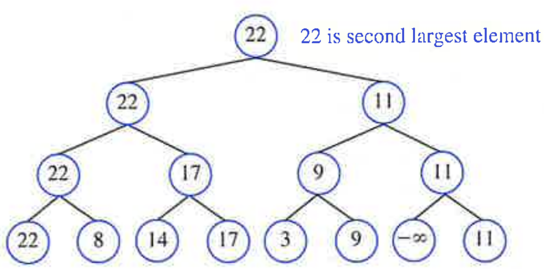
11. Find the least number of comparisons needed to sort four elements and devise an algorithm that sorts these elements using this number of comparisons.

***12.** Find the least number of comparisons needed to sort five elements and devise an algorithm that sorts these elements using this number of comparisons.

The **tournament sort** is a sorting algorithm that works by building an ordered binary tree. We represent the elements to be sorted by vertices that will become the leaves. We build up the tree one level at a time as we would construct the tree representing the winners of matches in a tournament. Working left to right, we compare pairs of consecutive elements, adding a parent vertex labeled with the larger of the two elements under comparison. We make similar comparisons between labels of vertices at each level until we reach the root of the tree that is labeled with the largest element. The tree constructed by the tournament sort of 22, 8, 14, 17, 3, 9, 27, 11 is illustrated in part (a) of the figure. Once the largest element has been determined, the leaf with this label is relabeled by $-\infty$, which is defined to be less than every element. The labels of all vertices on the path from this vertex up to the root of the tree are recalculated, as shown in part (b) of the figure. This produces the second largest element. This process continues until the entire list has been sorted.



(a)



(b)

13. Complete the tournament sort of the list 22, 8, 14, 17, 3, 9, 27, 11. Show the labels of the vertices at each step.

14. Use the tournament sort to sort the list 17, 4, 1, 5, 13, 10, 14, 6.

15. Describe the tournament sort using pseudocode.

16. Assuming that n, the number of elements to be sorted, equals 2^k for some positive integer k, determine the number of comparisons used by the tournament sort to find the largest element of the list using the tournament sort.

17. How many comparisons does the tournament sort use to find the second largest, the third largest, and so on, up to the $(n-1)$st largest (or second smallest) element?

18. Show that the tournament sort requires $\Theta(n \log n)$ comparisons to sort a list of n elements. [*Hint:* By inserting the appropriate number of dummy elements defined to be smaller than all integers, such as $-\infty$, assume that $n = 2^k$ for some positive integer k.]

19. Which of these codes are prefix codes?

a) a: 11, e: 00, t: 10, s: 01

b) a: 0, e: 1, t: 01, s: 001

c) a: 101, e: 11, t: 001, s: 011, n: 010

d) a: 010, e: 11, t: 011, s: 1011, n: 1001, i: 10101

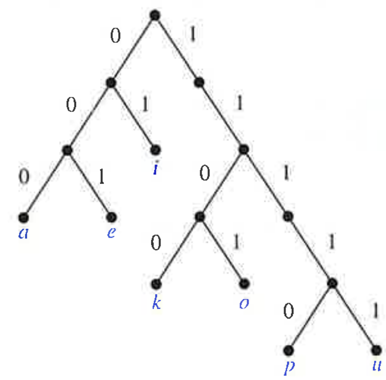
20. Construct the binary tree with prefix codes representing these coding schemes.

a) a: 11, e: 0, t: 101, s: 100

b) a: 1, e: 01, t: 001, s: 0001, n: 00001

c) a: 1010, e: 0, t: 11, s: 1011, n: 1001, i: 10001

21. What are the codes for a, e, i, k, o, p, and u if the coding scheme is represented by this tree?



22. Given the coding scheme a: 001, b: 0001, e: 1, r: 0000, s: 0100, t: 011, x: 01010, find the word represented by

a) 01110100011. **b)** 0001110000.

c) 0100101010. **d)** 01100101010.

23. Use Huffman coding to encode these symbols with given frequencies: a: 0.20, b: 0.10, c: 0.15, d: 0.25, e: 0.30. What is the average number of bits required to encode a character?

