Contents

Applied Mathematics 140 – Discrete Mathematics

Tutorials and Practice Classes: QUESTIONS

Printed in October 2003

ACKNOWLEDGEMENT: The content of this booklet originates from the efforts of Dr Chris Radford and Dr Zhuhan Jiang. I have reordered the topics, made some additions and some minor amendments. All errors are mine. -- Dr Tim Dalby

TUTORIAL 1
The Bubble Sort

Question 1

One of the most important and frequently used tasks in computer science is that of sorting a file. There are many methods of sorting. Perhaps the simplest – but not necessarily the best – is the **bubble sort**. It makes little difference what kind of items are in the list. For example they could be numbers or names. We shall consider a list of names. Our job is to put the list in alphabetical order.

The basic idea in the bubble sort is to pass through the list comparing each item with the next and interchanging them if they are out of order. If at the end of the pass the list is in the correct order, nothing further needs to be done. If not, another pass is made through the list and adjacent names are interchanged as necessary. The passes are repeated until no name is out of order. The process is indicated below. Observe how particular names bubble upward and others sink downward to their correct position. Indeed, after the first pass the last name in the list must be in its correct position. So if a second pass is necessary, we do not bother to compare the last item in the list with the one above it. After the second pass the last two items in the list must be in their correct position, and so on. Thus the process must stop after at most $n - 1$ passes.

Example

Original Lists	List after Pass 1	List after Pass 2
	6 exchanges done	*5 exchanges done*
Kris	Kris	Anita
Nicole	Anita	Amelia
Anita	Amelia	Kris
Amelia	Nicole	Deborah
Sandra	Deborah	Nicole
Deborah	Paula	Maria
Paula	Maria	Louise
Maria	Louise	Paula
Louise	Sandra	Sandra

(i) Complete the above process until the list is in alphabetical order.

The bubble sort algorithm is a particularly simple sorting algorithm, but there are other algorithms which in general involve fewer steps and which are therefore quicker to execute.

Questions marked by stars “*” in this booklet are *optional* and are usually difficult. Avoid such questions completely if you happen to experience some difficulties with them.

The bubble sort has its worst case when the list happens to be in reverse alphabetical order. We shall now examine this worst case.

Let L_n be a list of n names in reverse alphabetical order. In the sorting process, each comparison of adjacent names is called a step. Let $B(n)$ be the number of steps to put this list in correct order.

(ii) Let L_n be $a_1, a_2, \ldots, a_n$. Write down the list after one pass.

(iii) How many comparisons have been made in this first pass?

Observe that a_1 is now in its correct position, and $a_2, a_3, \ldots, a_n$ is in reverse alphabetical order. So this list of $n-1$ names requires $B(n-1)$ steps to put it in alphabetical order.

(iv) So $B(n) = B(n-1) + ???$.

(v) Observing that $B(1) = 0$, use iteration to find $B(10)$.

(vi) Using induction, prove that $B(n) = \frac{1}{2}(n^2 - n)$.

(vii) Use (vi) to verify your answer in (v).

TUTORIAL 2
Divide and Conquer

It is sometimes possible to divide a problem into smaller subproblems, solve the subproblems, and then combine their solutions to obtain the solution to the original problem. This general approach, which is a very common one in computer science, is called *divide and conquer.* Since the subproblems are usually of the same type as the original problem, a divide and conquer strategy can often be implemented as a recursive algorithm.

We shall meet this technique here in the context of the MERGE SORT. (You should compare the MERGE SORT with the BUBBLE SORT, which we studied in Tutorial 1.)

We begin with the task of merging two lists which have already been sorted. For example the two lists:

LIST 1	LIST 2
Chifley	Curtin
Fadden	Gorton
Forde	Hawke
Fraser	Howard
Holt	Menzies
Keating	Whitlam

Our *algorithm* for merging these two lists is as follows: At each step the elements at the top of each list are compared, and the one which is "smaller" (that is, first in the alphabetical ordering) is removed and put at the bottom of a combined list (which is empty to begin with). The process is continued until one list is empty. Then the remainder of the other list is put at the tail of the combined list.

So in the above example we begin by comparing "Chifley" and "Curtin ". "Chifley" is smaller, so it is removed and the combined list now contains "Chifley". Next we compare "Curtin" and "Fadden". "Curtin" is smaller, so it is removed and the combined list becomes:

Combined List
Chifley
Curtin

(i) How many comparisons are needed to merge LIST 1 and LIST 2?

(ii) More generally, if we are given two sorted lists, one with n elements and the other with m elements, prove that the total number of comparisons required to merge these two lists is not more than $m + n - 1$.

Suppose now that we are given the following list containing 12 members and asked to sort it:

Keating, Fadden, Fraser, Holt, Forde, Chifley, Hawke, Curtin, Whitlam, Gorton, Menzies, Howard

We adopt the divide and conquer technique. First, divide the group of 12 into two groups of 6. We shall sort each of these groups of 6, and then merge the two sorted groups.

How do we sort each group of 6? We divide it into 2 groups of 3, sort these and then merge.

How do we sort a group of 3? We divide it into two groups, one with 2 elements and the other with 1 element, sort the group with 2 elements and merge it with the group with 1 element.

How do we sort a group of 2 elements? Divide it into two groups with 1 element and then merge.

So now we know how to sort the list with 12 elements!!

(iii) Show that, using this procedure, any list with 3 elements can be sorted in not more than 3 comparisons.

(iv) Show that, using this procedure, any list with 6 elements can be sorted in not more than 11 comparisons.

(v) Show that, using this procedure, any list with 12 elements can be sorted in not more than 33 comparisons.

Recall that the BUBBLE SORT required $\frac{(n^2-n)}{2}$ comparisons in the worst case. So if $n = 12$, then 66 comparisons may be needed. The MERGE SORT, though not as easy to describe, is significantly better.

We shall now analyze the general case of MERGE SORT. To do this, we need some notation. Let $T(n)$ be the number of comparisons needed to sort a list with n elements using the MERGE SORT.

(vi) Prove that $T(2^n) < 2T(2^{n-1}) + 2^n$, for any positive integer n.

[Hint: Consider a list with 2^n elements, and look at the first step of the MERGE SORT; that is, dividing the list into two groups each with 2^{n-1} elements.]

(vii) Using induction and (vi), prove that $T(2^n) < n.2^n$, for any positive integer n.

(viii) For any real number a, let $\lfloor a \rfloor$ denote the largest integer not greater than a. (e.g., $\lfloor 2.3 \rfloor = 2$, and $\lfloor -4.6 \rfloor = -5$.) Show that $a \leq 2^{\lfloor \log a \rfloor + 1}$, for any positive real number a.

(ix) Using (vii), (viii), and the fact that $m \leq k$ implies $T(m) \leq T(k)$, prove that $T(m) \leq 4m \log m$, for any integer $m \geq 2$.

(x) Deduce that T is $\mathcal{O}(n \log n)$.

(xi)* Prove that T is not $\mathcal{O}(n)$.

(xii)* Prove that, in the notation of Tutorial 1, B is not $\mathcal{O}(n \log n)$.

In Tutorial 1 we saw that, in the worst case, the BUBBLE SORT is $\mathcal{O}(n^2)$. We have now shown that the MERGE SORT is $\mathcal{O}(n \log n)$, which is significantly better. Part (xii) shows that the BUBBLE SORT is not $\mathcal{O}(n \log n)$.

Indeed it can be shown that no (sequential) sorting algorithm can do better than $\mathcal{O}(n \log n)$ – but this does not mean that there are not better sorting algorithms.

We have ignored many important questions which are central to implementing a sorting algorithm. For example, how much storage is needed to implement either the MERGE SORT or the BUBBLE SORT. You should give some thought to what other features of a sorting algorithm should be considered when evaluating it.

TUTORIAL 3
Insertion Sort and Selection Sort

Question 1

The INSERTION SORT is one of the simplest sorting algorithms. Consider the following:

LIST 1	List 2	List 3	List 4
	(1 comparison)	(2 comparisons)	(2 comparisons)
Britain	Britain	Austria	Austria
Sweden	Sweden	Britain	Britain
Austria	Austria	Sweden	Hong Kong
Hong Kong	Hong Kong	Hong Kong	Sweden
Thailand	Thailand	Thailand	Thailand
Norway	Norway	Norway	Norway
Canada	Canada	Canada	Canada
United States	United States	United States	United States
Spain	Spain	Spain	Spain
Taiwan	Taiwan	Taiwan	Taiwan
Singapore	Singapore	Singapore	Singapore
India	India	India	India
Australia	Australia	Australia	Australia
New Zealand	New Zealand	New Zealand	New Zealand

List 1 is the given list. At the i^{th} step we regard the first i elements as already sorted and put the $(i+1)^{st}$ element in its correct position among the first i elements. In the above example this is done as follows:

In each list we have sorted down to the element underlined. List 1 is the given list, so we regard "Britain" as being in its correct position. Now we look at the second member of the list, "Sweden", and compare it with "Britain". It is later in the alphabet, so in List 2 we put "Sweden" in second position, and we have sorted the first two elements. We have used 1 comparison.

Next we compare "Austria" with "Sweden". It comes earlier in the alphabet. So we compare "Austria" with "Britain". Once again it comes earlier in the alphabet, so "Austria" moves to the top of the list. Thus in List 3 we have sorted the first 3 members. This time we used 2 comparisons.

Next we compare "Hong Kong" with "Sweden". It comes earlier in the alphabet, so we compare "Hong Kong" with "Britain". It comes later in the alphabet. So in List 4 we have sorted the first 4 members, and used 2 more comparisons.

(i) Continue the insertion sort by writing down LIST 5, LIST 6, and LIST 7, recording at each stage the number of comparisons performed.

