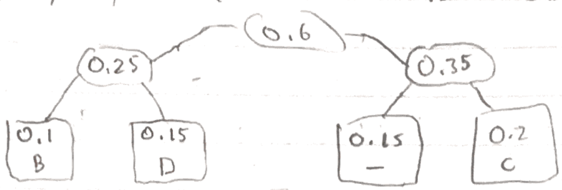
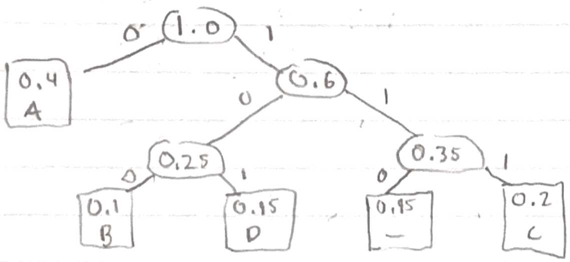
Homework 9

9.4 1a

symbol	A	B	C	D	_
frequency	0.4	0.1	0.2	0.15	0.15





A	B	C	D	_
0	100	111	101	110

b. A B A C A B A D

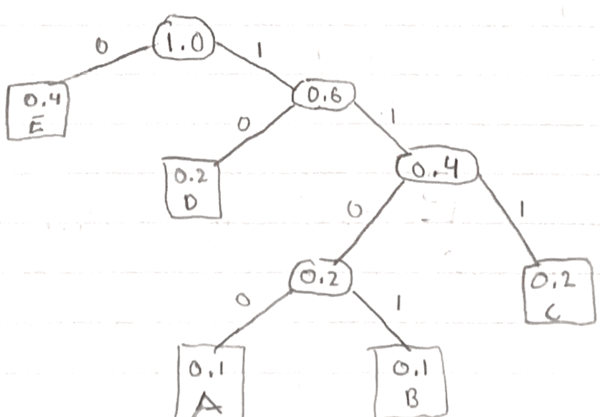
0 100 0 111 0 100 0 101

c. 1000 101 110 01010

B A D _ A D A

2.

symbol	A	B	C	D	E
probability	0.1	0.1	0.2	0.2	0.4



A	B	C	D	E
1100	1101	111	10	0

Average $= 4 \times 0.1 + 4 \times 0.1 + 3 \times 0.2 + 2 \times 0.2 + 1 \times 0.4$
$= 0.4 + 0.4 + 0.6 + 0.4 + 0.4$
$= \boxed{2.2}$

Variance $= (4-2.2)^2 \times 0.1 + (4-2.2)^2 \times 0.1 + (3-2.2)^2 \times 0.2 + (2-2.2)^2 \times 0.2 + (1-2.2)^2 \times 0.4$
$= (1.8)^2 \times 0.1 + (1.8)^2 \times 0.1 + (0.8)^2 \times 0.2 + (-0.2)^2 \times 0.2 + (-1.2)^2 \times 0.4$
$= 3.24 \times 0.1 + 3.24 \times 0.1 + 0.64 \times 0.2 + 0.04 \times 0.2 + 1.44 \times 0.4$
$= 0.324 + 0.324 + 0.128 + 0.008 + 0.576$
$= \boxed{1.36}$

5a. Algorithm HuffmanTree (w [0 ... n-1])
// Input : Array of weights
// Output : Huffman tree
// Create Priority Queue Q with one node tree and priories equal to elements of array of weights
While size of Q is more than one do
$T_l \leftarrow$ Minimum weight tree in Q
Delete minimum weight tree in Q
$T_r \leftarrow$ Minimum weight tree in Q
Delete minimum weight tree in Q
Create new tree T by combining T_l and T_r and the weight of this new tree will be equal to sum of weights of T_l and T_r.
Insert new tree T in Q
return T

b. The time efficiency class of the algorithm for constructing a Huff-man tree as a function of alphabet size is linear or O(n) because if the alphabet symbols are in a sorted order of their frequencies then deletion of smallest element in the priority queue will take 2(n-1) time and computation of weight of a new tree and insertion of new tree into priority queue will take (n-1) time. Implementation of this in a min heap will be O(nlogn) and for array or linked list will be $O(n^2)$

11.1 2. Since the Tower of Hanoi's recurrence relation for the number of moves is $M(n) = 2M(n-1) + 1$ $M(1) = 1$, which becomes $M(n) = 2^i M(n-i) + 2^i - 1$ after i substitutoons, then

$$\begin{aligned} M(n) &= 2^{n-1} M(n-(n-1)) + 2^{n-1} - 1 \\ &= 2^{n-1} M(1) + 2^{n-1} - 1 \\ &= 2^{n-1} + 2^{n-1} - 1 \\ &= 2^n - 1 \end{aligned}$$

thus this proves the recursive algorithm for Tower of Hanoi puzzer makes minimum number of disk moves needed to solve the problem, $2^n - 1$.

