1. What's wrong with the following proofs by induction?

   (a) All binary strings are identical. The proof is by induction on the size of the string. For $n=0$ all binary strings are empty and therefore identical. Let $X = b_n b_{n-1} \ldots b_1 b_0$ be an arbitrary binary string of length $n+1$. Let $Y = b_n b_{n-1} \ldots b_1$ and $Z = b_{n-1} \ldots b_1 b_0$. Since both Y and Z are strings of length less than $n+1$, by induction they are identical. Since the two strings overlap, X must also be identical to each of them.
   ANSWER
   This is not a subtly flawed proof. It is a terribly flawed proof. The problem is with the induction step. It is never true that the pasting together of two identical strings is identical to the strings. Furthermore, the $n = 1$ case fails for another reason: A string of length 1 is not the pasting together of two strings of length 0.

   (b) Any amount of change greater than or equal to twenty can be gotten with a combination of five cent and seven cent coins. The proof is by induction on the amount of change. For twenty cents use four five-cent coins. Let $n > 20$ be the amount of change. Assume that $n = 7x + 5y$ for some non-negative integers x and y. For any $n > 20$, either $x > 1$, or $y > 3$. If $x > 1$, then since $3(5) - 2(7) = 1$, $n+1 = 5(y+3)+7(x-2)$. If $y > 3$, then since $3(7) - 4(5) = 1$, $n+1 = 7(x+3)+5(y-4)$. In either case, we showed that $n+1 = 7u + 5v$ where u and v are non-negative integers.
   ANSWER
   For any $n > 20$, either $x > 1$ or $y > 3$ is false. For example, if $n = 21$, then $21 = 0 \cdot 5 + 3 \cdot 7$.

2. Prove by induction that:

   (a) The nth Fibonacci number, F_n equals

   $$G_n = \frac{1}{\sqrt{5}} \left[(\frac{1 + \sqrt{5}}{2})^n - (\frac{1 + \sqrt{5}}{2})^n \right]$$

   where $F_0 = 0$ and $F_1 = 1$.
   ANSWER
   Base Case: These hold since

   $$G_0 = \frac{1}{\sqrt{5}} \left[(\frac{1 + \sqrt{5}}{2})^0 - (\frac{1 + \sqrt{5}}{2})^0 \right] = \frac{1}{\sqrt{5}}(1 - 1) = 0$$

   and

   $$G_1 = \frac{1}{\sqrt{5}} \left[(\frac{1 + \sqrt{5}}{2})^1 - (\frac{1 + \sqrt{5}}{2})^1 \right] = \frac{1}{\sqrt{5}} \left[\frac{1 + \sqrt{5}}{2} - \frac{1 - \sqrt{5}}{2} \right] = 1$$

   Induction Step: Suppose that $G_n = F_n$ and $G_{n-1} = F_{n-1}$ We would like to show

   $$G_{n+1} = F_{n+1}$$

   Let $a = (\frac{1}{2} + \frac{\sqrt{5}}{2})$
   Let $b = (\frac{1}{2} - \frac{\sqrt{5}}{2})$
   We will use the fact that a, b are roots of $r^2 - r - 1 = 0$ and hence

   $$a^2 = a + 1 \qquad b^2 = b + 1$$

1

Starting with G_{n+1},

$$
\begin{aligned}
G_{n+1} = \frac{1}{\sqrt{5}}(a^{n+1} - b^{n+1}) &= \frac{1}{\sqrt{5}}(a^2 a^{n-1} - b^2 b^{n-1}) \\
&= \frac{1}{\sqrt{5}}((a+1)a^{n-1} - (b+1)b^{n-1}) \\
&= \frac{1}{\sqrt{5}}(a^n - b^n) + \frac{1}{\sqrt{5}}(a^{n-1} - b^{n-1})) \\
&= F_n + F_{n-1} \\
&= F_{n+1}
\end{aligned}
$$

So the induction step is proven. QED.