The INSERTION SORT has its worst case (that is the one requiring the most comparisons) when the list is in reverse alphabetical order. (You should NOT think that every sort has its worst case when the list is in reverse alphabetical order. Some sorts have their worst case when the given list is already in alphabetical order.)

Let $L_n = \{a_1, a_2, \ldots, a_n\}$ be a list of names in `reverse alphabetical order`. Let $I(n)$ be the number of comparisons required to put this list in alphabetical order.

(ii) Apply the INSERTION SORT and write down LIST 1, LIST 2, LIST 3, and LIST 4. Each time indicate how many comparisons are performed.

(iii) Prove that $I(n) = n(n-1)/2$.

(iv) Prove that I is $\mathcal{O}(n^2)$.

Question 2

The SELECTION SORT is another very simple sorting algorithm. Consider the following:

<u>LIST 1</u>	<u>List 2</u>	<u>List 3</u>	<u>List 4</u>
Britain	<u>Australia</u>	Australia	Australia
Sweden	Sweden	<u>Austria</u>	Austria
Austria	Austria	Sweden	<u>Britain</u>
Hong Kong	Hong Kong	Hong Kong	Hong Kong
Thailand	Thailand	Thailand	Thailand
Norway	Norway	Norway	Norway
Canada	Canada	Canada	Canada
United States	United States	United States	United States
Spain	Spain	Spain	Spain
Taiwan	Taiwan	Taiwan	Taiwan
Singapore	Singapore	Singapore	Singapore
India	India	India	India
Australia	Britain	Britain	Sweden
New Zealand	New Zealand	New Zealand	New Zealand

List 1 is the given list. First we find the element which is earliest in the alphabet, and interchange it with the element at the top of the list. It is found by comparing “Britain” and “Sweden”and saying “Britain” is earlier; so we compare “Britain” with “Austria”– - “Austria” is earlier; next we compare “Austria” and “Hong Kong” — “Austria” is earlier; next we compare “Austria” and “Thailand”— “Austria” is earlier; we continue until we have compared “Austria” and “Australia”, and found that “Australia” is earlier. Then we compare “Australia” and “New Zealand” and find “Australia” is earlier. At this stage we know that “Australia” is first in the alphabetical ordering. So in LIST 2, “Australia” and “Britain” are interchanged. Next we find the second earliest element and exchange it with "Sweden". We find the second earliest element by the same process, except we begin by comparing "Sweden" and "Austria", and move down LIST 2. So in LIST 2 the top element is in its correct position. In LIST 3 the top 2 are in their correct position.

(i) How many comparisons were done in proceeding from LIST 1 to LIST 2?

(ii) How many comparisons were done in proceeding from LIST 2 to LIST 3?

(iii) How many comparisons were done in proceeding from LIST 3 to LIST 4?

(iv) How many comparisons are used in the entire sorting process.

(v) Have you noticed that the number of comparisons is INDEPENDENT of the order of the given list? Is this a good feature?

(vi) Let $S(n)$ be the number of comparisons needed to sort a list with n elements by SELECTION SORT. Find a "closed form expression" for $S(n)$, and hence verify that S is $\mathcal{O}(n^2)$.

TUTORIAL 4
Quicksort

Question 1

In previous tutorials we have studied BUBBLE SORT, MERGE SORT, INSERTION SORT, and SELECTION SORT. In this tutorial we study one of the best sorting algorithms, namely QUICKSORT. It is very fast – indeed, its `average running time` is less than that of all other known sorting algorithms.

The algorithm for QUICKSORT is a little difficult to describe, so we shall simplify the analysis by considering only the number of comparisons required, and ignoring the number of interchanges of elements needed. (*If you study Computer Science, you will meet QUICKSORT in its full glory. For those who cannot wait, see "Data Structures and Algorithms" by A. Aho, J.E. Hopcroft, and J.D. Ullman [Addison-Wesley, 1983] and "Handbook of Algorithms and Data Structures" by G.H. Gonnet [Addison- Wesley, 1984].)* So this tutorial should be regarded as an *introduction* to QUICKSORT.

In Tutorial 2 you met the technique called DIVIDE AND CONQUER – this is the method in which a problem is solved by dividing it into smaller subproblems, solving the subproblems, and then combining their solutions to produce the solution to the original problem. This technique is very useful when the subproblems are of the same type as the original problem so that a recursive approach is possible. This is the case for QUICKSORT.

The idea behind QUICKSORT is easy: to QUICKSORT a list L, select an arbitrary element x of L and call this element the *pivot*. Then rearrange the list so that all of the entries that are smaller than x precede it, and all of the entries that are bigger than x follow it. As a result of this rearrangement x is in its correct position. Next let L_1 be the list of entries that are greater than x and L_2 the list of entries that are less than x. Since x is in its correct position, all we need to do is to sort L_1 and L_2.

Thus we have reduced the problem of sorting the list L to that of sorting the smaller lists L_1 and L_2 . That is we are dividing and conquering. So we repeat this procedure on L_1 and L_2, and proceed recursively.

We shall now look at an example. To make the analysis easier – rather than choosing a random element of the list as the pivot <u>we shall always choose the first element of the list.</u>

LIST 1	LIST 2	LIST 3	LIST 4
EASTERN SPINEBILL	CURRAWONG	COCKATOO	*butcherbird**
currawong	cockatoo	butcherbird	*cockatoo**
rosella	butcherbird	crow	*crow**
magpie	crow	*currawong**	*currawong**
wedge tailed eagle	*eastern spinebill**	*eastern spinebill**	*eastern spinebill**
cockatoo	ROSELLA	MAGPIE	FAIRY WREN
wattlebird	magpie	fairy wren	galah
butcherbird	wedge tailed eagle	galah	kookaburra
crow	wattlebird	kookaburra	*magpie**
fairy wren	fairy wren	*rosella**	*rosella**
galah	galah	WEDGE TAILED EAGLE	*wattlebird**
kookaburra	kookaburra	wattlebird	*wedge tailed eagle**

LIST 1 is the given list of common Australian birds. As indicated above, we choose the top element "eastern spinebill" as the pivot. (To indicate what we are doing, pivots are typed in upper case letters.) So we compare it with all other elements in LIST 1. There are 4 entries which come earlier in the alphabet. So "eastern spinebill" is put in position 5 in LIST 2 and the 4 smaller entries are put above it (in the same order as they were in LIST 1). The other 7 entries are put after "eastern spinebill" in the same order as they appeared in LIST 1. [The precise manner of implementing this on a computer is omitted so as to simplify our analysis.] (We use asterisks to indicate elements which we know are in their final position. So in LIST 2 "eastern spinebill" is followed by an asterisk.)

Next we sort LIST 2. BUT LIST 2 is really two lists, namely the list of 4 elements above "eastern spinebill" and the list of 7 elements under it. ("eastern spinebill" is already in its final position.) Our pivot in the top list of 4 elements is "currawong". There are 3 entries in the top list which are smaller than it. So in LIST 3 "currawong" appears in position 4. The 3 smaller elements appear above it. The pivot for the bottom list of 7 elements in LIST 2 is "rosella". Four of the entries below it are smaller than it. So in LIST 3 these 4 entries are above it; that is "rosella" appears in position 10, and the other 2 entries appear below it. So in LIST 3, "currawong", "eastern spinebill", and "rosella" are in their final positions.

Now we have 3 sublists of LIST 3 to sort. They have pivots "cockatoo", "magpie", and "wedge tailed eagle" ... and so on.

(i) Complete the QUICKSORT of this example.

QUICKSORT, as described above, has its worst case when the given list is already in alphabetical order.

Let LIST 1 be the list $a_1, a_2, \dots, a_n$ and assume that it is in alphabetical order. We now QUICKSORT this list.

(ii) Write down LIST 2 and LIST 3 which are obtained from this LIST 1 and indicate how many comparisons are used.

(iii) How many comparisons are used in sorting LIST 1 entirely using QUICKSORT?

We see from the answer to (iii) that, in the worst case, QUICKSORT uses $T(n)$ comparisons, where T is $\Theta(n^2)$. BUT we shall show that on the `average`, QUICKSORT uses $Q(n)$ comparisons, where Q is $\mathcal{O}(n \log n)$. We now turn to the average case.

Our first task is to define what we mean by the `average number of comparisons` needed to sort a list with n elements.

Let $S = \{b_1, b_2, \ldots, b_n\}$ be a set of n words.

(iv) How many different orderings of this set are there?

We can use QUICKSORT to sort each of these ordered sets and note the number of comparisons used. We define $Q(n)$ to be the average of these numbers; that is, $Q(n)$ is obtained by adding the number of comparisons used in QUICKSORTING all these ordered sets and dividing by $n!$.

(v) Evaluate $Q(3)$ by considering the 3–element set $\{banana,\ pear,\ apple\}$.

(vi) Verify that $Q(n) \leq \frac{3}{2} n \log n$, for $n = 1, 2, 3$.

Question 2

QUICKSORT the following list into alphabetical order:

Joan, Gary, Chris, Sid, Norm, Gwenda .

The rest of this tutorial is for OPTIONAL reading.

Returning now to our set S with n elements, we see that the probability that the pivot b_1 is the i^{th} smallest element is $\frac{1}{n}$, since all n possibilities are equally likely. If b_1 is the i^{th} smallest element then, after $n-1$ comparisons, we know the $i-1$ elements which are smaller than it and the $n-i$ elements which are bigger than it. Next we have to sort the set with $i-1$ elements and the set with $n-i$ elements. Thus we obtain

$$Q(n) = n - 1 + \frac{1}{n} \sum_{i=1}^{n} [Q(i-1) + Q(n-i)].$$

(a) Prove that $\sum_{i=1}^{n} Q(n-i) = \sum_{j=0}^{n-1} Q(j)$.