24. Use Huffman coding to encode these symbols with given frequencies: A: 0.10, B: 0.25, C: 0.05, D: 0.15, E: 0.30, F: 0.07, G: 0.08. What is the average number of bits required to encode a symbol?

25. Construct two different Huffman codes for these symbols and frequencies: t: 0.2, u: 0.3, v: 0.2, w: 0.3.

26. a) Use Huffman coding to encode these symbols with frequencies a: 0.4, b: 0.2, c: 0.2, d: 0.1, e: 0.1 in two different ways by breaking ties in the algorithm differently. First, among the trees of minimum weight select two trees with the largest number of vertices to combine at each stage of the algorithm. Second, among the trees of minimum weight select two trees with the smallest number of vertices at each stage.

b) Compute the average number of bits required to encode a symbol with each code and compute the variances of this number of bits for each code. Which tie-breaking procedure produced the smaller variance in the number of bits required to encode a symbol?

27. Construct a Huffman code for the letters of the English alphabet where the frequencies of letters in typical English text are as shown in this table.

Letter	*Frequency*	*Letter*	*Frequency*
A	0.0817	N	0.0662
B	0.0145	O	0.0781
C	0.0248	P	0.0156
D	0.0431	Q	0.0009
E	0.1232	R	0.0572
F	0.0209	S	0.0628
G	0.0182	T	0.0905
H	0.0668	U	0.0304
I	0.0689	V	0.0102
J	0.0010	W	0.0264
K	0.0080	X	0.0015
L	0.0397	Y	0.0211
M	0.0277	Z	0.0005

Suppose that m is a positive integer with $m \geq 2$. An m-ary Huffman code for a set of N symbols can be constructed analogously to the construction of a binary Huffman code. At the initial step, $((N-1) \mathbf{mod} (m-1)) + 1$ trees consisting of a single vertex with least weights are combined into a rooted tree with these vertices as leaves. At each subsequent step, the m trees of least weight are combined into an m-ary tree.

28. Describe the m-ary Huffman coding algorithm in pseudocode.

29. Using the symbols 0, 1, and 2 use ternary ($m = 3$) Huffman coding to encode these letters with the given frequencies: A: 0.25, E: 0.30, N: 0.10, R: 0.05, T: 0.12, Z: 0.18.

30. Consider the three symbols A, B, and C with frequencies A: 0.80, B: 0.19, C: 0.01.
   a) Construct a Huffman code for these three symbols.
   b) Form a new set of nine symbols by grouping together blocks of two symbols, AA, AB, AC, BA, BB, BC, CA, CB, and CC. Construct a Huffman code for these nine symbols, assuming that the occurrences of symbols in the original text are independent.
   c) Compare the average number of bits required to encode text using the Huffman code for the three symbols in part (a) and the Huffman code for the nine blocks of two symbols constructed in part (b). Which is more efficient?

31. Given $n+1$ symbols $x_1, x_2, \ldots, x_n, x_{n+1}$ appearing 1, $f_1, f_2, \ldots, f_n$ times in a symbol string, respectively, where f_j is the jth Fibonacci number, what is the maximum number of bits used to encode a symbol when all possible tie-breaking selections are considered at each stage of the Huffman coding algorithm?

*32. Show that Huffman codes are optimal in the sense that they represent a string of symbols using the fewest bits among all binary prefix codes.

33. Draw a game tree for nim if the starting position consists of two piles with two and three stones, respectively. When drawing the tree represent by the same vertex symmetric positions that result from the same move. Find the value of each vertex of the game tree. Who wins the game if both players follow an optimal strategy?

34. Draw a game tree for nim if the starting position consists of three piles with one, two, and three stones, respectively. When drawing the tree represent by the same vertex symmetric positions that result from the same move. Find the value of each vertex of the game tree. Who wins the game if both players follow an optimal strategy?

