Задача 1.3 «Автобусы»
Решение Евченко Богдана ПИ-15-1
https://github.com/olderor/advanced1

# Автобусы

| | |
|---|---|
| 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 $M$ ($\leqslant 100$).

Вторая строка — $D$ ($\leqslant 300$).

Третья строка — $N$ ($\leqslant 300$).

Четвёртая строка — $L(1)$ $L(2)$ ... $L(N)$. Все $L(i) \leqslant D$.

Все входные параметры — натуральные числа.

## Формат вывода

Необходимо вывести единственное число — искомое количество людей.

## Пример

| Ввод | Вывод |
|---|---|
| 2<br>10<br>4<br>6 6 6 4 | 3 |

### Решение.

1. Создадим и считаем переменные $busCount$ – количество автобусов ($M$ по условию), $busCapacity$ – вместимость автобусов ($D$ по условию) и $peopleCount$ – количество людей ($N$ – по условию).

1. Создадим массив $peopleWeights$ размером $peopleCount$ элементов. Тут будут храниться веса каждого пассажира ($L(i)$ по условию). Заполним массив значениями из условия.

2. Создадим массив $weightsPerOneBus$ размером $peopleCount * peopleCount$ элементов. Тут будут храниться максимальное количество пассажиров из диапазона от $i$ до $j$, $i \leq j$, которые могут поместиться в один автобус.

2.1. Для каждой строки $i$ посчитаем это максимальное количество. Для этого создадим набор $currentWeights$, в котором будем хранить набор весов пассажиров, а также две переменные $currentWeightsSum$ – сумма элементов набора, то есть сумма весов всех пассажиров, которых мы можем посадить в автобус, и $currentWeightsSize$ – количество элементов в наборе, то есть количество пассажиров, которые мы можем посадить в автобус.

2.2 Переберем всех пассажиров с индексами $j$ от $i$ до $peopleCount$. Далее рассмотрим следующие случаи:

2.2.1 В автобусе есть достаточно свободного места, и мы можем спокойно посадить пассажира $j$ в автобус, то есть $peopleWeights(j) \leq busCapacity - currentWeightsSum$. Добавляем вес пассажира $j$ в набор $currentWeights$, увеличиваем сумму $currentWeightsSum$ на вес пассажира $j$ и инкрементируем переменную $currentWeightsSize$.

2.2.2 Если пассажир не влезает в автобус, необходимо взять максимальный вес пассажира $top$, который находится в наборе $currentWeights$. Если вес $top$ больше, чем вес $peopleWeights(j)$ пассажира $j$, заменим этот вес на вес пассажира $j$, а также изменим вместимость $currentWeightsSum += peopleWeights(j) - top$. Таким образом мы не меняем максимальное количество пассажиров, однако увеличиваем доступную вместимость автобуса, что

дает нам больше свободы в будущем при добавлении новых пассажиров. Иначе следует пропустить пассажира, поскольку его добавление не делает вместимость оптимальным.

2.3 Запишем полученное максимальное количество в массив $weightsPerOneBus(i)(j) = currentWeightsSize$.

Исходя из этого, можно сделать вывод, что для хранения набора $currentWeights$ следует использовать структуру данных $priority\_queue$ для того, чтобы мы могли быстро находить максимальное значение в этом наборе. А значит, что значение $top$ (максимальный вес пассажира в набое) мы можем получить используя $currentWeights.top()$ за $O(1)$. Для того, чтобы заменить максимальный вес пассажира $top$ на вес $peopleWeights(j)$ пассажира $j$ следует достать максимальный вес $top$ с помощью $currentWeights.pop()$, который работает за $O(\log\ n)$, где $n$ – количество элементов в наборе ($currentWeightsSize$), и добавить новый вес $currentWeights.push(peopleWeights(j))$, который работает за $O(\log\ n)$, где $n$ – количество элементов в наборе ($currentWeightsSize - 1$). Поскольку $currentWeightsSize$ может принимать максимальное значение $N$ – количество людей $peopleCount$, то время работы изменения очереди в целом будет работать за $O(\log N)$.