(b) Prove that $\sum_{i=1}^{n} Q(i-1) = \sum_{j=0}^{n-1} Q(j)$.

(c) From (a) and (b) we obtain that

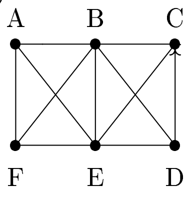
$$Q(n) = n - 1 + \frac{2}{n} \sum_{j=0}^{n-1} Q(j).$$

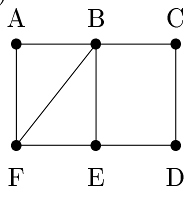
Using this, evaluate $Q(4)$ and $Q(5)$.

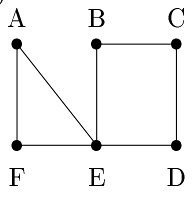
TUTORIAL 5
Features of Some Graphs

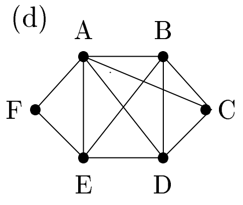
Question 1

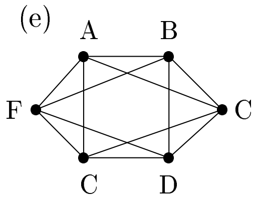
Without trying to find a Eulerian path, state which of the following graphs are Eulerian, which are not Eulerian but contain an Eulerian path, and which ones do not contain an Eulerian path.

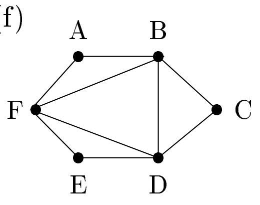
(a)


(b)


(c)


(d)


(e)


(f)


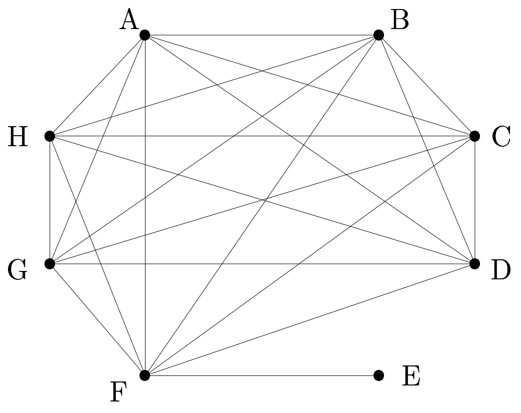
Question 2

(i) For each graph (b), (c) and (f) in Question 1:

(a) write down an ordered edge list, and

(b) use Fleury's algorithm to construct an Eulerian path.

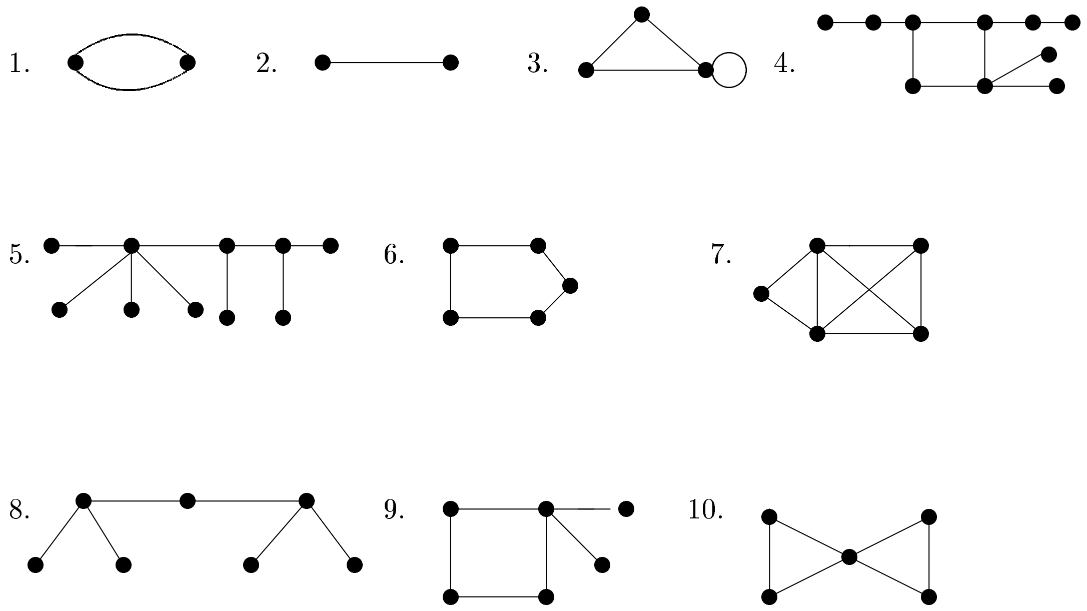
(ii) Repeat (a) and (b) for the following graph



Question 3

(i) Complete the <u>first four columns</u> of the following table for the examples below. See if you can guess some pattern based on the degree of the vertices.

	Number of edges	Number of vertices	Number of vertices of odd degree	Number of vertices of even degree	N_G
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					



(ii) For a graph G with V vertices of degree $d_1, d_2, \cdots, d_v$, define N_G to be the sum of the degrees of all the vertices; i.e. $N_G = d_1 + d_2 + \cdots + d_v$. Use this definition to complete Column 5 in the table.

(iii) Guess a simple formula for N_G, then prove that your guess is correct.

(iv) Use (iii) to prove that there is an even number of vertices of odd degree. (This should have been your guess in (i).)

Question 4

A simple graph is one without multiple edges or loops. What is the minimum and maximum number of edges a simple graph with V vertices can have?

Would your answer change if the graph is required to be connected?

TUTORIAL 6
Kruskal's Algorithm

Recall the algorithm, covered in the lectures notes, for finding a spanning tree for a connected graph with vertices $v_1, v_2, \ldots, v_n$ and edges $e_1, e_2, \ldots, e_m$.

(i) (a) For $i = 1$ to n, let $L(i) = i$. Let $E = \emptyset$.

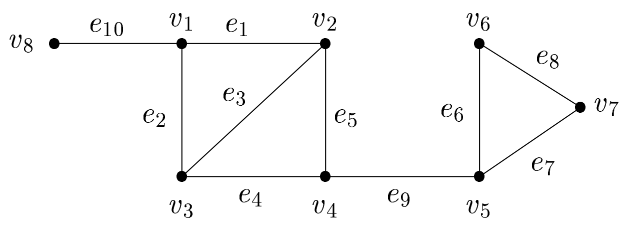
(ii) For $r = 1$ to m, perform the following steps.

  (a) If $e_r = \{v_i, v_j\}$ with $L(i) \neq L(j)$, then let $E = E \cup \{e_r\}$, and let $L(i)$, $L(j)$, and any $L(k)$ such that $L(k) = L(i)$ or $L(k) = L(j)$, be replaced by $\min\{L(i),\ L(j)\}$.

  (b) If $e_r = \{v_i, v_j\}$ with $L(i) = L(j)$ then leave $E, L(i)$ and $L(j)$ as they are.

Question 1

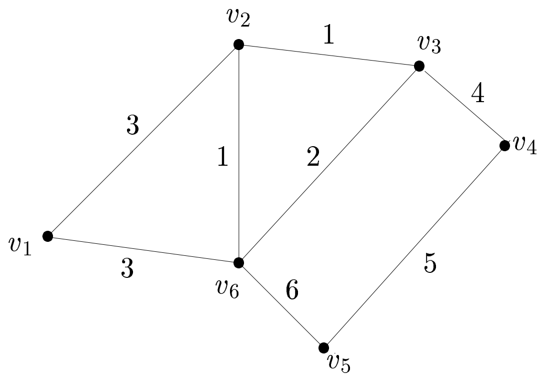
Use the algorithm above to find a spanning tree for the following graph. Set out your answer in a table, as in the lecture notes.



Kruskal's Algorithm

Consider the problem of building a communications network linking six centres, where the cost of links is not constant.

The following is a graph representing the problem. The vertices represent the centres, and the edges represent the links connecting the centres. The number attached to each edge is the amount (in \$1000's) of the cost of building the link. This cost factor is called the *weight* of the edge.



Obviously not all of the links need be built, but which ones should be built so that each centre is able to communicate with any other centre, and the network can be built as cheaply as possible?

We are able to solve this problem by finding a spanning tree of *minimal weight* for the graph. We do this using a modification of the above algorithm.

Before implementing the algorithm, we order the edges in increasing weight. So, for instance, the edge list for the graph above could be

e_i	$w(e_i)$
$e_1 = \{v_2, v_3\}$	$w(e_1) = 1$
$e_2 = \{v_2, v_6\}$	$w(e_2) = 1$
$e_3 = \{v_3, v_6\}$	$w(e_3) = 2$
$e_4 = \{v_1, v_2\}$	$w(e_4) = 3$
$e_5 = \{v_1, v_6\}$	$w(e_r) = 3$
$e_6 = \{v_3, v_4\}$	$w(e_6) = 4$
$e_7 = \{v_4, v_5\}$	$w(e_7) = 5$
$e_8 = \{v_5, v_6\}$	$w(e_8) = 6$

We then consider each edge $e_1, e_2, \ldots, e_8$ in order.

An extra column is needed in the table for the weight tally at each step.

