**Задача Автобусы.**

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* (https://contest.yandex.ru/testsys/tex/render/XGxlcXNsYW50IDEwMA==.png). Вторая строка — *D* (https://contest.yandex.ru/testsys/tex/render/XGxlcXNsYW50IDMwMA==.png). Третья строка — *N* (https://contest.yandex.ru/testsys/tex/render/XGxlcXNsYW50IDMwMA==.png). Четвёртая строка — *L(1)* *L(2)* … *L(N)*. Все https://contest.yandex.ru/testsys/tex/render/TChpKVxsZXFzbGFudHt9RA==.png. Все входные параметры — натуральные числа.

Необходимо вывести единственное число — искомое количество людей.

**Решение.**

Задача решается методом динамического программирования. Будем рассматривать следующий набор подзадач: какое максимальное число пассажиров можно перевезти, если уже уехало *i* автобусов, были перевезены некоторые из пассажиров с 1-го по *j*-ый и в *i*-м автобусе занято *k* единиц объема (*d*(*i, j, k)*).

*d*(*i, j, k)* можно получить следующим образом:

В первом варианте текущий человек (*j*-ый) не садится в автобус, а уезжает на такси. При этом *j* должно быть больше 0.

Во втором варианте текущий человек (*j*-ый с объемом *L*[*j*]) садится в автобус. При этом *j* должно быть больше 0, а *k* больше либо равно *L*[*j*].

В третьем варианте мы занимаем единицу объема пустотой. Это сделано с целью учета частично заполненного автобуса, когда остаются пустые единицы объема. При этом *k* должно быть больше 0.

Кроме того, при переходе к новому автобусу (один автобус уезжает, а новый приезжает) получается следующее:

Это выходит из того, что состоянию нового приехавшего автобуса с 0 занятых единиц объема соответствует состояние последнего уехавшего автобуса со всеми занятыми единицами объема (*D*). Благодаря учету частичного заполнения автобусов для данного присвоения нам не нужно искать максимальное значение из *d(i-1,j,k)*(*k* от 0 до *D*).

База динамики:

Окончательный ответ будет находиться в *d(M-1,N,D)*.

Таким образом алгоритм будет иметь следующий вид:

1. Задаем трехмерный массив для хранения ответов подзадач (res).
2. Записываем базу динамики(res[0][0][0] = 0).
3. Далее проходим по всем элементам массива (рассматриваем все подзадачи (res[i][j][k]) от 0 до М не включая, от 0 до N, от 0 до D):
4. В случае j > 0, k >= L[j], k > 0 нам подходит максимальный из трех вариантов рассмотренные выше:
5. В ином случае если j = 0, k > 0 нам подходит только последний вариант:
6. В ином случае если j > 0, k<L[j], k > 0 нам подходит максимальный из первого и второго:
7. В случае перехода к новому автобусу (k = 0, i > 0):
8. По завершению прохода выводим в качестве ответа res[M-1][N][D].

Алгоритм является корректным, так как он в любом случае найдет какое-то количество увезенных автобусами людей.

Так как в текущее состояние есть максимум 3 варианта перехода () и мы берем максимальное значение этих состояний (если они возможны) то в каждом следующем состоянии мы получаем большее или равное значение чем у предыдущих. Отсюда следует что в последнем состоянии (L[M-1][N][D] – состояние с ответом) мы получим максимальное значение.

Общая сложность алгоритма – O(M\*N\*K).

Затраты по памяти – O(M\*N\*K) на хранение трехмерного массива, необходимого для хранения ответов подзадач.