1 Задача 1

Пусть E — множество вершин. Заведем 2 фукнции $first$ и $count$ — первая возвращает первый единичный бит маски, вторая — количество единичных битов.
Заведем массив dp, в $dp[mask][i]$ будем хранить длину минимального пути от $first(mask)$ до i в множестве вершин $mask$. Проинициализируем $dp[2^i][i]$ нулями для всех i, остальные бесконечностями. Будем считать динамику следующим образом:

$$dp[mask][i] = \begin{cases} 0, \text{ если } count(mask) = 1 \text{ и } mask[i] = 1; \\ \min\limits_{2^j \in mask - 2^i, (j,i) \in E} dp[mask - 2^i][j] + w(j, i), \text{ если} \\ count(mask) > 1, \ mask[i] = 1 \text{ и } i \neq first(mask); \\ \infty, \text{иначе.} \end{cases}$$

Докажем корректность подсчета dp: считая $dp[mask][i]$ мы должны взять минимум из всех $dp[mask - i][j]$, где j — пробегает все возможные вершины $mask - 2^i$ на которые может заканчиваться гамильтонов путь из $first(mask - i)$ и проверяет есть ли между j и i ребро, если есть, то у нас существует гамильтонов путь длины $dp[mask-2^i][j]+w(j, i)$ из $first(mask)$ в i где последнее ребро — (j, i), тогда взяв минимум по всем j получим корректно насчитанное dp.
Найдем ответ для $mask = A$: заведем массив ответа ans. Переберем все вершины i в A кроме $first(mask)$, тогда, очевидно, кратчайший гамильтонов цикл $\min\limits_{i \in A, (i, first(mask) \in E)} (dp[mask][i] + w(i, first(mask)))$ тоесть ребро $(i, first(mask))$ и минимальный гамильтонов путь от $first(mask)$ до i. Положим это i в ans, тогда рекурсивно будем искать остальные вершины цикла функцией от $mask$ и i — запустим ее от A, i. Она будет устроена так: переберем j аналогично как в построении и если $dp[mask - 2^i][j] + w(j, i) = dp[mask][i]$, то мы пришли из этой вершины и она входит в цикл — положим j в ans и запустимся от $mask - 2^i$, j. Когда $count(mask)$ станет равно 1 положим j в ответ и закончим алгоритм. Тогда в ans после выполнения алгоритма будет содержаться минимальный по весу гамильтонов цикл в A в порядке обхода.

2 Задача 2

Даны две последовательности X и Y, длины m и n соответвенно. Не теряя общности, будем считать, что $m \geq n$. Разобьем X на две равные строки $x_1[0...\frac{m}{2}]$ и $x_2[\frac{m}{2}+1...m]$. Найдем длину НОП для x_1 и всех префиксов Y, и для развернутого x_2 и всех развернутых префиксов Y. Затем выберем такой i, что НОП$(x_1, y[1...i])$ + НОП$(reverse(x_2), reverse(y[i+1...n]))$ максимально. Запустим алгоритм рекурсивно для пар $(x_1, y[1...i])$ и $(x_2, y[i+1...n])$. Будем продолжать, пока в x не останется одного элемента, тогда проверим проходом по y содержится ли он в ней, и если да, то положим его в строку ответа.
Доказательство

1. Корректность:
   Рассмотрим какое-то разделение X на две части. Какая-то часть НОП лежит в первой половине, оставшаяся во второй. Пусть x_i — последний символ из НОП в первой половине, наш алгоритм найдет для него соответсвующий y_i, тоесть все символы из y связанные со второй половиной x будут правее, а все, что первой — левее $\implies$ мы свели поиск исходной НОП к поиску двух независимых частей.

