Question 1 (Program Runtimes, 20 points). Consider the following programs:

Alg1(n):
For i = 1 to n
    j = 1
    while i+j < n
        j = j+1
Alg2(n):
For i = 1 to n
    j = 1
    while i*j < n
        j = j+1

For each of these algorithms, compute the asymptotic runtime in the form \( \Theta(-) \).

Solution.  
(a) We count the number of primitive instructions performed when we run these programs.

The outer for loop runs \( n \) times. The instruction \( j = 1 \) is thus performed \( n \) times. For each iteration of the for loop, the inner while loop runs \( n - i - 1 \) times. The instruction \( j = j + 1 \) is thus performed \( n - i - 1 \) times (except the last iteration, where the inner loop is not executed at all) for each iteration of the outer loop. Since, \( i \) ranges from 1 to \( n \), the total number of times it is performed is thus (last iteration is not counted)

\[
\sum_{i=1}^{n-1} n - i - 1 = \frac{(n-2)(n-1)}{2} = \frac{n^2 - 3n + 2}{2}
\]

Adding the \( n \), \( j = 1 \) instructions performed in the outer loop, we get a total of \( \frac{n^2 - n + 2}{2} \) instructions.

Thus, the asymptotic runtime is \( \Theta(n^2) \).

(b) Similar to Part (a), the outer loop runs for \( n \) times. The inner loop however, runs for \( \lfloor n/i \rfloor \) many times. The total number of times the instruction \( j = 1 \) is performed is

\[
\sum_{i=1}^{n} \lfloor n/i \rfloor \leq \sum_{i=1}^{n} n/i \approx n \log(n)
\]

Adding the \( n \), \( j = 1 \) instructions performed in the outer loop, we get a total of \( n \log(n) + n \) instructions.

Thus, the asymptotic runtime is \( \Theta(n \log(n)) \).
Question 2 (Big-O Computations, 20 points). For each of the following functions, determine whether or not the expression in question is $\Theta(n^c)$ for some constant $c$, and if so determine the value of such a $c$. Remember to justify your answer.

- $a(n) = \frac{n^2}{7} + 21 + \log(n)$
- $b(n) = n + (n - 1) + (n - 2) + \ldots + 1$
- $c(n) = 3^{\lceil \log_2(n) \rceil}$
- $d(n) = \lfloor \log_2(n) \rfloor$
- $e(n) = 2^n$

Solution. 

(a) $a(n) = \frac{n^2}{7} + 21 + \log(n)$. Note that $\frac{n^2}{7} = \Theta(n^2)$ and that $21$ and $\log(n)$ are both $o(n^2)$. Thus, $a(n) = \Theta(n^2)$.

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

$$b(n) = \frac{n(n + 1)}{2} \leq n^2 \quad \forall n \geq 1$$

Thus $b(n) = O(n^2)$.

Also,

$$2b(n) = n(n + 1) \geq n^2 \quad \forall n \geq 1$$

Thus $n^2 = O(b(n))$.

Hence $b(n) = \Theta(n^2)$.

(c) $c(n) = 3^{\lceil \log_2(n) \rceil}$

$$c(n) \geq 3^{\log_2(n)}$$

$$= 3^{\log_3(n)/\log_3 2}$$

$$= n^{1/\log_3 2}$$

$$= n^{\log_2 3}$$

Thus $n^{\log_2 3} = O(c(n))$.

Also,

$$c(n) \leq 3^{\log_2(n)+1}$$

$$= 3 \times 3^{\log_3(n)/\log_3 2}$$

$$= 3 \times n^{1/\log_3 2}$$

$$= 3n^{\log_2 3}$$

Thus $c(n) = O(n^{\log_2 3})$.