35. Suppose that we vary the payoff to the winning player in the game of nim so that the payoff is n dollars when n is the number of legal moves made before a terminal position is reached. Find the payoff to the first player if the initial position consists of
   a) two piles with one and three stones, respectively.
   b) two piles with two and four stones, respectively.
   c) three piles with one, two, and three stones, respectively.

36. Suppose that in a variation of the game of nim we allow a player to either remove one or more stones from a pile or merge the stones from two piles into one pile as long as at least one stone remains. Draw the game tree for this variation of nim if the starting position consists of three piles containing two, two, and one stone, respectively. Find the values of each vertex in the game tree and determine the winner if both players follow an optimal strategy.

37. Draw the subtree of the game tree for tic-tac-toe beginning at each of these positions. Determine the value of each of these subtrees.

a)

O	X	X
X	O	O
		X

b)

X	O	X
O	X	X
		O

c)

X		O
O	O	
X		X

d)

	O	X
	X	O
X	O	

38. Suppose that the first four moves of a tic-tac-toe game are as shown. Does the first player (whose moves are marked by Xs) have a strategy that will always win?

a)

X	O	X
	O	

b)

X	O	X
	O	

c)

	O	X
	X	
O		

d)

X		
X	O	
O		

39. Show that if a game of nim begins with two piles containing the same number of stones, as long as this number is at least two, then the second player wins when both players follow optimal strategies.

40. Show that if a game of nim begins with two piles containing different numbers of stones, the first player wins when both players follow optimal strategies.

41. How many children does the root of the game tree for checkers have? How many grandchildren does it have?

42. How many children does the root of the game tree for nim have and how many grandchildren does it have if the starting position is

a) piles with four and five stones, respectively.

b) piles with two, three, and four stones, respectively.

c) piles with one, two, three, and four stones, respectively.

d) piles with two, two, three, three, and five stones, respectively.

43. Draw the game tree for the game of tic-tac-toe for the levels corresponding to the first two moves. Assign the value of the evaluation function mentioned in the text that assigns to a position the number of files containing no Os minus the number of files containing no Xs as the value of each vertex at this level and compute the value of the tree for vertices as if the evaluation function gave the correct values for these vertices.

44. Use pseudocode to describe an algorithm for determining the value of a game tree when both players follow a minmax strategy.

11.3 Tree Traversal

Introduction

Links

Ordered rooted trees are often used to store information. We need procedures for visiting each vertex of an ordered rooted tree to access data. We will describe several important algorithms for visiting all the vertices of an ordered rooted tree. Ordered rooted trees can also be used to represent various types of expressions, such as arithmetic expressions involving numbers, variables, and operations. The different listings of the vertices of ordered rooted trees used to represent expressions are useful in the evaluation of these expressions.

Universal Address Systems

Procedures for traversing all vertices of an ordered rooted tree rely on the orderings of children. In ordered rooted trees, the children of an internal vertex are shown from left to right in the drawings representing these directed graphs.

We will describe one way we can totally order the vertices of an ordered rooted tree. To produce this ordering, we must first label all the vertices. We do this recursively:

1. Label the root with the integer 0. Then label its k children (at level 1) from left to right with $1, 2, 3, \ldots, k$.
2. For each vertex v at level n with label A, label its k_v children, as they are drawn from left to right, with $A.1, A.2, \ldots, A.k_v$.

Following this procedure, a vertex v at level n, for $n \geq 1$, is labeled $x_1.x_2.\ldots.x_n$, where the unique path from the root to v goes through the x_1st vertex at level 1, the x_2nd vertex at level 2, and so on. This labeling is called the **universal address system** of the ordered rooted tree.

We can totally order the vertices using the lexicographic ordering of their labels in the universal address system. The vertex labeled $x_1.x_2.\ldots.x_n$ is less than the vertex labeled $y_1.y_2.\ldots.y_m$ if there is an i, $0 \leq i \leq n$, with $x_1 = y_1, x_2 = y_2, \ldots, x_{i-1} = y_{i-1}$, and $x_i < y_i$; or if $n < m$ and $x_i = y_i$ for $i = 1, 2, \ldots, n$.