3. Создадим массив $optimalWeights$ размером $peopleCount * busCount$. Заполним его с помощью динамического программирования, разбив задачу на подзадачи «максимальное количество пассажиров $optimalWeights(i)(j)$, которое можно поместить в количество автобусов $j$, взяв пассажиров от 0 до $i$. Поскольку у нас уже имеются такие данные для одного автобуса, то база динамики – $optimalWeights(i)(j) = weightsPerOneBus(0)(i)$. Для заполнения будем брать максимальное $optimalWeights(i)(j)$, то есть $optimalWeights(i)(j) = \max(optimalWeights(i)(j),$ $optimalWeights(k)(j-1) + weightsPerOneBus(k+1)(i))$, где $k$ от 0 до $i$. То есть либо $optimalWeights$ для пассажиров от 0 до $i$ для количества автобусов $j$ уже есть оптимальным, либо мы можем взять больше пассажиров, если возьмем оптимальное количество пассажиров $optimalWeights(k)(j-1)$ для пассажиров от 0 до $k$ для количества автобусов $(j-1)$, а также добавим новый автобус для остальных пассажиров $weightsPerOneBus(k+1)(i)$ – максимальное количество пассажиров на отрезке от $k+1$ до $i$.

4. Ответ будет храниться в $optimalWeights(peopleCount-1)(busCount-1)$ – максимальное количество пассажиров, которые можно посадить в $busCount$ автобусов, используя $peopleCount$ пассажиров.

## Доказательство

В пункте 2 заполняем массив $weightsPerOneBus$. Перебирая пассажиров, заполняем очередь $currentWeights$, с помощью которой определяется максимальное количество пассажиров на заданном промежутке. При рассмотрении нового пассажира следует рассмотреть такие случаи: 1) пассажир вмещается в автобус – помещаем пассажира в автобус и увеличиваем максимальное количество, 2) пассажир не помещается в автобус: 2.1) если объем этого пассажира меньше, чем максимальный объем пассажира, который уже находится в автобусе, следует их заменить, таким образом количество не меняется, но увеличивается свободное место, что даст нам больше возможностей для помещения следующих пассажиров, 2.2) объем пассажира больше, чем максимальный объем пассажира, который уже находится в автобусе, то этого пассажира следует пропустить, поскольку на количество это не влияет, а заменив их только потеряется свободное место, которое мог бы занять какой-либо из следующих пассажиров. Следовательно, после рассмотрения каждого из пассажиров мы будем иметь максимальное возможное количество пассажиров на заданном промежутке.

В пункте 3 заполняем массив $optimalWeights$. Перебирая все возможное количестов автобусов, а также в каждый раз перебирая пассажиров на отрезках от 0 до $k$ и от $k+1$ до $i$ будут перебраны все возможные варианты размещения в автобусе. Для этого необходимо каждый раз сравнивать текущее оптимально размещение и выбирать наибольшее: либо в $j$ автобусах уже хранится максимальное количество пассажиров, либо можно взять $j-1$ автобусов с уже оптимальным размещением пассажиров от 0 до $k$, которое мы нашли раннее, и добавить новый автобус с пассажирами от $k+1$ до $i$. Таким образом, на каждой итерации будет выбираться наибольшее количество пассажиров, а поскольку изначальные данные $weightsPerOneBus$,

которые были занесены в $optimalWeights$ с количеством автобусов 1, были уже оптимальными, то и каждое вычисление для следующего количества автобусов будет тоже оптимальным. Это значит, что выбрав $M$ автобусов с пассажирами на промежутке от 0 до $N$, получим максимальное количество пассажиров, используя $M$ автобусов и $N$ людей, что и есть ответом на задачу. Следовательно, алгоритм работает правильно.

## Итоги

Оценка памяти. Для решения задачи потребовалось: массив $peopleWeights$ размером $N$; массив $weightsPerOneBus$ размером $N * N$; очередь $currentWeights$, максимальный размер которой мог достичь $N$; а также массив $optimalWeights$ размером $N * M$. Следовательно, полная сложность алгоритма по памяти равна $O(N + N * N + N + N * M) \approx O(N * N + N * M)$.

Оценка времени работы. Для решения задачи потребовалось: считать данные в массив $peopleWeights$ за $O(N)$; заполнить массив $weightsPerOneBus$ с помощью двух вложенных циклов по $i$ и $j$, за $O(N)$ каждый, а также изменять очередь за $O(\log N)$, что в целом составляет $O(N * N * \log N)$; а также заполнить массив $optimalWeights$ с помощью трех вложенных циклов по $j$, который работал за $O(M)$ и $i, k$, которые работали за $O(N)$ каждый, что в целом составляет $O(M * N * N)$. Следовательно, полная сложность алгоритма по времени работы равна $O(N + N * N * \log N + M * N * N) \approx O(M * N * N + N * N * \log N)$.

Итог:

- затраты по памяти $O(N * N + N * M)$.
- затраты по времени $O(M * N * N + N * N * \log N)$.