(b) The sum of the geometric series $1 + a + a^2 + \ldots + a^n$ equals $(1-a^{n+1})/(1-a)$, where a does not equal one.

ANSWER

Base Case: $n = 0$

$$
\frac{(1 - a^{0+1})}{1 - a} = 1
$$

Inductive Step: Suppose

$$
1 + a + a^2 + \ldots + a^n = \frac{1 - a^{n+1}}{1 - a}
$$

Then

$$
\begin{aligned}
1 + a + a^2 + \ldots + a^n + a^{n+1} &= \frac{1 - a^{n+1}}{1 - a} + a^{n+1} \\
&= \frac{1 - a^{n+1} + a^{n+1} - a^{n+2}}{1 - a} \\
&= \frac{1 - a^{n+2}}{1 - a}
\end{aligned}
$$

And the inductive step is proven. Q.E.D.

(c) 21 divides $4^{n+1} + 5^{\,2n-1}$

ANSWER

Base Case: If $n = 1$, then

$$
4^2 + 5^1 = 21,
$$

which is divisible by 21. Inductive step: Assume that for some n

$$
4^{n+1} + 5^{2n-1}
$$

is divisible by 21. Then

$$
\begin{aligned}
4^{n+2} + 5^{2n+1} &= 4 \cdot 4^{n+1} + 5^2 \cdot 5^{2n-1} \\
&= 4 \cdot 4^{n+1} + 4 \cdot 5^{2n-1} + 21 \cdot 5^{2n-1} \\
&= 4(\cdot 4^{n+1} + \cdot 5^{2n-1}) + 21 \cdot 5^{2n-1}
\end{aligned}
$$

The first term is divisible by 21 by the induction hypothesis, and the second term is manifestly divisible by 21. Hence the induction step holds. Q.E.D.

2

(d) The number of leaves in a complete binary tree is one more than the number of internal nodes. (Hint: Split the tree up into two smaller trees).

ANSWER
We induct on the depth of the tree.
Base Case: For a tree of depth 1, there is one internal node and two leaves.
Inductive step: Suppose for all complete binary trees T of depth at most n, that

$$L(T) = I(T) + 1$$

Where $L(t)$ is the number of leaves of T and $I(T)$ is the number of internal nodes of T.
Let B_{n+1} be a complete binary tree of depth $n + 1$. We would like to verify

$$L(B_{n+1}) \quad ? =? \quad I(B_{n+1}) + 1$$

This will follow from the following relations between B_{n+1} and its left and right subtrees B_n, B'_n.

- $I(B_{n+1}) = I(B_n) + I(B'_n) + 1$ (+1 for the new root)
- $L(B_{n+1}) = L(B_n) + L(B'_n)$

So

$$
\begin{aligned}
L(B_{n+1}) &= L(B_n) + L(B'_n) \\
&= (I(B_n) + 1) + (I(B'_n) + 1) \quad \text{by induction} \\
&= (I(B_n) + I(B'_n) + 1) \\
&= I(B_{n+1}) + 1
\end{aligned}
$$

And the induction step is proven. Q.E.D.

(e) A graph's edges can be covered by n edge-disjoint paths, but not *n-1*, if and only if the graph has n pairs of odd-degree vertices. (Euler discussed the case for *n = 1*).

ANSWER
There are two directions to the proof.
$\Leftarrow$
Thanks to Sam Klein for the argument in the first direction.
Suppose that a graph G has n pairs of odd degree vertices, $a_1, b_1, a_2, b_2, ..., a_n, b_n$. Construct a graph G' that is G with the n edges $\{a_1, b_1\}, \cdots \{a_n, b_n\}$ thrown in. In G' all the vertices are of even degree, so there is an Euler circuit, i.e. a path which traverses every edge of G' without going over any edge twice. Removing the edges we added to G, we chop the loop into n edge-disjoint paths that cover G.

