Question 1: (Question 5 of Assignment 1).

Let G have two connected components G_1 and G_2. Let G_1 have n_1 vertices and G_2 have n_2 vertices. It is easy to observe that $\min\{n_1, n_2\} \leq \frac{n}{2}$. Let $n_1 \leq \frac{n}{2}$. For every node in G_1 can have a degree of at most $\frac{n}{2}$. This contradicts the fact that every node in G has a min degree of $\frac{n}{2}$.

We can generalize this argument to k connected components and get same implication.

Grading instruction: No partial marking for question 1.

Question 2:

(a) Third roots of unity are $1, \frac{-1+\sqrt{3}\,i}{2}, \frac{-1-\sqrt{3}\,i}{2}$.

Primitive root is $\frac{-1+\sqrt{3}\,i}{2} \leftarrow$ call this ω. Alternate sln $\rightarrow e^{\frac{2\pi i}{3}}$

$\omega^2 = \frac{-1-\sqrt{3}\,i}{2} \neq 1$; and $\omega^3 = 1$.

No partial marking.

(b) DFT matrix

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega^4 \end{bmatrix}$$

$\omega^4 = \omega$

Full mark for either of these

$$= \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{bmatrix}$$

Inv DFT matrix

Full marks for either.

$$\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega^{-1} & \omega^{-2} \\ 1 & \omega^{-2} & \omega^{-4} \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega^2 & \omega \\ 1 & \omega & \omega^2 \end{bmatrix}$$

Grading Instruction: 1 mark for DFT and 1 mark for inv DFT. No further partial marking.

(c) DFT of (1,1,1)

Acceptable solution.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1+1+1 \\ 1+\omega+\omega^2 \\ 1+\omega^2+\omega \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix}$$

No partial marking.

Question 3:

2 marks for running time.

(5.16)

$$T(n) \le T(n/2) + c$$

when $n > 2$, *and*

$$T(2) \le c.$$

Solving this we get $T(n) = O(\log n)$.

So suppose we look at the value $A[n/2]$. From this value alone, we can't tell whether p lies before or after $n/2$, since we need to know whether entry $n/2$ is sitting on an "up-slope" or on a "down-slope." So we also look at the values $A[n/2-1]$ and $A[n/2+1]$. There are now three possibilities.

- If $A[n/2-1] < A[n/2] < A[n/2+1]$, then entry $n/2$ must come strictly before p, and so we can continue recursively on entries $n/2+1$ through n.
- If $A[n/2-1] > A[n/2] > A[n/2+1]$, then entry $n/2$ must come strictly after p, and so we can continue recursively on entries 1 through $n/2-1$.
- Finally, if $A[n/2]$ is larger than both $A[n/2-1]$ and $A[n/2+1]$, we are done: the peak entry is in fact equal to $n/2$ in this case.

In all these cases, we perform at most three probes of the array A and reduce the problem to one of at most half the size. Thus we can apply (5.16) to conclude that the running time is $O(\log n)$.

Every given array contains a peak ← 2 marks for this

- If there are no elems s.t $A[i-1] \le A[i] \ge A[i+1]$, look at boundary cases. they give the peak
- Else, peak is given by an elem s.t $A[i-1] \le A[i] \ge A[i+1]$.

Question 4: (Expected solution)

3 marks for algorithm. (No partial marks for incorrect solutions)

Algorithm:

- Put the item with max value to weight ratio, in as high a quantity as possible.
- If "space" is left in the bag, pick the next item with max value to weight ratio. Repeat.

3 marks for correctness. (may give partial marks)

Correctness: Optimal solution contains items in ~~decreasing~~ non-increasing order of their value to weight ratios. Suppose (for the sake of contradiction) the optimal solution picks x amount of chocolate j when x amount of chocolate i was still left where $\frac{v_i}{w_i} > \frac{v_j}{w_j}$. But by swapping out x amount of j for x amount of i will give us a solution with a higher value, contradicting the optimality.

Question 5:

```
Merge-and-Count(A,B)
  Maintain a Current pointer into each list, initialized to
    point to the front elements
  Maintain a variable Count for the number of inversions,
    initialized to 0
  While both lists are nonempty:
    Let a_i and b_j be the elements pointed to by the Current pointer
    Append the smaller of these two to the output list
    If b_j is the smaller element then
      Increment Count by the number of elements remaining in A
    Endif
    Advance the Current pointer in the list from which the
      smaller element was selected.
  EndWhile
```

```
Once one list is empty, append the remainder of the other list
    to the output
Return Count and the merged list
```

1. Merge-and-Count (A,B) takes $O(\max\{|A|, |B|\})$ time.
2. Sort-and-Count (L) takes $O(|L \log L|)$ time.

$$T(n) = T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + O(n)$$

$$\approx 2T\left(\frac{n}{2}\right) + O(n)$$

$$\approx O(n \log n)$$

5 marks for filling in the code. No partial marking for incomplete code.

3 marks for algo analysis. Partial marking for framing the recursive equation and run time.