Hence $c(n) = \Theta(n^{\log_2 3})$.
(d) \( d(n) = \lfloor \log_2(n) \rfloor! \)

\[
d(n) \geq (\log_2(n) - 1)! = (\log_2(n/2))!
\]

Note that \( m! \geq (m/2)(m/2 + 1) \cdots (m) \geq (m/2)^{m/2} \). Thus, \( d(n) \geq (\log_2(n/2)/2)^{\log_2(n/2)/2} = (n/2)^{f(n)} \) where \( f(n) \to \infty \) as \( n \to \infty \). We claim that this is not \( O(n^c) \) for any \( c \). This is because for some \( N \), \( f(n) > c + 1 \) for all \( n > N \). Then for \( n > \max(N,C2^{c+1}) \) we have that \( d(n) \geq (n/2)^{c+1} = n^22^{-c-1} \geq An^c \). Thus, \( d(n) \) is not \( \Theta(n^c) \) for any \( c \).

(c) \( e(n) = 2^n \)

This is also not \( \Theta(n^c) \) for any \( c \). Once again, this is because it is not even \( O(n^c) \). To show this, we need to show that for any \( c \) and \( A \), there are arbitrarily large values of \( n \) so that \( 2^n > An^c \). On the other hand, consider the function \( f(n) = \frac{n^c}{n^c} \). We claim that this has limit \( \infty \) as \( n \to \infty \). Note that the ratio of consecutive values of \( f \) is \( 2(1 + 1/n)^c \), which goes to \( 2 \) as \( n \to \infty \). Therefore, for sufficiently large \( n \), we have that \( f(n)/f(n-1) \geq 3/2 \). Thus, for \( m \) at least this large and all \( n \), we have that \( f(n) \geq f(m)(3/2)^{n-m} \). Therefore \( f(n) \to \infty \). This proves that \( e(n) \) is not \( O(n^c) \) for any \( c \), and thus that it is not \( \Theta(n^c) \) for any \( c \).

Question 3 (Cycle Finding, 30 points). Recall that a 4-cycle in a graph \( G \) is a collection of four vertices \( v_1, v_2, v_3, v_4 \) so that \((v_1,v_2),(v_2,v_3),(v_3,v_4)\) and \((v_4,v_1)\) are all edges of \( G \).

(a) Show that if \( G \) is a graph with \( |E| \geq 2|V|^{3/2} \), that \( G \) must contain a 4-cycle. Hint: For each vertex \( v \in V \) consider all the pairs \((u,w)\) of vertices so that \( u \) and \( w \) are both adjacent to \( v \). If the same pair \((u,w)\) shows up for two different \( v \)'s, show that there is a 4-cycle.

(b) Find an efficient algorithm to determine whether or not a given graph \( G \) contains a 4-cycle. What is the asymptotic runtime of this algorithm? You should attempt to do better than the trivial algorithm of simply checking all quadruples \((v_1,v_2,v_3,v_4)\) of vertices.

Solution (Cycle Finding). Let \( G = (V,E) \) be an undirected graph.

(a) For each vertex \( v \in V \), let \( d_v \) denote the number of vertices adjacent to \( v \) and let \( W_v \) denote the set of all pairs of vertices adjacent to \( v \), \( \{(u,w) \mid (u,v) \in E \text{ and } (w,v) \in E\} \) (wedges centered around \( v \)). Then, assuming order doesn’t matter, \( |W_v| = \binom{d_v}{2} = \frac{d_v(d_v-1)}{2} \). Let \( W = \sum_{v \in V} |W_v| \) be the number of such pairs for every vertex in the graph. Then,

\[
W = \sum_{v \in V} \frac{d_v(d_v-1)}{2} = \frac{1}{2} \left( \sum_{v \in V} d_v^2 - \sum_{v \in V} d_v \right). \tag{1}
\]

Now, assume for the sake of a contradiction that \( |E| \geq 2|V|^{3/2} \) but \( G \) has no 4-cycle. Since \( G \) has no 4-cycle, it must mean that no \((u,w)\) pair occurs for different lists \( W_v,W_{v'} \) since otherwise \( v,u,v',w \) would form a 4-cycle. Namely, \( W \) is upper-bounded by the number of distinct pairs of vertices:

\[
W \leq \binom{|V|}{2} = \left\lfloor \frac{|V|(|V| - 1)}{2} \right\rfloor. \tag{2}
\]

Combining (1) and (2), we have

\[
\sum_{v \in V} d_v^2 - \sum_{v \in V} d_v \leq |V|(|V| - 1). \tag{3}
\]

We will make use of the following two facts:
Fact 1. \[ \sum_{v \in V} d_v = 2|E|. \]

(Try to prove this for yourselves.)

