Question 1

(a) $f(n) = 2n^5 - n^3 + 6n^2 + 10n + 5$

Let $g(n) = n^5$

We need to find constants c_1, c_2 and n_0 that:

$c_1 \times g(n) \leq f(n) \leq c_2 \times g(x)$ for all $n > n_0$

For the upper bound ($O(n^5)$)

$2n^5 - n^3 + 6n + 10n + 5 \leq 3n^5$ for all $n > 3$

For all $n > 1$, $c_2 = 3$, we have $f(n) \leq 3n^5$

For the lower bound ($\Omega(n^5)$)

$2n^5 - n^3 + 6n + 10n + 5 \geq n^5$

for all $n > 2$, $c_1 = 1$ we have $f(n) \geq n^5$

Since $f(n)$ is both $O(n^5)$ and $\Omega(n^5)$,

we have $f(n) \in \Theta(n^5)$

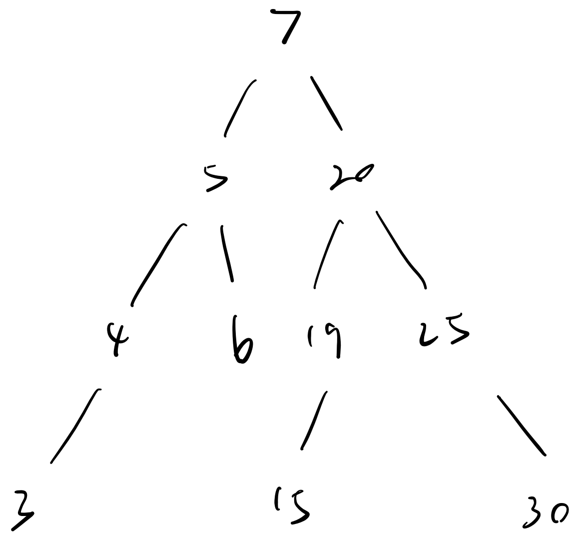
(b) Pre order : 1, 2, 4, 5, 8, 3, 6, 7, 9, 10

In order : 4, 2, 8, 5, 1, 6, 3, 9, 7, 10

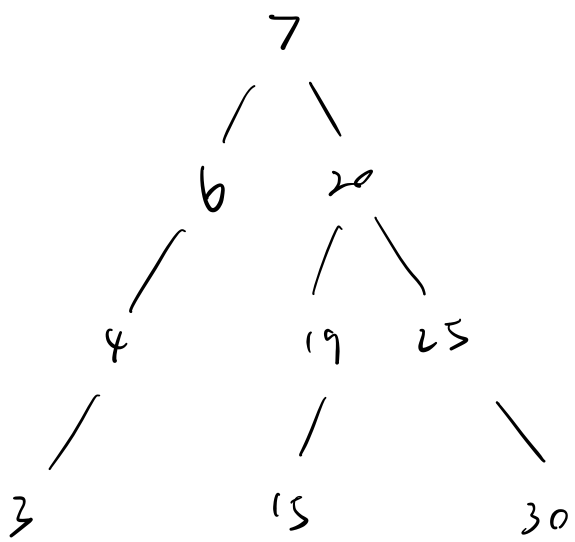
Post order : 4, 8, 5, 2, 6, 9, 10, 7, 3, 1

(c)

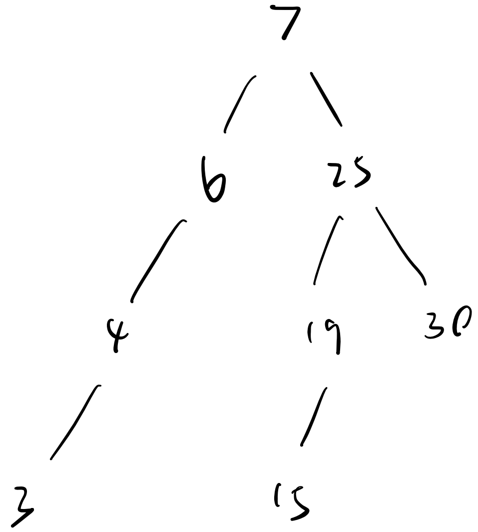
(i)



(ii)



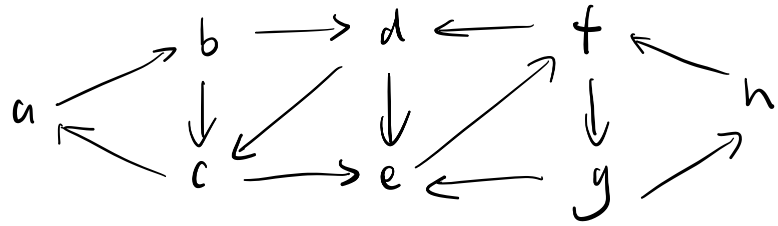
(iii)