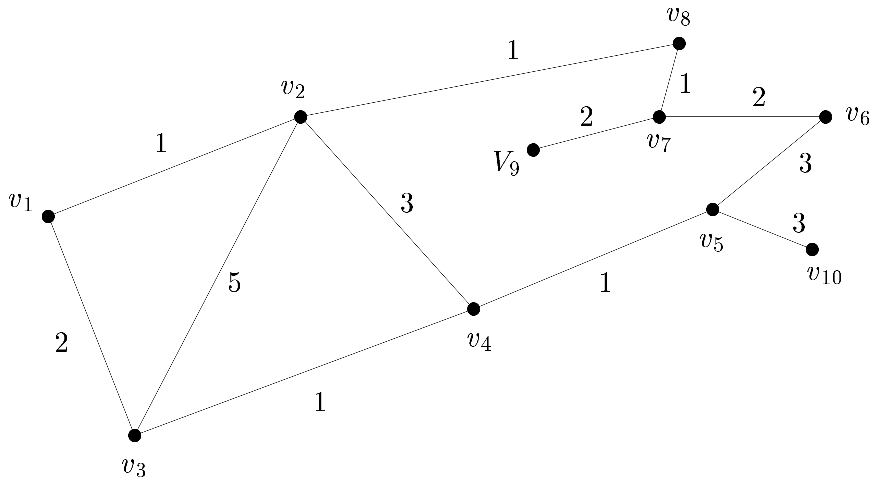
Question 2

Use Kruskal's algorithm to find a minimal spanning tree for the graph.

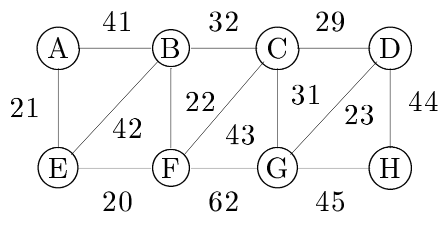
Question 3

Use Kruskal's algorithm to find minimal spanning trees for the following graphs.

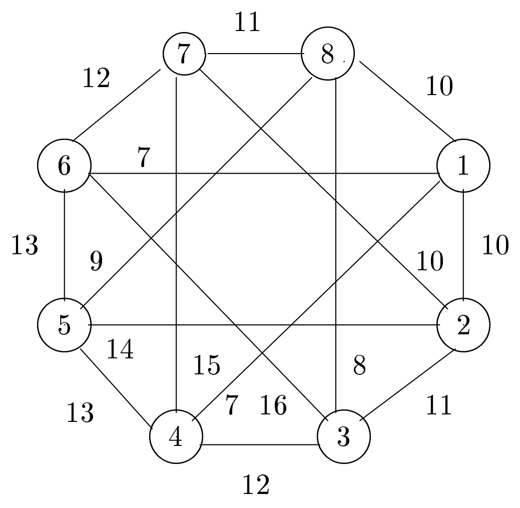
(a)



(b)



(c)



Question 4

Let G be a connected graph. What can you say about an edge of G which appears in

(i) every spanning tree?

(ii) no spanning tree?

Justify your answers.

TUTORIAL 7
Boolean Algebra and Voting Machines

Question 1

Two terms in a Boolean expression are adjacent if they differ by exactly one literal, and this literal appears in one term, while its complement appears in the other.

e.g. xyz and xyz' are adjacent,

but xyz and $x'yz'$ are not adjacent.

Write down all pairs of adjacent terms in the following Boolean expressions.

(i) $xyz + x'yz + x'y'z' + x'yz'$

(ii) $wx'yz + wxy'z + w'x'yz + wx'y'z + w'xyz'$

(iii) $w'x'y'z + wx'y'z + wxyz' + wxy'z + wxy'z' + wxyz$

(iv) $wxy'z + w'x'y'z + w'y'z + wy'z + w'xy'z + wxyz$

Question 2

Two adjacent terms in 1(i) are xyz and $x'yz$. We can simplify the sum of these terms by using the properties of the Boolean algebra (see B1 – B5 of the 2nd Lecture). For example,

$$\begin{aligned} xyz + x'yz &= (x+x')yz && \text{using } (f) \\ &= 1yz && \text{using } (i) \\ &= yz && \text{using } (h) \end{aligned}$$

Use the properties of the Boolean algebra to simplify the Boolean expressions given in Question 1.

Question 3

(i) Draw a switching circuit for the expression given for the Boolean function $f : S \times S \times S \to S$ given by $f(x, y, z) = xyz + xyz' + xy'z' + x'yz'$.

(ii) Simplify this expression for $f(x, y, z)$.

(iii) Draw a switching circuit for the simplified expression.

Question 4

Designing Electronic Voting Machines

Suppose there is a committee of four people, each with a switch to record their vote, say w, x, y and z. The voting rights are weighted so that

w has a block of 5 votes,

x has a block of 4 votes, and

y and z each have a block of 3 votes.

A motion will be passed if it gains at least half of the votes, and the switching circuit should be closed if the motion is passed.

(i) Complete the following voting table.

				Votes	Votes	Circuit
w	x	y	z	for	against	condition
0	0	0	0	0	15	0
0	0	0	1	3	12	0
0	0	1	0	3	12	0
0	0	1	1	6	9	
0	1	0	0	4	11	
0	1	0	1	7	8	
0	1	1	0	7	8	
0	1	1	1	10	5	
1	0	0	0	5	10	
1	0	0	1	8	7	
1	0	1	0	8	7	
1	0	1	1	11	4	
1	1	0	0	9	6	
1	1	0	1	12	3	
1	1	1	0	12	3	
1	1	1	1	15	0	

(ii) From the table, write down the corresponding Boolean expression.

(iii) Draw the switching circuit for this Boolean expression.

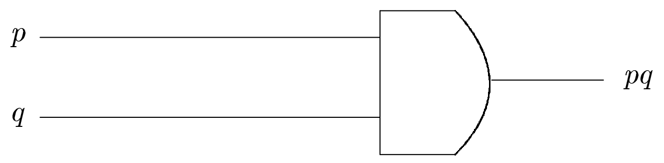
(iv) Simplify the Boolean expression found in (ii).

(v) Draw a simplified circuit for the electronic voting machine.

TUTORIAL 8
Logic Gates and Digital Adders

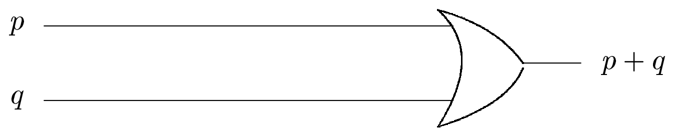
Computers contain silicon chips and printed on these chips are logic circuits. We will design 'small' logic circuits, a silicon chip may have thousands of such 'small' circuits. The basic building blocks of a logic circuit are called logic gates. These logic gates carry out the various logical operations on their inputs, we will define five of them.

(a)



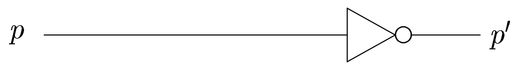
'and' gate (forms $p \wedge q$ from p, q)

(b)



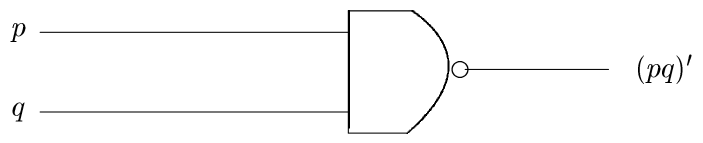
'or' gate (forms $p \vee q$)

(c)



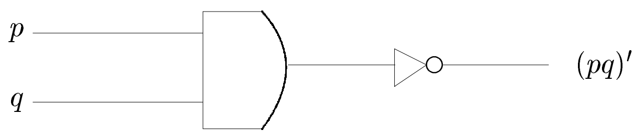
'not' or 'inverter' or 'complement' gate.

(d)

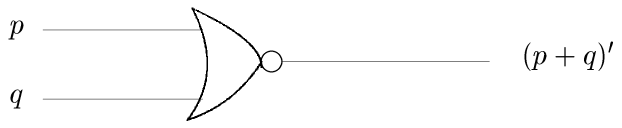


'nand' gate ('not and')

this is equivalent to

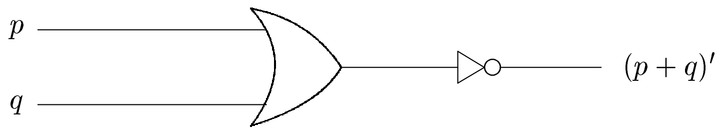


(e)



'nor' gate ('not or')

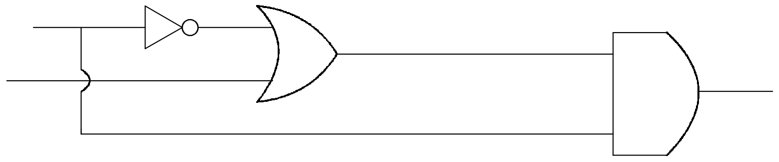
this is equivalent to



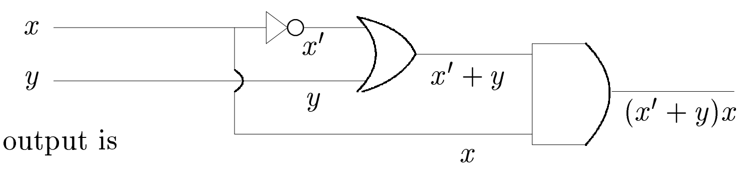
To calculate the output from a logic circuit or to design a logic circuit implementing a given logical proposition we use the machinery of Boolean algebra.

Examples

(i). Calculate the output from



Solution: We simply work our way from the left (input) to the right (output) putting in the various outputs according to (a) to (e) above.