2. Время работы:
   Рекурсия представима в виде дерева с глубиной $\log_2(m)$, на глубине h находятся 2^h вершин с частью X длины $\frac{m}{2^h}$ и частью Y длины k_i, где сумма всех k_i равна n. Тогда на глубине h получаем:
   $\sum\limits_{i=0}^{2^h-1} \frac{m}{2^h}k_i = \frac{m}{2^h}\sum\limits_{i=0}^{2^h-1} k_i = \frac{mn}{2^h}$
   Общее время работы:
   $\sum\limits_{i=0}^{\log m} \frac{mn}{2^i} = mn\sum\limits_{i=0}^{\log m} \frac{1}{2^i} < mn\sum\limits_{i=0}^{\infty} \frac{1}{2^i} = 2nm \implies$ итоговая ассимптотика — $O(mn)$

3. Память
   Последовательности X, Y и ответ требуют $n + m + \min(n, m)$ памяти. Дополнительно на каждом шаге рекурсии вызываются две функции НОП которые суммарно требуют $4k_i$, где k_i — длина части Y на данном шаге, так как для нахождения длины НОП нужно 2 строки матрицы. Эти массивы удаляются перед рекурсивным вызовом.
   Затраты памяти для поиска длины НОП на глубине h

$\sum_{i=0}^{2^h-1} 4k_i = 4 \sum_{i=0}^{2^h-1} k_i = 4n$
Тогда итоговые затраты памяти: $n + m + \min(m, n) + 4n = O(n + m)$

3 Задача 3

Будем доказывать от противного. Пусть оптимально набрать такие предметы, что суммарный вес предметов с максимальным отношением меньше $W - z^2$. $W = w_i k + (w_k + ... + w_j)$(элементы в скобке — какой-то набор элементов не включающий i). В силу того, что $(w_k + ... + w_j) > z^2$, количество этих предметов больше z, так как $w_j \leq z$ для всех j. Посчитаем все префиксные суммы набора этих элементов не равных i в массиве $pref$, заведем второй массив ost, $ost = pref \mod w_i$, тогда, в силу того, что элементов в скобке больше z, а $z \geq w_i$ — найдется $ost[j] = ost[k] \implies (pref[k] - pref[j]) \mod w_i = 0$, тоесть существует набор элементов не равных i с суммой кратной w_i, тогда заменим их на соответствующее количество элементов i, так как они более оптимальны для этого веса. Пришли к противоречию — взятый набор неоптимален $\implies$ чтобы набрать максимальный вес нужно взять предметов с максимальным отношением $\frac{c_i}{w_i}$ с суммарным весом хотя бы $W - z^2$.

4 Задача 4

Пусть E — множество вершин. Заведем 2 функции $first$ и $count$ — первая возвращает первый единичный бит маски, вторая — количество единичных битов.
Заведем массив dp, в $dp[mask][i]$ будем хранить количество гамильтоновых путей от $first(mask)$ до i в множестве вершин $mask$. Проинициализируем $dp[2^i][i]$ единицами для всех i, остальные нулями.
Будем пересчитывать динамику так:

$$dp[mask][i] = \begin{cases} 1, \text{ если } count(mask) = 1 \text{ и } mask[i] = 1; \\ \sum\limits_{2^j \in mask - 2^i, (j,i) \in E} dp[mask - 2^i][j], \text{ если} \\ count(mask) > 1, \ mask[i] = 1 \text{ и } i \neq first(mask); \\ 0, \text{ иначе.} \end{cases}$$

Докажем корректность подсчета dp: считая $dp[mask][i]$ мы должны прибавить в него все $dp[mask - 2^i][j]$, где j — пробегает все возможные вершины $mask - 2^i$ на которые может заканчиваться гамильтонов путь из $first(mask - 2^i)$ и проверяет есть ли между j и i ребро, если есть, то у нас существует $dp[mask - 2^i][j]$ гамильтоновых путей из $first(mask)$ в i где последнее ребро — (j, i), тогда просуммировав по всем j получим корректно насчитанное dp.

Ответом на задача будет:

$\frac{1}{2}\sum_{i \in [1 \dots n], mask \in [1 \dots 2^n - 1], count(mask) \geq 3, (first(mask), i) \in E} dp[mask][i]$

Мы перебираем все возможные подмножества вершин и пытаемся замкнуть гамильтонов путь в них в цикл, мы можем это делать если у нас есть ребро между $first(mask)$ и i. Почему делим на два? Вот почему — пусть у нас есть какой то цикл, тогда в силу неориентированности графа мы посчитаем его дважды, так как можно пройти цикл в две стороны(один и тот же цикл мы можем замкнуть с двух сторон).