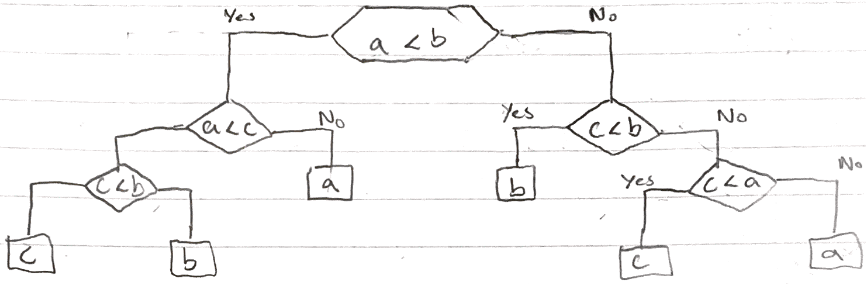
3a. The trivial lower-bound class for finding the largest element in an array is linear, because to find the largest element in the array, all elements need to be checked. It is tight because efficiency of the algorithm for this problem is $\Theta(n)$.

b. The trivial hower-bound class for checking completeness of a graph represented by its adjacency matrix is quadratic because for n vertices, the number of pairs of vertices of a graph is $\frac{n(n-1)}{2}$. It is tight because the algorithm for this problem checks all elements of the uppar triangular part of the adjacency matrix until either a '0' is found or all elements are checked.

4. No, we can not use the same information-theoretic argument as the one in the text for the number of questions in the guessing game to conclude that any algorithm for identifying the fake will need at least $\lceil \log_2 n \rceil$ weighings in the worst case. The problem can be solved with fewer weighings by dividing the coins into three subsets rather than two subsets with about the same number of coins each. The information-theoretic has to take into account that one weighing reduces uncertainty more than for the number-guessing problem because it has three rather than two outcomes

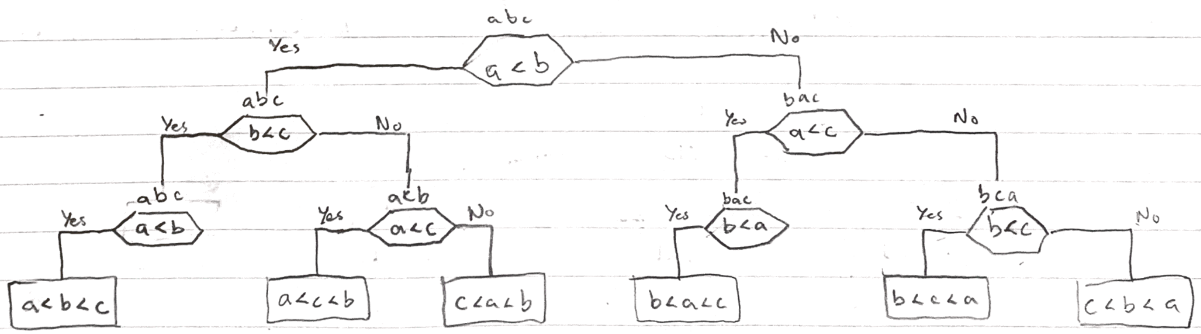
11.2 2a. The information-theoretic lower bound for comparison-based algorithm solving this problem is $\log_2(6)$ since the information-theoretic lower bound is $\log_2(\ell)$.

b.



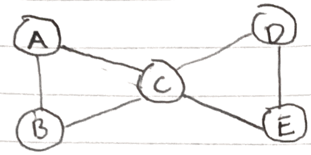
c. Quicksort is an algorithm that matches the lower bound since it can sort elements in $O(n\log n)$ time, and the lower bound of sorting / comparison sorting is $O(n\log n)$. It is $O(n\log n)$ because all possible orders are at leaves, thus the height is the lower bound and the height equals $h \in \log_2(n!) \in \Omega(n\log n)$.