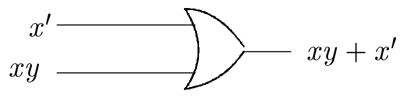
the output is

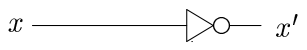
$$\begin{aligned}(x' + y)x &= x'x + xy \\ &= xy \text{ (by axiom } (j)).\end{aligned}$$

(ii). Give the gate implementation of

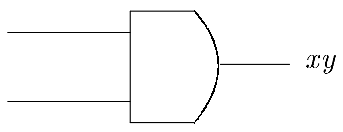
$$xy + x'$$

Solution: We work backwards from the output:
Firstly,

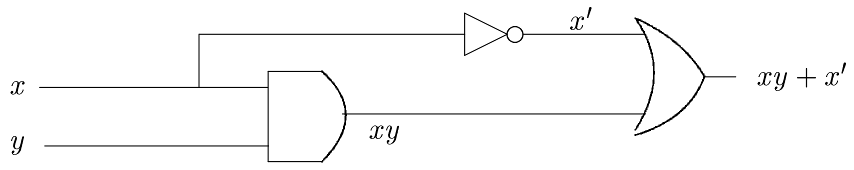


Now x' is produced by 

and xy by

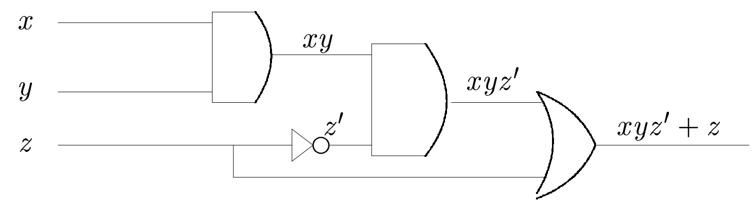


Putting this together we have



(iii). Give the gate implementation of $xyz' + z$.

Solution:



Question 1

Write down minimal expressions for the following functions, and give the gate implementation for the outputs.

(a)

x	y	output
0	0	1
0	1	0
1	0	0
1	1	1

(b)

x	y	z	output
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

(c)

x	y	z	output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Question 2

Half Adders Consider adding two binary digits x and y (each could be 0 or 1) and expressing their sum as a two-digit binary number $(cs)_2$.

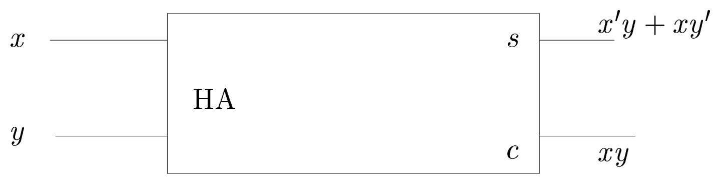
(a) Fill in the following tables for s and c.

x	y	c
0	0	
0	1	
1	0	
1	1	

x	y	s
0	0	
0	1	
1	0	
1	1	0

(b) Give the gate implementations for the outputs s and c.

(c) Give one gate implementation that inputs x and y and outputs s and c.
The circuit you have just drawn is called a half-adder and is denoted by



Question 3

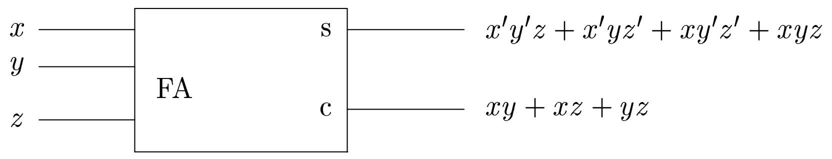
Full Adders Suppose you wanted to design a logic circuit for the addition of three binary digits x, y and z. Let $x + y + z = (cs)_2$.

(a) Fill in the following table.

x	y	z	c	s
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

(b) Using half adders, give the gate implementation for the circuit that inputs x, y and z, out outputs c and s.

The circuit you have just drawn is called a full adder and is denoted by



*(c) Full adders are used in performing the operation of adding 2 multiple-digit binary numbers. Give the gate implementation for the addition of two two-digit binary numbers.

Question 4

(a) Give the gate implementation of the ‘nand’ gate using only ‘and’, ‘or’ and ‘not’ gates.

(b) Give the gate implementation of the ‘nor’ gate using only ‘and’, ‘or’ and ‘not’ gates.

Question 5

Give the gate implementation of the Boolean expression $y + (xz)'$ using

(a) only nand gates;

(b) only nor gates.

TUTORIAL 9
More on Karnaugh Maps

In general, when we use Karnaugh maps, we will want to

(a) use the least number of adjacency circles to cover all the 1's (to minimize the number of product terms);

(b) include each 1 in the largest possible adjacency circle (to minimize the number of literals);

(c) Condition (a) takes precedence over (b).

These conditions are satisfied if we follow the algorithm of in the lecture notes.

Question 1

Use a Karnaugh map to find a minimal representation for the Boolean expression

$$w'xyz + wxyz' + wx'yz' + w'x'yz' + wx'y'z' + w'x'y'z' + w'xy'z' + wx'y'z$$

Question 2

Use a Karnaugh map to find a minimal representation for the Boolean expression

$$wxyz + wx'yz + w'xyz + wxyz' + wx'yz' + w'x'yz' + wx'y'z' + w'xy'z' + wx'y'z + w'x'y'z$$

Question 3

(a) Use a Karnaugh map to find <u>two</u> minimal representations for the Boolean expression

$$w'xyz + wx'y'z' + wxy'z + wx'y'z + w'xy'z$$

(b) Explain why these <u>equivalent</u> expression are both minimal representations.

Question 4

A 4×8 grid is required when using a Karnaugh map to minimize a Boolean expression in 5 variables.

(a) Complete the labeling of the Karnaugh map.

	vwx	vwx'			$v'w'x$			
yz								
						Y		

Check that (a) is correct before going on!!

(b) The double line is a line of symmetry. Mark all boxes adjacent to **Y**.

(c) Use a Karnaugh map to minimize the following Boolean expression:

(i)

$$vwxyz + vwxyz' + vw'x'yz' + v'w'x'yz' + v'wxyz' + vwx'y'z' + vw'x'y'z' + v'w'x'y'z'$$

$$+ v'wx'y'z' + vwx'y'z + vw'x'y'z + v'w'x'y'z + v'wx'y'z + v'wxy'z$$

(ii)

$$vwx'yz + vw'x'yz + v'w'x'yz + v'wx'yz + vwxyz' + vw'x'yz' + v'wxyz'$$

$$+ vwxy'z' + vw'x'y'z' + v'w'x'y'z' + v'wxy'z' + vw'x'y'z$$

PRACTICE CLASS 1
Sets and Mathematical Induction

Question 1

Let A, B and C be subsets of set U. Show

a) $A - B = A \cap B'$.

b) $(A \cup B) - (C - A) = (A \cup B) \cap (C' \cup A)$,
i.e. represent the l.h.s. without the use of the difference "–" .

c) $(A \cup B) - (C - A) = A \cup (B - C)$.

Question 2

Let set $A = \{1, 2\}$.

a) Give the power set $\mathcal{P}(A)$ of A.

b) Give the Cartesian product $A \times A$.

c) Give the Cartesian product $\mathcal{P}(A) \times A$.

Question 3

Write the following in $\sum$-notation:

a) $1 + 2 + \ldots + 10$

b) $1 + 3 + \ldots + (2n - 1)$

c) $1^3 + 2^3 + \ldots + k^3$

d) $4^2 + 6^2 + \ldots + k^2$

Question 4

Evaluate the following sums,

a) $\sum_{k=3}^{6} k^2$

b) $\sum_{n=2}^{4} \frac{1}{n}$

c) $\sum_{i=0}^{3} 3^i$

d) $\sum_{m=2}^{5} \frac{m}{2}$

Question 5

Prove by induction that $1 + 3 + 9 + \ldots + 3^{n-1} = \frac{3^n - 1}{2}$ for each positive integer n.

Question 6

Prove by induction that $\displaystyle\sum_{i=1}^{n} i^3 = \left[\frac{n(n+1)}{2}\right]^2$, for all integers $n \geq 1$.

Question 7

Prove by induction that $\displaystyle\sum_{j=1}^{n} \frac{1}{j(j+1)} = \frac{n}{n+1}$, for all integers $n \geq 1$.

Question 8

Prove by induction that $2n + 1 < 2^n$, for all integers $n \geq 3$.

Question 9

Use the result in Question 8 and induction to prove that $n^2 < 2^n$, for all integers $n \geq 5$.

Question 10

Prove by induction that $4^n - 1$ is divisible by 3, for integers $n \geq 1$.

Question 11

Prove by induction that $n^3 - n$ is divisible by 6, for all integers $n \geq 1$.

Question 12

Prove by induction that for every $n \in \mathbf{N}$, $3^n + 7^n - 2$ is divisible by 8.

PRACTICE CLASS 2
Horner's Algorithm, $\mathcal{O}$ the Θ Notation

Question 1

i) Write $f(x) = 4x^3 - 3x^2 + x - 2$ in telescoping form.

ii) Evaluate $f(1)$ and $f(-1)$ by Horner's method.

Question 2

(i) Write $f(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ in telescoping form.

(ii) Let c be any real number. How many multiplications are used in evaluating $f(c)$ when $f(x)$ is written in telescoping form.

(iii) How many multiplications are used in evaluating $f(c)$ from the given form of $f(x)$ rather than the telescoping form?

(iv) How many additions/subtractions are used in evaluating $f(c)$ when $f(x)$ is written in telescoping form?

(v) How many additions/subtractions are used in evaluating $f(c)$ from the given form of $f(x)$ rather than the telescoping form?

(vi) Is it clear from the above calculations why Horner's method is used in preference to the standard substitution method?

Question 3

Let $f(x) = 2x^3 + 3x^2 + x - 1$ for $x \in \mathbb{R}$. Prove $f(x) = \mathcal{O}(x^3)$.

Question 4

Let $\log_2$ be denoted simply by log, as is often the case in the literature of computer science. For simplicity, we allow n below only to be positive integers.

(i) In Question 6, Practice Class 1, you proved that $n \le 2^n$, for each $n \in \mathbf{N}$. Using this fact prove that $\log n \le n$, for all integers $n \ge 1$.

