ЕГЭ по Информатике 2021

Задание 26

Роман Чулков

СПбГУТ

29 апреля 2021 г.

Условие задачи

Системный администратор раз в неделю создаёт архив пользовательских файлов. Однако объём диска, куда он помещает архив, может быть меньше, чем суммарный объём архивируемых файлов. Известно, какой объём занимает файл каждого пользователя.
По заданной информации об объёме файлов пользователей и свободном объёме на архивном диске определите максимальное число пользователей, чьи файлы можно сохранить в архиве, а также максимальный размер имеющегося файла, который может быть сохранён в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

Входные данные

В первой строке входного файла находятся два числа: S — размер свободного места на диске (натуральное число, не превышающее $10\ 000$) и N — количество пользователей (натуральное число, не превышающее 3000). В следующих N строках находятся значения объёмов файлов каждого пользователя (все числа натуральные, не превышающие 100), каждое в отдельной строке.

Выходные данные

Запишите в ответе два числа: сначала наибольшее число пользователей, чьи файлы могут быть помещены в архив, затем максимальный размер имеющегося файла, который может быть сохранён в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

Пример

Пример входного файла

```
100 4
80
30
50
40
```

При таких исходных данных можно сохранить файлы максимум двух пользователей. Возможные объёмы этих двух файлов 30 и 40, 30 и 50 или 40 и 50. Наибольший объём файла из перечисленных пар — 50.

Ответ

```
2 50
```

Подход к решению

Разделим решение на 2 подзадачи - нахождение максимального числа пользователей, файлы которых сможет содержать архивный диск, и нахождение максимального размера файла, при условии хранения максимального количества файлов.

Подсчет файлов

Для нахождения максимального количества хранимых файлов, воспользуемся жадной стратегией.

Подсчет файлов

Для нахождения максимального количества хранимых файлов, воспользуемся жадной стратегией.

Жадная стратегия

Пусть S - оптимальное решение задачи. Тогда файл с наименьшим размером всегда входит в оптимальное решение.

Подсчет файлов

Для нахождения максимального количества хранимых файлов, воспользуемся жадной стратегией.

Жадная стратегия

Пусть S - оптимальное решение задачи. Тогда файл с наименьшим размером всегда входит в оптимальное решение.

Доказательство

Предположим, что S_1 - оптимальное заполнение архивного диска, которое не включает файл минимального размера. Заменим любой файл из S_1 на минимальный. Получим новое заполнение диска $S_2 < S_1$, что противоречит с нашим предположением.

Алгоритм подсчета файлов

Пусть $x_1, x_2, ..., x_N$ - множество файлов пользователей, и S_0 - изначальный объём архивного диска.
Заметим, что сделав жадный шаг и добавив на архивный диск файл i, мы получим новую подзадачу, в которой множество файлов пользователей равно $\{x_1, x_2, ..., x_N\} \setminus x_i$ и объём архивного диска $S_1 = S_0 - x_i$.

Алгоритм подсчета файлов

Пусть $x_1, x_2, ..., x_N$ - множество файлов пользователей, и S_0 - изначальный объём архивного диска.
Заметим, что сделав жадный шаг и добавив на архивный диск файл i, мы получим новую подзадачу, в которой множество файлов пользователей равно $\{x_1, x_2, ..., x_N\} \setminus x_i$ и объём архивного диска $S_1 = S_0 - x_i$.

Алгоритм

1. Отсортируем файлы пользователей по возрастанию.
2. Если множество файлов пусто, или объём архивного диска меньше минимального файла, закончим вычисления.
3. Изымем первый элемент массива файлов и добавим в ответ.
4. Вычтем из объёма архивного диска вес текущего файла.
5. Перейдем на шаг 2.

Вычисление максимального файла

Для вычисления максимального объема файла, при условии хранения максимального количества файлов пользователей, так же воспользуемся жадной стратегией.

Вычисление максимального файла

Для вычисления максимального объема файла, при условии хранения максимального количества файлов пользователей, так же воспользуемся жадной стратегией.