(d)

(i) False (ii) True (iii) False

(e)



{a, b, c, d, f, h, g, e}, one SCC

Question 2

(a)

	a	b	c	d	e	f	g	h
pre	1	2	10	3	4	5	13	14
post	12	9	11	8	7	6	16	15

(b) Tree edge : (a,b), (b,d), (e,f), (d,e), (a,c), (g,h)

Cross edge : (e,a)

Forward edge : (a,d), (a,f), (d,f)

Back edge : (c,e), (c,d), (g,b), (h,d)

(c) {a, b, c, d, e}, {f}, {g}, {h}

Question 3

(a) Algorithm A

$T(n) = 4T(\frac{n}{2}) + \Theta(n^2 \log n + \log n)$

$a = 4,\ b = 2,\ f(n) = \Theta(n^2 \log n + \log n)$

Algorithm B

$T(n) = 2T(\frac{n}{4}) + \Theta(\log n)$

$a = 2,\ b = 4,\ f(n) = \Theta(\log n)$

Algorithm C

$T(n) = 3T(\frac{n}{9}) + \Theta(\frac{n^{0.51}}{\log n})$

$a = 3,\ b = 9,\ f(n) = \Theta(\frac{n^{0.51}}{\log n})$

(b) $T(n) = 4T(\frac{n}{2}) + \Theta(n^2 \log n + \log n)$

According to Case 2, $T(n) = \Theta(n^{\log_b a} \log^{k+1} n) = \Theta(n^2 \log^2 n)$

$T(n) = 2T(\frac{n}{4}) + \Theta(\log n)$

According to Case 1, $T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\frac{1}{2}})$

$T(n) = 3T(\frac{n}{9}) + \Theta(\frac{n^{0.51}}{\log n})$

According to Case , $T(n) = \Theta(f(n)) = \Theta(\frac{n^{0.51}}{\log n})$

(c) fastest to slowest : B, C, A

Question 4

(a) A1 depend on B1 (A1 = B1 × 2)

A2 depend on A1 and B1 (A2 = A1 + B1)

B2 depend on A1 (B2 = A1 + 5)

A3 depend on A1 and A2 (A3 = Sum(A1:A2))

B3 depend on B2 and A2 (B3 = B2/A2)

$$
\begin{tikzcd}
B1 \arrow[r] & A1 \arrow[r] \arrow[d] & B2 \\
 & A2 \arrow[r] \arrow[d] & A3 \\
 & B2 \arrow[r] & B3
\end{tikzcd}
$$

(b) Compute (cells)

For each cell C:

For each dependency D of C, add edge D → C

inDegree [C] = number of dependecies

Init queue Q with all cells C where inDegree [C] = 0

While Q is not empty:

Dequeue a cell C from Q

Compute C's value

For each node N that depend on C:

inDegree[n] -= 1

if inDegree[n] == 0, then enqueue N

All cells are computed

(c)

$$\begin{tikzcd} 1\,B_1 \arrow[r] & 2\,A_1 \arrow[r] \arrow[d] & 4\,B_2 \arrow[r] & B_3\ 6 \\ & 3\,A_2 \arrow[r] & 5\,A_3 & \end{tikzcd}$$

(d) Building the Graph : $O(n+m)$

Topological Sorting : $O(n+m)$

So overall time is $O(n+m)$, which is linear

Question 3

(a)

1. Choose the middle
2. Check if mid is local min
3. If left A[mid-1] is less than A[mid], go left half.
   else, go right half
4. Recursion

```
FindLocal (A, start, end):
    if start == end:
        return start
    mid = (start + end)/2
    if (mid == 1 or A[mid-1] >= A[mid])
        and
        (mid == len(A) or A[mid] <= A[mid+1]):
        return mid

    if mid > start and A[mid-1] < A[mid]:
        return findLocal (A, start, mid-1)

    else:
        return findLocal (A, mid+1, end)
```

(b) $T(n) = T(\frac{n}{2}) + \Theta(1)$

(c) $a=1, b=2 \rightarrow n^{\log_b a} = n^{\log_2 1} = n^0 = 1$

$f(n) = \Theta(1)$ which match $\Theta(n^0)$

by case 2, we can get $T(n) = \Theta(\log n)$

(d) Call 1: start = 1, end = 16, mid: A[8] = 5,

A[7] = 7 ≥ 5, but A[9] = 4 ≤ 5, not local min - go right half

Call 2: start = 9, end = 16, mid: A[12] = 3,

A[11] = 3 ≥ 3, and A[13] = 4 ≥ 3, get the local min