(ii) Write down the definitions of $\log n$ being $\mathcal{O}(n)$, i.e. $\log n = \mathcal{O}(n)$.

(iii) Prove, using (i), that $\log n = \mathcal{O}(n)$.

(iv) 1 can be regarded as the function g: $\mathbf{N} \to \mathbf{R}$ with $g(n) = 1$ for all $n \in \mathbf{N}$. Prove that $1 = \mathcal{O}(\log n)$.

(v) Prove that $n = \mathcal{O}(n \log n)$.

(vi) Prove that $n \log n = \mathcal{O}(n^2)$.

(vii) Prove that $2^n < n!$, for all $n \ge 4$.

(viii) Using (vii) prove that $2^n = \mathcal{O}(n!)$.

*(ix) Prove $n^2 = \mathcal{O}(2^n)$.

$$\boxed{\begin{array}{l}\text{Hence we have shown in this question}\\[1em] \qquad 1 = \mathcal{O}(\log n), \quad \log n = \mathcal{O}(n), \quad n = \mathcal{O}(n \log n),\\[1em] \qquad n \log n = \mathcal{O}(n^2), \quad n^2 = \mathcal{O}(2^n), \quad 2^n = \mathcal{O}(n!)\ .\end{array}}$$

Question 5

Let $f(n) = n^3 - 4n^2 - n + 1$ for $n \in \mathbb{N}$. Prove $f(n) = \Theta(n^3)$.

Question 6

Let $f(n) = \dfrac{4n^2 + 2\log_2 n}{n}$ for $n = 1, 2, 3, \cdots$. Prove $f(n) = \Theta(n)$.

Question 7

Define $f : \mathbf{N} \to \mathbf{R}$ by $f(n) = n^3 - 2n^2 + \log n - 3$. Prove $f(n) = \Theta(n^3)$.

PRACTICE CLASS 3
Symbolic Logic and Logical Equivalence

Question 1

Let p be 'Mark is rich' and q be 'Mark is happy'. Write each of the following in symbolic form. (Assume 'poor' and 'not rich' are the same!)

(i) Mark is poor and happy.

(ii) Mark is neither poor nor happy.

(iii) Mark is poor or else he is both rich and happy.

Question 2

In each case decide whether the proposition is True or False.

(i) $(3 + 7 > 10)$ or $(6 < 10)$.

(ii) $\sim (3 + 7 > 10)$.

(iii) $\sim ((2 + 2 = 3) \wedge (2 + 3 = 15))$.

(iv) If $1 + 2 = 4$ then $16 = 35$.

(v) If x is a positive integer and $x^2 \leq 3$ then $x = 1$.

Question 3

(i) Write down a truth table to show $p \to q$ is equivalent to $(\sim q) \to (\sim p)$ <u>and</u> to $(\sim p) \vee q$.

(ii) Use (i) to write expressions using only $\wedge$, $\vee$ and $\sim$ that are equivalent to:
(a) $(p \wedge q) \to r$; (b) $p \to (p \wedge q)$.

Question 4

(i) Write down a truth table to show that the negation of $p \to q$ is equivalent to $p \wedge (\sim q)$.

(ii) To show $p \to q$ is false, we must show p is ... and q is ... .

Question 5

Use de Morgan's Laws to simplify the following:

(i) $\sim (p \wedge \sim q)$ (ii) $\sim (\sim p \vee q)$

(iii) $\sim (\sim p \wedge \sim q)$ (iv) $\sim ((p \wedge q) \vee \sim q)$.

PRACTICE CLASS 4
Propositions and Predicates

Question 1

Use a truth table to show that

$$\sim p \wedge \sim q,\ \sim p \rightarrow r,\ \sim q \rightarrow r,\ \therefore r$$

is a valid argument.

Question 2

For the following valid argument form

$$s \rightarrow \sim p,\quad t \vee s,\quad \sim s \rightarrow u \wedge q,\quad p,\quad t \wedge q \rightarrow r,\quad \therefore r$$

list the valid elementary argument forms or inference rules that are used to derive the conclusion.

Question 3

For the following valid argument form

$$\sim p \vee q \rightarrow r,\quad s \vee \sim q,\quad \sim t,\quad p \rightarrow t,\quad \sim p \wedge r \rightarrow \sim s,\quad \therefore \sim q$$

list the valid elementary argument forms or inference rules that are used to derive the conclusion.

Question 4

Show that the following propositions

1. $P \rightarrow (Q \rightarrow P)$
2. $(\sim Q \rightarrow \sim P) \rightarrow (P \rightarrow Q)$
3. $(\forall x, P(x) \rightarrow Q(x)) \rightarrow ((\forall x, P(x)) \rightarrow (\forall x, Q(x)))$

are all tautologies.

Question 5

Indicate whether the arguments in (i) and (ii) below are valid or invalid. Support your answers by drawing set diagrams.

(i) All people are mice.
All mice are mortal.
$\therefore$ All people are mortal.

(ii) All teachers occasionally make mistakes.

No gods ever make mistake.

∴ No teachers are gods.

Question 6

Reorder the premises in each of the arguments so that the conclusion follows logically.

1. I trust every animal that belongs to me.
2. Dogs gnaw bones.
3. I admit no animals into my study unless they will beg when I tell them to do so.
4. All animals in the yard are mine.
5. I admit every animal that I trust into my study.
6. The only animals that are really willing to beg when told to do so are dogs.

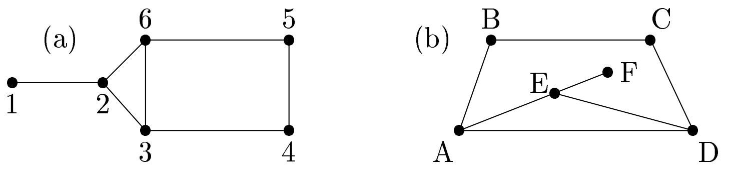
∴ All the animals in the yard gnaw bones.

PRACTICE CLASS 5
Isomorphism and Adjacency Matrix

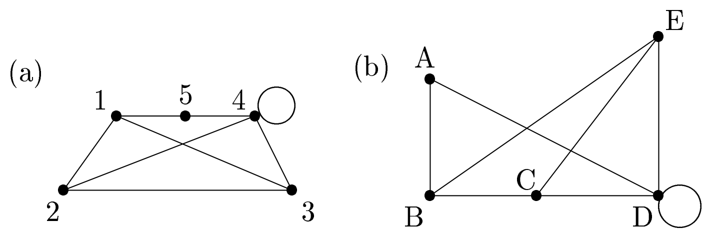
Question 1

For each of the following pairs of graphs, find <u>all</u> the possible isomorphisms from graph (a) onto graph (b).

(i)



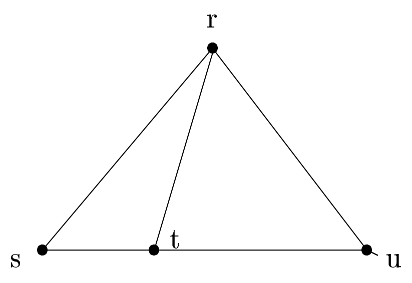
(ii)



Question 2

Another way of describing a graph is to use an $n \times n$ matrix, an *adjacency matrix.* The entry in the ij-th position in the matrix is the number of edges connecting the ith and jth vertices. This method of describing a graph is most useful for storing the information in a computer. For example, the following gives two descriptions of the same graph.

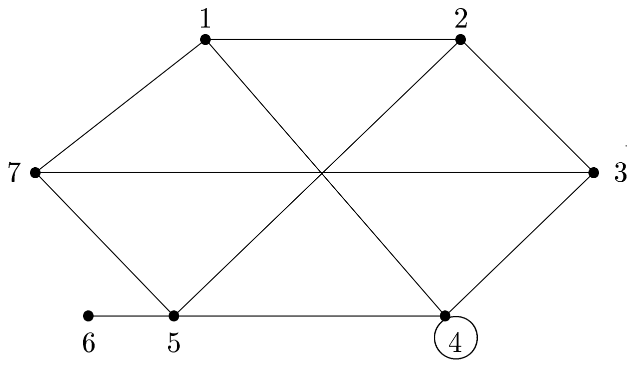
$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$



(i) Notice that the matrix is symmetrical about the leading diagonal. Why?

(ii) Why does the leading diagonal contain only zeros?

(iii) Write the corresponding adjacency matrix for the following graph.



Question 3

Recall that a complete graph K_n has n vertices, and each pair of vertices are joined by an edge. For which values of n is the graph K_n planar? Justify your answer.

Question 4*

Draw three non-isomorphic planar graphs with 4 edges and 5 vertices and prove that they are not isomorphic.

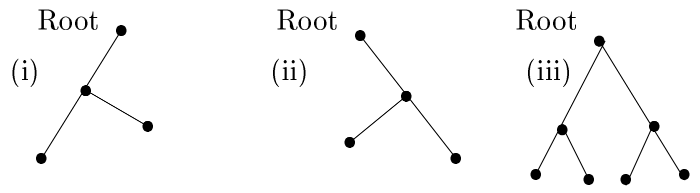
Question 5*

Find all non-isomorphic, non-planar graphs with 6 vertices and containing no loops or multiple edges. Explain why you have all possible graphs.

PRACTICE CLASS 6
Binary Trees for Representation and Sorting

A *Binary Tree* is a special sort of tree, which can be built up from a single vertex, called the *root*, by successfully adding either a *left edge* or a *right edge*, or both.

Examples



In example (i), the root has a left edge, but no right edge, and Example (ii) has a right edge but no left edge. We consider these two trees to be different.

Question 1

The number of different binary trees with n vertices is $\binom{2n}{n}/(n+1) = \frac{(2n)!}{(n+1)!n!}$.