3a.

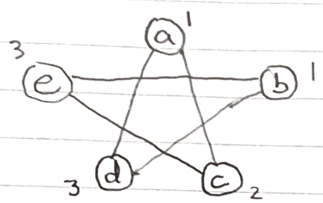


In the worst case and average case, the number of key comparisons in a three-element basic bubble sort decision tree is three, because for any of its inputs exactly three comparisons happen.

11.3 3b graph with an Eulerian circuit but without a Hamiltonian circuit



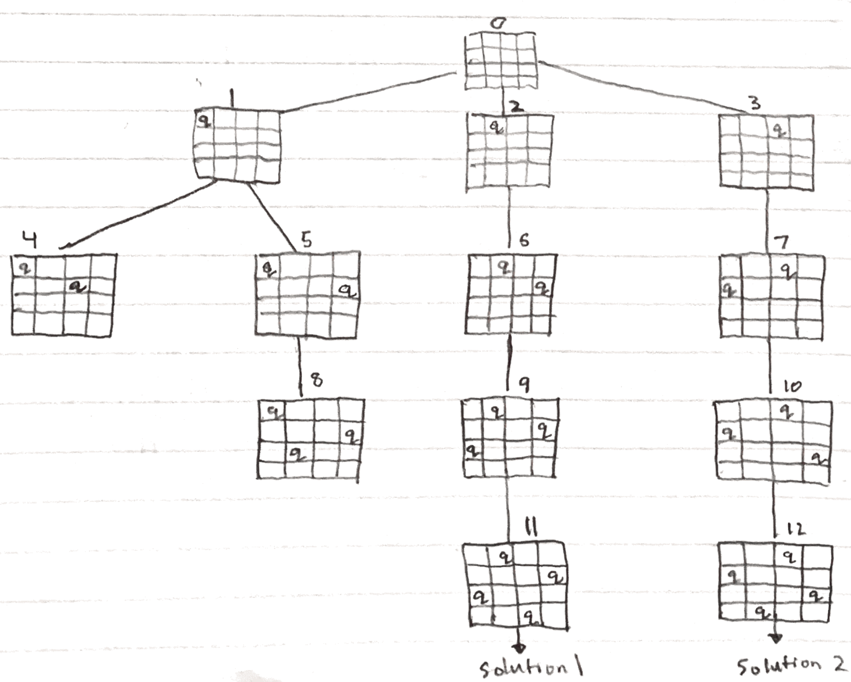
4b.



The chromatic number for this graph is 3

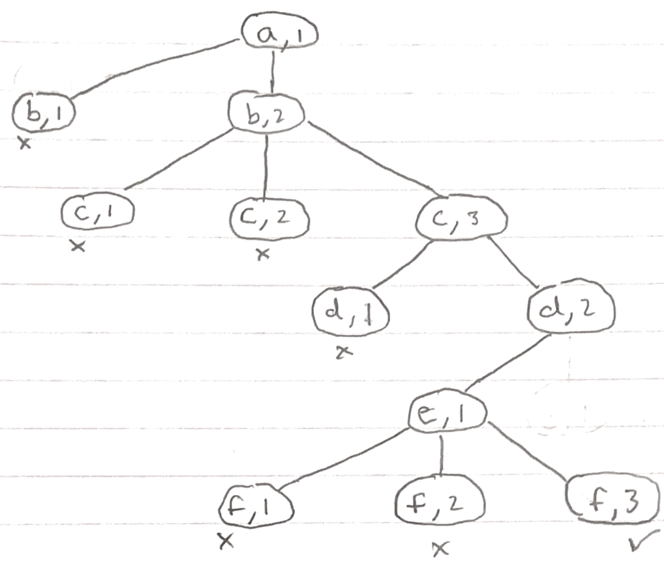
8. The partition problem can be reduced to the decision version of the knapsack problem by dividing the sum of n positive integers by 2 and setting that value to max weight/capacity. Then set each item's weights to their int value and each item's value to one. Finally run the knapsack problem to find a subset of items whose value is the largest. This will give you two subsets whose sum is half of the total sum of n positive integers that are not necessarily equal in size.

12.1 1a.



b. The board's symmetry can be used to find the second solution by reflection with respect to the middle verticle line that goes through the middle of the board.

6



Thus resulting graph...