To prove the rest, we will use the following fact: *if k edge-disjoint paths cover a graph X, then X has at most k pairs of odd-degree vertices.*

This is because an odd degree vertex must lie at the beginning or the end of some path, or else there will be an outgoing edge for each incoming edge at the vertex and so it will necessarily be even degree. So it follows that there cannot be $n - 1$ edge-disjoint paths if there are n pairs of odd-degree vertices.

$\Rightarrow$
Conversely, if there are n edge disjoint paths but not $n - 1$ (or any smaller number). We know then that there are at most n pairs of odd vertices. If there were fewer than n pairs of odd vertices then by the other direction of the proof, we would have fewer than n edge-disjoint paths that cover the graph, which would contradict our assumption. Therefore there must be n pairs of odd-degree vertices.

3. Solve the following recurrence equations using the techniques for linear recurrence relations with constant coefficients. State whether or not each recurrence is homogeneous.

(a) $a_n = 6a_{n-1} - 8a_{n-2}$, and $a_0 = 4$, $a_1 = 10$.

      ANSWER $a_n = 6a_{n-1} - 8a_{n-2}$, and $a_0 = 4, a_1 = 10$
      This is a homogeneous recurrence equation.

$$
\begin{aligned}
\frac{r^k}{r^{k-2}} &= \frac{6r^{k-1} - 8k^{k-2}}{r^{k-2}} \\
r^2 &= 6r - 8 \\
r^2 - 6r + 8 &= 0 \\
(r-2)(r-4) &= 0
\end{aligned}
$$

      The roots are 2, 4.

$$
\begin{aligned}
a_n &= \alpha_1 2^n + \alpha_2 4^n \\
a_0 &= \alpha_1 2^0 + \alpha_2 4^0 = \alpha_1 + \alpha_2 = 4 \\
a_1 &= \alpha_1 2^1 + \alpha_2 4^1 = 2\alpha_1 + 4\alpha_2 = 10
\end{aligned}
$$

      $\alpha_1 = 3$ and $\alpha_2 = 1$. The solution is $a_n = 3(2^n) + 1(4^n)$.

(b) $a_n = a_{n-1} + 2a_{n-2}$, and $a_0 = 0$, $a_1 = 1$.

      ANSWER This is a homogeneous recurrence equation.
      The solution is $a_n = \frac{1}{3}(2^n) - \frac{1}{3}(-1^n)$.

(c) $a_n = 7a_{n-1} - 10a_{n-2} + 3^n$, and $a_0 = 0$, $a_1 = 1$.

      ANSWER This is a nonhomogenous equation. First, determine the roots of the associated homogeneous equation:

$$
\begin{aligned}
\frac{r^k}{r^{k-2}} &= \frac{7r^{k-1} - 10k^{k-2}}{r^{k-2}} \\
r^2 &= 7r - 10 \\
r^2 - 7 + 10 &= 0 \\
(r-5)(r-2) &= 0
\end{aligned}
$$

The roots are 5 and 2. Next, find the particular solution, setting $p3^n = a_n$:

$$
\begin{aligned}
p3^n &= 7p3^{n-1} - 10p3^{n-2} + 3^n \\
\frac{p3^n}{3^n} &= \frac{\frac{7}{3}p3^n - \frac{10}{9}p3^n + 3^n}{3^n} \\
p &= \frac{7}{3}p - \frac{10}{9}p + 1 \\
\frac{9}{9}p - \frac{21}{9}p + \frac{10}{9}p &= 1 \\
\frac{2}{9}p &= -1 \\
p &= -\frac{9}{2}
\end{aligned}
$$

$$
\begin{aligned}
a_n &= \alpha_1 2^n + \alpha_2 4^n \\
a_0 = 0 &= \alpha_1 2^0 + \alpha_2 4^0 = \alpha_1 + \alpha_2 - \frac{9}{2} \\
a_1 = 1 &= \alpha_1 5^1 + \alpha_2 2^1 + \frac{9}{2}(3^1) = 5\alpha_1 + 2\alpha_2 - \frac{27}{2}
\end{aligned}
$$

Solving these equations, $\alpha_1 = \frac{11}{6}$, and $\alpha_2 = \frac{16}{6}$. The final solution is

$$
a_n = \frac{11(5^n) + 16(2^n) - 27(3^n)}{6}.
$$

(d) $a_n = 3 - 6a_{n-1} - 9a_{n-2}$, and $a_0 = 0$, $a_1 = 1$.