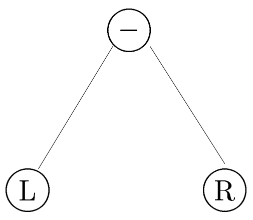
Verify this formula for $n = 2$ and $n = 3$ by drawing all binary trees with 2 or 3 vertices.

Question 2

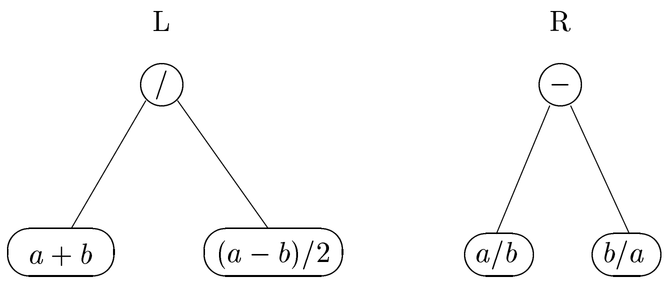
Binary trees can be used to handle *binary operations.* We will look at how they can be used to store algebraic expressions involving the usual operations − addition, multiplication, division and subtraction. Consider, for example the expression

$$((a + b)/((a - b)/2)) - ((a/b) - (b/a))$$

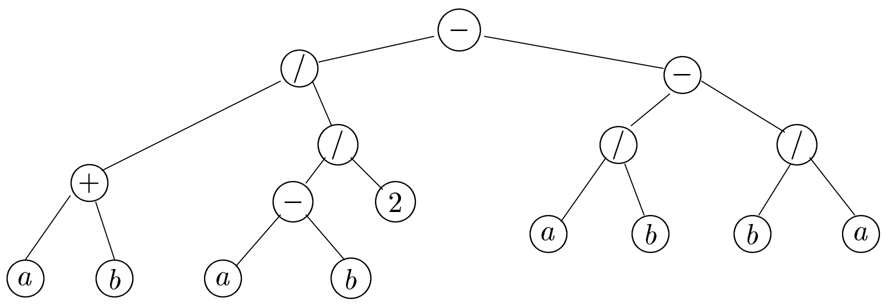
As a binary tree, we can describe this as



where $L = (a+b)/((a-b)/2)$ and $R = (a/b) - (b/a)$. Then we treat L and R as the binary trees; that is,



Continuing this process, we obtain the binary tree

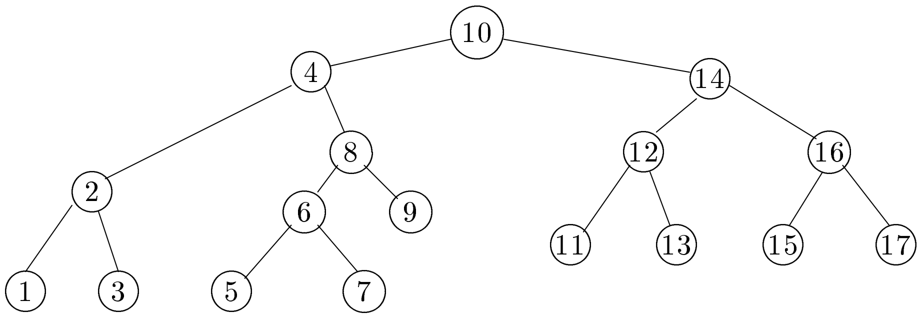


(a) Draw a binary tree to describe each of the following algebraic expressions.

(i) $(((a+b)/(c+d))+a) - (((c+d)/(a+b))+b)$;

(ii) $((a*b)+(b/a))/((a/b)+b)$.

We must also be able to retrieve the algebraic expression described in a binary tree. Take the example before part (a). If we numbered each pronumeral and binary operation in the order they appear in the expression, then they appear in the binary tree in the following order.



Notice that we don't get to the root 10 until we have been to every vertex in the left subtree (which has root 4). We don't get to the root 4 until we have been to every vertex in the left subtree (which has root 2).

When we have reached the root of a tree, we then visit the right subtree.

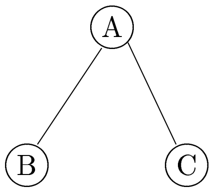
This can be described as follows:

Traversing Binary Trees

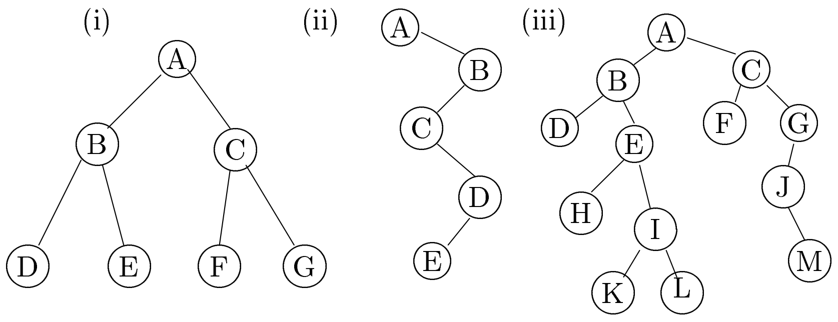
(I) Traverse the left subtree.
(II) Visit the root.
(III) Traverse the right subtree.

There are three commonly used methods for traversing binary trees – 'preorder traversal', 'postorder traversal' and 'inorder traversal'. The one described here is **inorder traversal**. Some calculators (for example, Hewlett-Packard) use postorder traversal.

So for example, the traversal list for the following binary tree is B, A, C



(b) For each of the following binary trees, describe the order in which the vertices are traversed.



Question 3 Sorting using Binary Trees

Suppose we wanted to sort a list of numbers into increasing order. We firstly create a binary tree containing these numbers. Then we use the method given above to create a traversal list.

> **Creating Binary Trees** Insert the first number into the root. We then insert each new number n in turn by:
>
> (A) moving down the left subtree if n is less than the number already at any vertex in the subtree;
>
> (B) moving down the right subtree if n is greater than or equal to the number already at any vertex in the subtree; and
>
> (C) creating a new vertex to hold the number n when the procedure in (A) and (B) would require moving down an 'empty' subtree.

(a) Create a binary tree containing the list 25, 17, 9, 20.

(b) Create a binary tree containing the list 25, 17, 9, 20, 13, 35, 30.

(c) Traverse the trees in (a) and (b).

(i) For each of the 6 possible listings of the numbers 1, 2, 3, create a binary tree to contain that list, noting the number of comparisons required.

(ii) For a list with 3 elements what is the *average* number of comparisons required?

In fact it can be shown that the average number of comparisons using this sorting technique has complexity

$$\Theta(n \log n)$$

(d) Use a binary tree to sort the following lists of numbers, noting the number of comparisons required in each case.

  (i) 1,2,3,4,5,6,7.

  (ii) 2,4,6,1,3,5,7.

  (iii) 4,2,6,1,3,5,7.

(e) Suppose you were given a list containing the first 15 numbers to sort.

  (i) What orderings of this list would require the most number of comparisons?

  (ii) What orderings of this list would require the least number of comparisons?

*(f) Repeat (e) for a list containing the first n numbers.

PRACTICE CLASS 7
Number Bases

Question 1 Use the Division Algorithm to convert the decimal expressions:

a)	115 to base 3	b)	115 to base 5	c)	206 to base 5
d)	12 to base 3	e)	67 to base 2	f)	67 to base 4
g)*	.12 to base 5	h)*	.6 to base 3	i)*	.56 to base 4

Question 2 Convert the following expression <u>in base 8</u> to base 2.

a)	34	b)	21	c)	74
d)	265	e)	752	f)	5724

Question 3 Do the following additions in binary arithmetic.

$$\begin{array}{lr@{\qquad}lr@{\qquad}lr}
\text{a)} & 10110 & \text{b)} & 101 & \text{c)} & 1101101 \\
 & +\underline{11011} & & +\underline{1111} & & +\ \underline{101011}
\end{array}$$

$$\begin{array}{lr@{\qquad}lr@{\qquad}lr}
\text{d)} & 1101 & \text{e)} & 1011 & \text{f)} & 10101 \\
 & 10000 & & 11001 & & 1101 \\
 & 1100 & & +\ \underline{1101} & & 11 \\
 & +\ \underline{10111} & & & & +\ \underline{101}
\end{array}$$

Question 4 Do the following subtractions in binary arithmetic.

$$\begin{array}{lr@{\qquad}lr@{\qquad}lr}
\text{a)} & 10011 & \text{b)} & 10111 & \text{c)} & 1101111 \\
 & -\ \underline{1101} & & -\ \underline{1110} & & -\ \underline{10001}
\end{array}$$

$$\begin{array}{lr@{\qquad}lr@{\qquad}lr}
\text{d)} & 1011100 & \text{e)} & 100111 & \text{f)} & 1100011 \\
 & -\ \underline{111100} & & -\underline{100101} & & -\ \underline{101111}
\end{array}$$

Question 5 Do the following multiplications and divisions.

a)	10111×1111	b)	10100×111
c)	$11011 \div 11$	d)	$1101100 \div 1001$

PRACTICE CLASS 8
Equivalence Relations and Partial Order Relations

Question 1

Let $\mathbb{R}$ be the set of real numbers, we define a relation R on $\mathbb{R}$ by

$$\forall x, y \in \mathbb{R} : \qquad x\,R\,y \quad \text{iff} \quad |x - y| \leq 1 \ .$$

(i) Draw the relation set R explicitly on the plane $\mathbb{R} \times \mathbb{R}$.

(ii) Is the relation reflexive?

(iii) Is the relation symmetric?

(iv) Is the relation an equivalence relation?

(v) Is the relation an partial order relation?

