**Задача 1.3 (С).** В городке Урюполе только один автобусный маршрут, соединяющий вокзал с главной местной достопримечательностью — продуктовым рынком, славящимся на всю округу большим ассортиментом и низкими ценами.

В Урюполь недавно пришел поезд из соседнего городка Крыжопинска, и на автобусной остановке возле вокзала образовалась очередь из *N* человек, желающих попасть на рынок.

В связи с этим, для развозки пассажиров к остановке собираются подать *M* автобусов вместимостью *D* каждый. Известно, что если пронумеровать людей от 1 до *N* в порядке очереди, то *i*-й из них при посадке в автобус займет *L(i)* единиц объема.

Однако автобус — не единственный транспорт в Урюполе: если человек устал ждать в очереди, он может выйти из очереди, сесть на такси и тут же уехать. При этом относительный порядок оставшихся в очереди людей не меняется.

Посадка в автобусы происходит следующим образом. Автобус подъезжает к остановке, открывает переднюю дверь, и в нее заходят люди в порядке очереди. Как только для очередного человека не хватает места, автобус закрывает дверь и уезжает, после чего к остановке подходит следующий автобус (если он есть).

Поскольку зарплата водителя автобуса зависит от количества перевезенных пассажиров, водители хотят знать, какое наибольшее суммарное количество людей из очереди они могут перевезти. Помогите им.

Первая строка содержит число *M* ().

Вторая строка — *D* ().

Третья строка — *N* ().

Четвёртая строка — *L(1)* *L(2)* … *L(N)*. Все .

Все входные параметры — натуральные числа.

Необходимо вывести единственное число — искомое количество людей.

**Решение.**

Для решения задачи будем использовать подход динамического программирования.

1. Создадим вектор L, который будет хранить объем автобуса, необходимый для каждого пассажира.
2. Создадим трехмерный вектор T размерностью (M) x (N + 1) x (D + 1) и заполним его нулями. Будем использовать его для сохранения промежуточных результатов исчислений. M – количество автобусов, N – количество возможных пассажиров, D – вместимость одного автобуса.
3. Заполним первый уровень массива T[0][j][k], для каждого проходим последовательно :

3.1. Присваиваем элементу T[0][j][k] максимальное значение с T[0][j - 1][k] и T[0][j][k - 1]. Выбирая T[0][j - 1][k] мы предполагаем, что пассажир уехал на такси, выбирая T[0][j][k - 1] мы оставляем единицу объема автобуса пустотой;

3.2. Если k больше или равно объема необходимого для пассажира с номером j – 1: если 1 + T[0][j - 1][k – L[j - 1]] больше чем текущее значение T[0][j][k], то присваиваем первое в T[0][j][k]. Это значит, что пассажир под номером j – 1 сел в этот автобус, иначе переходим к другому значению k.

1. Последовательно заполним массив T[i][j][k]: , , :
   1. Если k = 0 (автобус пустой), иначе переходим к 4.2:
      1. Присваиваем текущему элементу (T[i][j][0]) значение T[i - 1][j][D];
   2. Присваиваем элементу T[i][j][k] максимальное значение с T[i][j - 1][k] и T[i][j][k - 1]. Выбирая T[i][j - 1][k] мы предполагаем, что пассажир уехал на такси, выбирая T[i][j][k - 1] мы оставляем единицу объема автобуса пустотой;
   3. Если k больше или равно объема необходимого для пассажира с номером j – 1: если 1 + T[i][j - 1][k – L[j - 1]] больше чем текущее значение T[i][j][k], то присваиваем первое в T[i][j][k]. Это значит, что пассажир под номером j – 1 сел в этот автобус, иначе переходим к другому значению k.
2. После заполнения вектора максимальное количество пассажиров, которых можно посадить в имеющиеся автобусы хранится в ячейке T[M - 1][N][D].

**Доказательство:**

Окончательный результат задачи мы получаем, основываясь на решения подзадач.

Рассмотрим состояние T[i][j][k], которое может иметь значение (T[i][j-1][k], T[i][j-1][k-L[j] + 1, T[i][j][k-1]) {1}. В ходе выполнения алгоритма в это состояние будет записано максимальное значение из возможных. Выбрав не самое большое возможное значение (с {1}), мы получим меньшее решение выбранной подзадачи, соответственно для каждой следующей подзадачи мы будем получать меньшее или равное от оптимального. В итоге мы получим результат, который будет либо меньшим, либо таким же как результат, получаемый алгоритмом. Такое утверждение истинно для любого i, j, k.

Вектор для хранения промежуточных значений имеет размер

M x N x D. Вектор обходится один раз, поэтому сложность алгоритма O(M·N·D), затраты по памяти O(M·N·D).