ANSWER This is not a homogeneous recurrence equation.
The solution is $a_n = -\frac{3}{16}(-3^n) - \frac{1}{12}n(-3^n) + \frac{3}{16}$.

4. A particular graph-matching algorithm on n nodes, works by doing n^2 steps, and then solving a new matching problem on a graph with one vertex less.

(a) Show that the number of steps it takes to run the algorithm on a graph with n nodes is equal to the sum of the first n perfect squares.

ANSWER Taking $T_0 = 0$ as the base case, we can use substitution to show that this algorithm is equal to the sum of the squares from one to n.

$$
\begin{aligned}
T_n &= T_{n-1} + n^2 \\
&= (T_n - 1 + (n-1)^2) + n^2 \\
&= (T_n - 2 + (n-2)^2) + (n-1)^2 + n^2 \\
&= T_n - r + (n-r)^2 + (n-(r+1))^2 + (n-(r+2))^2) \ldots (n-1)^2 + n^2 \\
&= 0 + 0 + 1^2 + 2^2 + 3^2 + \ldots + (n-1)^2 + n^2 \\
&= 1^2 + 2^2 + 3^2 + \ldots + (n-1)^2 + n^2
\end{aligned}
$$

(b) Derive the formula for the sum of the first n perfect squares by constructing an appropriate linear non-homogeneous recurrence equation and solving it.

5

ANSWER The general equation for the recurrence equation $a_n = a_{n-1} + n^2$ can be expressed as $a_n = \alpha_1(1^n) + n(p_2 n^2 + p_1 n + p_0)(1^n)$. First solve to find $a_n^{(p)}$:

$$a_n = n(p_2 n^2 + p_1 n + p_0) = p_2 n^3 + p_1 n^2 + p_0 n \equiv (p_2(n-1)^3 + p_1(n-1)^2 + p_0(n-1)) + n^2$$

$$p_2 n^3 + p_1 n^2 + p_0 n \equiv (p_2 n^3 - p_2 3n^2 + p_2 3n - p_2) + (p_1 n^2 - p_1 2n + p_1) + (p_0 n - p_0) + n^2$$

$$
\begin{aligned}
p_2 3n^2 - p_2 3n + p_2 + p_1 2n - p_1 + p_0 - n^2 &= 0 \\
(3p_2 - 1)n^2 + (2p_1 - 3p_2)n + (p_2 - p_1 + p_0) &= 0 \\
(-1)n^2 + (0)n + (p_2 - p_1 + p_0) &= 0
\end{aligned}
$$

Solving the series of equations, $p_2 = \frac{1}{3}$, $p_1 = \frac{1}{2}$, and $p_0 = \frac{1}{6}$. So, $a_n^{(p)} =$

$$n\left(\frac{n^2}{3} + \frac{n}{2} + \frac{1}{6}\right) = n\left(\frac{2n^2 + 3n + 1}{6}\right) = \frac{n(2n+1)(n+1)}{6}.$$

Solving for $a_0 = 0$, $a_n^{(h)} = 0$. The closed form is then $\frac{n(2n+1)(n+1)}{6}$.

(c) Show that the time complexity of this algorithm is $\Theta(n^3)$.

ANSWER We can multiply the the solution from 4b to equal $\frac{1}{6}(2n^3 + 3n^2 + n)$, and show that this function is both $O(n^3)$ and $\Omega(n^3)$ as follows:

$$
\begin{aligned}
\frac{1}{6}(2n^3 + 3n^2 + n) &\leq n^3 \\
\frac{1}{6}(2n^3 + 3n^2 + n) &\geq \frac{1}{6}n^3
\end{aligned}
$$

with $n > 1$. Therefore, this function is $\Theta(n^3)$.

