Review

**Задача C. Автобусы**

Условие:

Ограничение времени 1 секунда

Ограничение памяти 64Mb

Ввод стандартный ввод или c.in

Вывод стандартный вывод или c.out

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.

Формат ввода

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.

Формат вывода

Необходимо вывести единственное число — искомое количество людей.

Решение:

1. Создадим массив занимаемых людьми объемов int peopleWeight[301].
2. Создадим матрицу для динамики int arr[301][301]. Несмотря на то, что динамика трехмерная, каждой следующей итерации по первому параметру(автобусы) нужны данные только предыдущей итерации, причем тех ячеек, которые гарантированно не были затронуты текущей итерацией. Поэтому хранить достаточно двумерную матрицу.
3. M раз (итератор не важен):
4. N раз, j= 1..N+1:
5. D раз, k = 0..D+1:
6. На каждом шаге динамики нужно решить, какое действие будет наиболее выгодным. Шагом здесь является переход к следующему человеку для конкретной максимальной вместимости. По сути имеем два варианта: либо поместить этого человека в текущий автобус, либо отправить его на такси. В этом случае формула абсолютно аналогична таковой в задаче про рюкзак: arr[j - 1][k - peopleWeight[j]] + 1. В другом случае максимальным значением является либо аналогичное для прошлого человека: arr[j - 1][k], либо оптимальное при максимальном наполнение предыдущего автобуса(когда ни один человек в текущий автобус еще не сел). Таким образом: arr[j][k] = std::max(arr[j - 1][k - peopleWeight[j]] + 1, std::max(arr[j - 1][k], arr[j][capacity])). Естественно, первый вариант не учитываем, если k – peopleWeight[j] < 0.

Легко заметить, что при таком алгоритме размер матрицы составляет D\*M, соответственно затраты по памяти – O(D\*M).

По времени алгоритм представляет собой трехмерную динамику по D, M и N. Таким образом, затраты – O(D\*M\*N).