Question 2

Let A be any subset of $\mathbf{N}$ and is composed of some integers ≥ 1. We define a relation $R \subseteq A \times A$ by

$$\forall m, n \in A : \qquad (m, n) \in R \quad \text{iff} \quad m \,|\, n \ . \tag{$*$}$$

Then show

(i) The relation R is a partial order relation.

(ii) Suppose A is given by $A = \{1, 2, 3, 9, 18\}$ and $a \preceq b$ iff $a \,|\, b$, then the relation defined by $\preceq$ is just a special case of (i) and is thus a partial order relation. List the ordered pairs in R and draw the associated Hasse diagram.

Question 3

Let A be the set of all integers greater than or equal to 2, and relation R be defined by

$$\forall m, n \in A : \qquad (m, n) \in R \quad \text{iff} \quad m \,|\, n \ .$$

We furthermore denote R simply by $\preceq$. Show

(i) 2, 3 and 5 are all minimal elements of A.

(ii) 4 is not a minimal element of A.

(iii) A does not have any maximal elements.

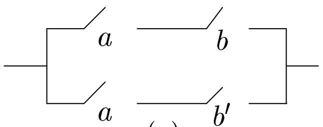
(iv) 2 is not a least element of A.

(v) For any integer $p \geq 2$, p is a prime number if and only if p is a minimal element of A.

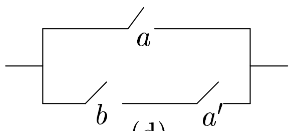
PRACTICE CLASS 9
Switching Circuits and Their Design

Question 1

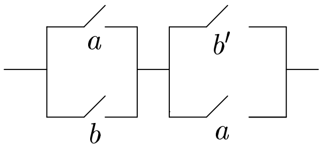
Simplify, where possible, each of the circuits. (You may need to write down the corresponding Boolean expression, simplify, then draw the simplified circuit.)



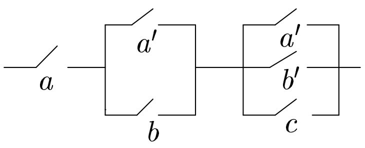
(a)



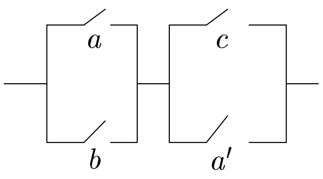
(d)



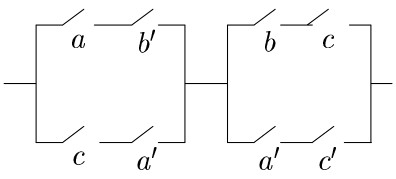
(b)



(e)



(c)



(f)

Question 2

Suppose we have a garage light that we wish to be able to control from either inside the house or the garage. So, the light will have two switches, and it must be possible to turn the light on or off from either switch.

(a) construct a table showing the possible positions (on or off) of the switches, and the corresponding outcomes.

(b) Write down a Boolean expression for the outcome.

(c) Draw a switching circuit for this situation.

PRACTICE CLASS 10
Solutions of Homogeneous Recurrence Relations

Question 1

Consider the following recurrence relations (difference equations) and initial conditions:

(a) $S(k+2) = 5S(k+1) - 4S(k)$, for $k \geq 0$, $S(0) = 2$, and $S(1) = 5$.

(b) $F(n+1) = 5F(n) + 5^n$, for $n \geq 0$, and $F(0) = 3$.

(c) $R(n+3) - 6R(n+2) + 11R(n+1) - 6R(n) = 0$, for $n \geq 0$, $R(0) = 3$, $R(1) = 6$, and $R(2) = 14$.

(d) $F_{k+2} = F_{k+1} + F_k$, for $k \geq 0$, $F_0 = 1$ and $F_1 = 1$.

(i) Write down the order of each of the above recurrence relations.

(ii) State which of the recurrence relations are homogeneous.

(iii) Write down the characteristic equation of each of the homogeneous relations.

(iv) Solve each of the homogeneous recurrence relations.

Question 2

Solve the recurrence relations

(a) $f(n+2) + 6f(n+1) + 9f(n) = 0, \text{ for n} \geq 2, \text{ f(0)} = 1, \text{ f(1)} = 0.$

(b) $A_{n+3} - 3A_{n+2} + 4A_n = 0.$

PRACTICE CLASS 11

Solutions of Recurrence Relations and Practical Problems

Question 1

Solve the following recurrence relations:

(a) $S(k+2) - 6S(k+1) + 9S(k) = 2, \quad$ for $k \geq 0$

(b) $T(n+1) - 5T(n) = 7^n, \quad$ for $n \geq 0 \quad T(0) = \frac{9}{2}$.

(c) $P(n+2) - 4P(n+1) + 4P(n) = n^2, \quad$ for $n \geq 0$.

In "real life" the problems you meet are expressed in words and you have to convert the words to symbols. Only then can you manipulate those symbols to get a solution. Questions 2 and 3 below are expressed in words. Do TRY to convert the problems into symbols WITHOUT seeking assistance.

Question 2

A particle is moving in the horizontal direction. The distance it travels in each second is equal to twice the distance it travelled in the previous one. Let a_r denote the position of the particle in the r^{th} second. Determine a_r given that $a_0 = 3$ and $a_1 = 10$.

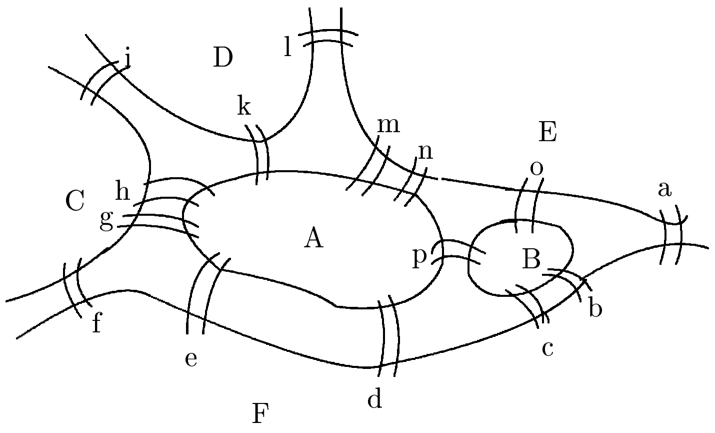
Question 3

You open a savings account which pays an annual interest rate of 10%. You deposit \$100 when you open the account and you double your deposit each year. Let $B(n)$ be the balance after n years. Find a "closed form expression" for $B(n)$.

SUPPLEMENTARY PRACTICE CLASS - REVISION

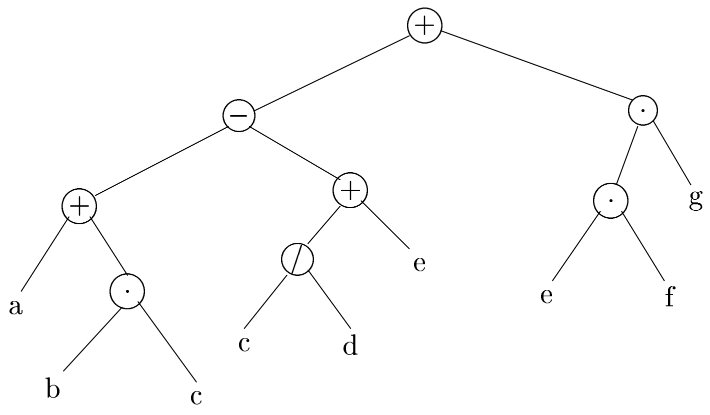
Question 1

Consider the following diagram of a river with two islands and 15 bridges. Is it possible to design a walk so that you traverse each bridge exactly once? Justify your answer.



Question 2

What is the algebraic expression held in the following binary tree?



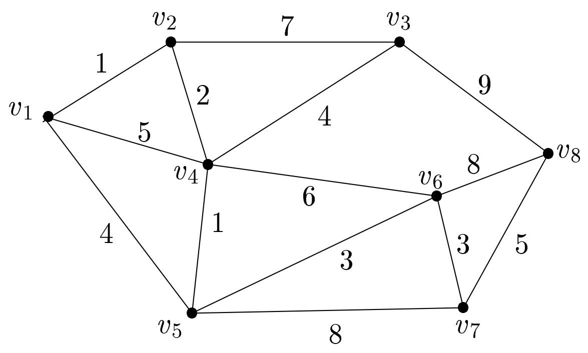
Question 3

Solve the following recurrence relation.

$$6f(n+2) - 5f(n+1) + f(n) = 0, f(0) = 1, f(1) = 0.$$

Question 4

Use Kruksal's algorithm to find a minimal spanning tree for the following weighted graph.



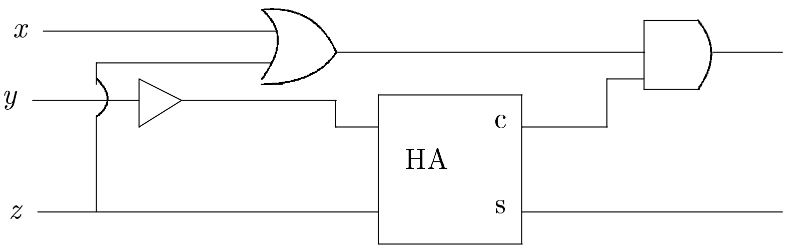
Question 5

Use the characteristic equation method to solve

$$S(n+2) - 2S(n+1) + S(n) = 4, \; S(0) = 2, \; S(1) = 2.$$

Question 6

What are the outputs of the following logic circuit? Show each intermediate value.



Question 7

Simplify then draw a switching circuit for the Boolean expression $(x + y')(x + yz') + xyz$.

Question 8

Let $f : \mathbf{N} \to \mathbf{R}$ be given by $f(n) = 2n^3 + 3n^2 \log n + 2n - 1$. Prove that $f(n) = \Theta(n^3)$.