5. Write a recurrence relation to compute the number of binary strings with n digits that do not have two consecutive 1's. Solve the recurrence, and determine what percentage of 8-bit binary strings do not contain two consecutive 1's.

ANSWER In the following table, T_n is the number of binary numbers which do not have consecutive ones with n binary digits.

n	T_n
1	2
2	3
3	5
4	8
5	13

This can be expressed in the recurrence equation $T_n = T_{n-1} + T_{n-2}$, with $T_1 = 2$ and $T_2 = 3$. This is just the Fibonacci recurrence relation with initial conditions shifted over by 2, since $F_3 = 2$, and $F_4 = 3$ That is,

$$T_n = F_{n+2} = \frac{1}{\sqrt{5}}\left[(\frac{1+\sqrt{5}}{2})^{n+2} - (\frac{1+\sqrt{5}}{2})^{n+2}\right]$$

The percentage binary numbers which do not have consecutive ones for $n = 8$ is $\frac{55}{256}$, or 21.5%. Note, we don't need a brute force method to figure out the recurrence relation. It can be proven as follows: Either the string begins with a 0, and there are T_{n-1} cases were this happens, or the string begins with a 1, so necessarilly, the first two bits are 10, and there are T_{n-2} cases were this happens.

6. Strassen's algorithm shows how to multiply two n by n matrices by multiplying 7 pairs of $n/2$ by $n/2$ matrices, and then doing n^2 operations to combine them. Write the recurrence equation for this algorithm, and calculate the complexity of Strassen's algorithm, by solving the recurrence by repeated substitution.

   ANSWER Strassen's algorithm can be expressed in the recurrence equation $T_n = 7T_{\frac{n}{2}} + n^2$. Assuming $T_1 = 1$, we can solve this algorithm iteratively as follows:

$$
\begin{aligned}
T_1 &= 1 \\
T_2 &= 7 + 2^2 \\
T_4 &= 7(7 + 2^2)4^2 = 7^2 + 7 \cdot 2^2 + 2^4 \\
T_8 &= 7(7^2 + 7 \cdot 2^2 + 2^4) + 8^2 = 7^3 + 7^2 2^2 + 7 \cdot 2^4 + 2^6 \\
T_{2^k} &= 7^k + 7^{k-1} \cdot 2^2 + 7^{k-2} 2^4 + \cdots + 7 \cdot 2^{2k-2} + 2^{2k}
\end{aligned}
$$

   This is a geometric series with a ratio of $\frac{2^2}{7}$. Its sum is

$$
\frac{2^{2k} \cdot 2^2 \cdot 7^{-1} - 7^k}{2^2 \cdot 7^{-1} - 1} = \frac{1}{3}(7^{k+1} - 2^{2k+2})
$$

   Because powers of seven beat powers of four, this is $\Theta(7^{k+1}) = \Theta(7^k)$. And since

$$
7^k = 2^{\log_2 7 \cdot k} = (2^k)^{(\log_2 7)} = n^{\log_2 7}
$$

   So the algorithm is $\Theta(n^{\log_2 7})$

7. Write and solve the recurrence equations for the time complexity of the following recursive algorithms. Explain why your equations are correct.

   (a) To search for a value in a sorted list, compare it to the middle value, and search the right half of the list if it is larger, and the left half if it is smaller.

   ANSWER The equation for this algorithm is $T_n = T_{\frac{n}{2}} + 1$. It is $O(\log n)$

   (b) The maximum of a list of numbers is the larger of the maximum of the first half and the maximum of the second half.

   ANSWER The equation for this algorithm is $T_n = 2T_{\frac{n}{2}} + n$. It is $O(n \log n)$.

   (c) To sort a list of numbers, divide the list into four equal parts. Sort each part. Merge these sorted four lists into two sorted lists, and then merge the two into one.