Жадная стратегия

Пусть S_0 - оптимальное решение из N файлов на множестве X, при объеме диска S. Тогда в множестве добавленных файлов присутствует файл максимального размера, который может поместиться в объем $S - \sum Y$, где Y - наименьшие $N - 1$ файлов в множестве X. Назовем такой файл *ключевым*.

Вычисление максимального файла

Для вычисления максимального объема файла, при условии хранения максимального количества файлов пользователей, так же воспользуемся жадной стратегией.

Жадная стратегия

Пусть S_0 - оптимальное решение из N файлов на множестве X, при объеме диска S. Тогда в множестве добавленных файлов присутствует файл максимального размера, который может поместиться в объем $S - \sum Y$, где Y - наименьшие $N - 1$ файлов в множестве X. Назовем такой файл *ключевым*.

Доказательство

Пусть S_1 - оптимальное решение, в которое не входит *ключевой* файл. Заменим $N - 1$ файлов, входящих в решение S_0 на множество Y. Тогда множество $S_1 \setminus Y$ состоит из одного файла из множества $X \setminus Y$, но меньшего *ключевого* файла, что является противоречием с тем, что S_1 является отпимальным решением, т.к. мы можем заменить этот файл на ключевой, получив новое решение из N файлов, но с большим максимальным файлом.

Алгоритм нахождения максимального файла

Прежде всего решим подзадачу нахождения какого-то решения, содержащего оптимальное количество файлов, с помощью алгоритма, рассмотренного ранее, после чего проделаем следующий алгоритм.

Алгоритм нахождения максимального файла

Прежде всего решим подзадачу нахождения какого-то решения, содержащего оптимальное количество файлов, с помощью алгоритма, рассмотренного ранее, после чего проделаем следующий алгоритм.

Алгоритм

1. Изымем из отсортированного массива файлов следующий элемент.
2. Попробуем добавить его в решение вместо максимального файла.
3. Если у нас не получилось, то закончим выполнение алгоритма, иначе вернемся на шаг 1.

Сортировка

Для решения задачи нам потребуется реализовать функцию сортировки массива. Так как мы минимизируем время выполнения задания, а не время работы алгоритма, напишем сортировку пузырьком.

Сортировка

Для решения задачи нам потребуется реализовать функцию сортировки массива. Так как мы минимизируем время выполнения задания, а не время работы алгоритма, напишем сортировку пузырьком.

Принцип работы

Сделаем проход по массиву, попарно сравнивая соседние элементы. Если *больший* элемент находится левее *меньшего*, то обменяем их местами. Не сложно заметить, что после такого прохода по массиву, наибольший элемент окажется в самом конце массива. Пройдясь по массиву $N - 1$ раз, на каждой итерации мы перенесем самый большой элемент в еще не отсортированной части на свое место, в результате чего получим отсортированный по возрастанию массив.

Пример кода на Python

Реализуем алгоритм сортировки пузырьком в виде функции.

```
def BubbleSort(input_list):
    for i in range(len(input_list) - 1):
        is_sorted = True
        for j in range(len(input_list) - 1):
            if input_list[j] > input_list[j + 1]:
                input_list[j], input_list[j + 1] = \
                        input_list[j + 1], input_list[j]
                is_sorted = False
        if is_sorted:
            break
```

Пример кода на Python

Подготовим всё необходимое для решения.

```
# Считаем входные ограничения
S, N = map(int, input().split())

# Ининициализируем нужные переменные
files = [] # Массив файлов пользователей
num_users = 0 # Макс. количество файлов пользователей
max_file = 0 # Максимальный объем файла
i = 0 # Индекс текущего элемента

# Считаем файлы пользователей и добавим в массив
for i in range(N):
    files.append(int(input()))
```

Пример кода на Python

Реализуем алгоритм и вывод ответа

```
BubbleSort(files) # Отсортируем массив файлов

# Заполним диск файлами
while i != len(files) and files[i] <= S:
    S -= files[i]
    max_file = files[i]
    i += 1

# Найдем максимальное количество файлов пользователей
num_users = i

# Найдем размер максимального файла
while i != len(files) and S - max_file + files[i] > 0:
    S -= files[i] - max_file
    max_file = files[i]
    i += 1

print(num_users, max_file) # Выведем ответ
```