Fact 2 (Cauchy-Schwarz inequality).
\[
\left( \sum_{i=1}^{n} x_i y_i \right)^2 \leq \left( \sum_{i=1}^{n} x_i^2 \right) \left( \sum_{i=1}^{n} y_i^2 \right).
\]

For the remainder of this proof, let \( n = |V| \) and \( m = |E| \). First, by letting \( x_i = d_v \) and \( y_i = 1 \),
\[
\left( \sum_{v \in V} d_v \right)^2 \leq n \sum_{v \in V} d_v^2.
\]
by Cauchy Schwarz (Fact 2). Plugging this into (3) yields
\[
\left( \sum_{v \in V} d_v \right)^2 - n \sum_{v \in V} d_v \leq n^2(n-1),
\]
and by Fact 1:
\[
(2m)^2 - 2nm \leq n^2(n-1) \quad \text{or},
\]
\[
4m^2 - 2nm - n^2(n-1) \leq 0.
\]
Solving the quadratic equation for \( m \) gives an upper bound of \( m \leq \frac{1}{4}(1 + \sqrt{4n - 3}) \). But this is less than \( 2n^{3/2} \) for \( n \geq 0 \), which contradicts our assumption that \( |E| \geq 2|V|^{3/2} \). Therefore, \( G \) must contain a 4-cycle.

(b) We can use the same infrastructure principle from the above proof to design an efficient algorithm for finding a 4-cycle in a given graph \( G \). Namely, generate the list \( W_v \) for every \( v \in V \), and if any pair \((u, w)\) appears on two different lists, there must be a 4-cycle.

The pairs in each \( W_v \) can be maintained in a \(|V| \times |V|\) square array \( A \), with rows and columns indexed by vertices \( v_1, v_2, \ldots, v_n \). The array is initially all 0’s. Then, for each vertex \( v_i \) in \( V \), the entry \( A[j][k] \) is changed to 1 if \((v_j, v_k) \in W_{v_i} \). If at any point we attempt to change an entry \( A[j][k] \) that has already been set to 1, it must be that \((v_j, v_k)\) was present in two different sets \( W_v, W_{v'} \), and we can conclude that a 4-cycle exists. Below is the algorithm:

let \( n = |V| \)
create an \( n \times n \) array \( A \) of zeros

for every \( v \) in \( V \):
  for every \( u \) adjacent to \( v \):
    for every \( w \) adjacent to \( v \):
      if \( u \) is not \( w \):
        if \( A[u][w] == 1 \):
          output ‘4-CYCLE FOUND’
          break
        else:
          set \( A[u][w] = 1 \)
output ‘NO 4-CYCLES FOUND’
Creating the array requires $O(|V|^2)$ time. Our algorithm performs work for each vertex, $v$ and for each pair of vertices $u$ and $w$. The former work is bounded by $O(|V|)$. For the latter, note that for each such pair we mark an element of our array to 1, and if we ever would have marked the same element twice, we output and abort early. Therefore, the amount of work performed by checking pairs of $u$ and $w$ is $O(|V|^2)$. Thus, the total runtime is $O(|V|^2)$.

**Question 4** (Recurrence Relations, 30 points). Consider the recurrence relation

$$T(1) = 1, \quad T(n) = 2T(\lfloor n/2 \rfloor) + n.$$  

(a) What is the exact value of $T(2^n)$?

(b) Give a $\Theta$ expression for $T(n)$. Hint: compare its value to that at nearby powers of 2.

(c) Consider the following purported proof that $T(n) = O(n)$ by induction:

If $n = 1$, then $T(1) = 1 = O(1)$.

If $T(m) = O(m)$ for $m < n$, then

$$T(n) = 2T(\lfloor n/2 \rfloor) + n = O(n) + O(n) = O(n).$$

Thus, $T(n) = O(n)$.