   ANSWER The equation for this algorithm is $T_n = 4T_{\frac{n}{4}} + 2n$. It is $O(n \log n)$.

8. Solving the following recurrence by a change of variable: $a_n = 2a\sqrt{n} + \lg n$, $a_1 = 0$. (Solve by setting $m = \lg n$). You should solve this only when n is 2 to the power of 2^k.

ANSWER

Let $m = \log n \quad \Rightarrow \quad n = 2^m$. Define $b_m = a_n = a_{2^m}$. Then,

$$b_m = a_n = 2a_{\sqrt{n}} + \log n$$
$$= 2a_{2^{\frac{m}{2}}} + m$$
$$= 2b_{\frac{m}{2}} + m$$

so

$$b_1 = C$$
$$b_2 = 2C + 1 = 2(C + 1)$$
$$b_4 = 2 \cdot 2(C + 1) + 2^2 = 2^2(C + 2)$$
$$b_8 = 2 \cdot 2^2(C + 2) + 2^3 = 2^3(C + 3)$$
$$\vdots$$
$$b_{2^k} = 2^k(C + k)$$

So

$$a_{2^{2^k}} = 2^k(C + k)$$

or

$$a_n = \log n(C + \log \log n)$$

9. Parenthesized Expressions

(a) A sequence of $n+1$ matrices $A_1 A_2 \ldots A_{n+1}$ can be multiplied together in many different ways dependent on the way n pairs of parentheses are inserted. For example for $n+1 = 3$, there are two ways to insert the parentheses: $((A_1 A_2)A_3)$ and $(A_1 (A_2 A_3))$. Write a recurrence equation for the number of ways to insert k pairs of parenthesis. Do not solve it. (Hint: Concentrate on where the *last* multiplication occurs).

ANSWER This algorithm can be expressed as

$$T_n = T_1 T_{n-1} + T_2 T_{n-2} + T_3 T_{n-3} + \ldots + T_{n-1} T_1.$$

(b) Write a list of the different ways to parenthesize a sequence of $n+1$ matrices for $n+1=2,3,4$.

ANSWER The following table shows the possible combinations for $n + 1 = 2, 3, 4$.

$n + 1$	possilibities
2	$(A_1 \cdot A_2)$
3	$(A_1 \cdot (A_2 \cdot A_3))$
	$((A_1 \cdot A_2) \cdot A_3)$
4	$((A_1 \cdot (A_2 \cdot A_3)) \cdot A_4)$
	$(((A_1 \cdot A_2) \cdot A_3) \cdot A_4)$
	$(A_1 \cdot (A_2 \cdot (A_3 \cdot A_4)))$
	$(A_1 \cdot ((A_2 \cdot A_3) \cdot A_4))$
	$((A_1 \cdot A_2) \cdot (A_3 \cdot A_4))$

(c) A balanced arrangement of parenthesis is defined inductively as follows:

The empty string is a balanced arrangement of parentheses. If x is balanced arrangement of parentheses then so is (x). If u and v are each a balanced arrangement of parentheses, then so is uv.

Write a list of strings that represent a balanced arrangement of n parentheses for $n=1,2,3$.

ANSWER The following table shows the possible combinations for $n = 1, 2, 3$.

n	possilibities
1	()
2	(())
	()()
3	(())()
	()()()
	((()))
	(()())
	()(())

(d) Describe a 1-1 correspondence between the strings that are balanced arrangements of n pairs of parentheses, and the number of ways to multiply a sequence of $n+1$ matrices.

ANSWER It's easier to see this, if we map both things to a third set of strings. For the pairs of parentheses replace the left parentheses with α and the right parentheses with β. For example:

n	possilibities	transformed form
1	()	$\alpha\beta$
2	(())	$\alpha\alpha\beta\beta$
	()()	$\alpha\beta\alpha\beta$
3	(())()	$\alpha\alpha\beta\beta\alpha\beta$
	()()()	$\alpha\beta\alpha\beta\alpha\beta$
	((()))	$\alpha\alpha\alpha\beta\beta\beta$
	(()())	$\alpha\alpha\beta\alpha\beta\alpha$
	()(())	$\alpha\beta\alpha\alpha\beta\beta$

For the ways to multiply $n + 1$ matrices, remove the matrices and the left parentheses from the expression, and replace any multiplication by α and any right parenthesis by β. For example:

$n + 1$	possilibities	transformed form
2	$(A_1 \cdot A_2)$	$\alpha\beta$
3	$(A_1 \cdot (A_2 \cdot A_3))$	$\alpha\alpha\beta\beta$
	$((A_1 \cdot A_2) \cdot A_3)$	$\alpha\beta\alpha\beta$
4	$((A_1 \cdot (A_2 \cdot A_3)) \cdot A_4)$	$\alpha\alpha\beta\beta\alpha\beta$
	$(((A_1 \cdot A_2) \cdot A_3) \cdot A_4)$	$\alpha\beta\alpha\beta\alpha\beta$
	$(A_1 \cdot (A_2 \cdot (A_3 \cdot A_4)))$	$\alpha\alpha\alpha\beta\beta\beta$
	$(A_1 \cdot ((A_2 \cdot A_3) \cdot A_4))$	$\alpha\alpha\beta\alpha\beta\beta$
	$((A_1 \cdot A_2) \cdot (A_3 \cdot A_4))$	$\alpha\beta\alpha\alpha\beta\beta$

10. Prove that any $O(|E|)$ time algorithm on a planar graph is also $O(|V|)$. (Hint: Use the fact that every face has at least three vertices and edges, and a counting argument, to calculate a relationship between the number of faces and the number of edges. Then use Euler's Theorem to derive a linear relationship between the number of edges and the number of vertices.)

ANSWER (The devil is in the details... Note that the statement is false without the condition that the graph is simple, i.e. that there is at most one edge connecting any two vertices. Otherwise we could add an arbitrary number of edges between two vertices. and so make the number of edges much much larger than the number or vertices)

The statement follows from the following claim: from page 505 of Rosen

On any simple connected planar graph with e edges and v vertices, $e \leq 3v - 6$.

Therefore if the algorithm satisfies $T_{Graph} \leq Ce$ for large enough e, it satisfies

$$T_{Graph} \leq C(3v - 6) < 3Cv$$

Hence the algorithm is $O(v)$

11. The following recurrence cannot be solved using the master theorem. Explain why. Solve it directly by substitution, and calculate its order of growth.

$$T(n) = 4T(n/2) + (nlogn)^2$$

and

$$T(1) = 1.$$

ANSWER It fails to satisfy the hypotheses of the master theorem.. In this case $a = 4$, $b = 2$ and $\log_b^a = 2$
The function $n^2 \log n$ is $\Omega(n^{\log_b^a}) = \Omega(n^2)$, however it is not polynomially greater than n^2 so the master theorem does not apply. Let's solve this for $n = 2^k$ by repeated substitution. The recurrence relation becomes

$$T(2^k) = 4T(2^{k-1}) + 4^k \cdot k^2$$

Assuming $T(2^0) = 1$ Then,

$$
\begin{aligned}
T(2^1) &= 4 + 4 \\
T(2^2) &= (4^2 + 4^2) + 4^2 \cdot 2^2 \\
T(2^3) &= (4^3 + 4^2 + 4^3 \cdot 2^2) + 4^3 \cdot 3^2 \\
T(2^4) &= (4^4 + 4^4 + 4^4 \cdot 2^2 + 4^4 \cdot 3^2) + 4^4 \cdot 4^2 \\
&\vdots \\
T(2^k) &= 4^k(1 + 1 + 2^2 + 3^2 + \cdots + k^2) \\
&= 4^k(1 + \frac{k(k+1)(2k+1)}{6})
\end{aligned}
$$

So $T(2^k)$ is $\Theta(k^3 4^k)$, or $\Theta((\log n)^3 n^2)$