What is wrong with this proof? Hint: consider the implied constants in the big-_os.

**Solution.** (a) To solve for powers of two, we repeatedly apply the recurrence relation until we reach the base case.

$$T(2^n) = 2T(2^{n-1}) + 2^n$$
$$= 2(2T(2^{n-2}) + 2^{n-1}) + 2^n$$
$$= 4T(2^{n-2}) + 2 \times 2^{n-1} + 2^n$$
$$= 8T(2^{n-3}) + 4 \times 2^{n-2} + 2 \times 2^{n-1} + 2^n$$
$$= 2^nT(1) + n \times 2^n$$
$$= 2^n(n + 1)$$

To verify the solution, we use induction

**Base Case:** $T(1) = T(2^0) = 2^0(0 + 1) = 1$

**Induction hypothesis:** Assume $T(2^{n-1}) = 2^{n-1}n$

**Prove for $T(2^n)$:**

$$T(2^n) = 2T(2^{n-1}) + 2^n$$
$$= 2 \times 2^{n-1}n + 2^n$$
$$= 2^n(n + 1)$$

(b) To establish a bound for the general case $T(n)$, we use the results obtained above. Let $2^l$ be the largest power of 2 such that $2^l \leq n$ and $2^r$ be the smallest power of 2 such that $2^r \geq n$. Thus, we must have $l = \lfloor \log(n) \rfloor$ and $r = \lceil \log(n) \rceil$. Since $T(n)$ is an increasing function, we must have:

$$T(2^l) \leq T(n) \leq T(2^r) \quad \forall n \geq 1$$

$$2^l(l + 1) \leq T(n) \leq 2^r(r + 1) \quad \forall n \geq 1$$

Now $l = \lfloor \log(n) \rfloor > \log(n) - 1$ and $r = \lceil \log(n) \rceil < \log(n) + 1$

Thus, we get

$$2^{\log(n)-1}(\log(n)) < T(n) < 2^{\log(n)+1}(\log(n) + 2) \quad \forall n \geq 1$$

$$\frac{n\log(n)}{2} < T(n) < 2n(\log(n) + 2) \quad \forall n \geq 1$$
Let's consider the two inequalities separately:

\[ T(n) > \frac{n\log(n)}{2} \quad \forall n \geq 1 \]
\[ \Rightarrow T(n) = \Omega(n\log(n)) \]

Similarly,

\[ T(n) < 2n(\log(n) + 2) \quad \forall n \geq 1 \]
\[ \Rightarrow T(n) < 2n(\log(n) + \log(n)) \quad \forall n \geq 5 \]
\[ \Rightarrow T(n) < 4n\log(n) \quad \forall n \geq 5 \]
\[ \Rightarrow T(n) = O(n\log(n)) \]

Thus, we have \( T(n) = \Theta(n\log(n)) \)

(c) Let us analyse the proof given:

To prove that \( T(n) = O(n) \), we must prove that \( T(n) \leq cn \) \( \forall n \geq n_0 \) for some constants \( c \) and \( n_0 \).
Since we assume \( T(m) = O(m) \) \( 1 \leq m < n \), we must have that \( T(m) \leq cm \) \( 1 \leq m < n \) for some constant \( c \). The inductive step tries to prove \( T(n) \leq cn \) for the same constant \( c \).

\[ T(n) = 2T(\lfloor n/2 \rfloor) + n \]
\[ \Rightarrow T(n) \leq 2c\lfloor n/2 \rfloor + n \]

However, we need to prove a stronger bound \( T(n) \leq cn \) which is not implied by this relation. Thus, this proof is invalid.

Essentially, the given proof shows that for every \( N \) there is a constant \( c \) so that for all \( n \leq N \) we have that \( T(n) \leq cn \). Unfortunately, to show that \( T(n) = O(n) \), we need to show that the same constant \( c \) can be used for all sufficiently large \( n \), while the inductive proof requires that we use larger and larger \( c \) as \( n \) grows.