ЗМІСТ

[Теоретичні відомості 3](#_Toc82357151)

[Динамічне програмування (доступно) 3](#_Toc82357152)

[Порядок пересчёта 3](#_Toc82357153)

[Многомерная динамика 6](#_Toc82357154)

[Пример №1: Количество СМСок 6](#_Toc82357155)

[Пример №2: Конь 7](#_Toc82357156)

[Динамика и матрица переходов 10](#_Toc82357157)

[Пример №3: Пилообразная последовательность 11](#_Toc82357158)

[Динамика по подотрезкам 14](#_Toc82357159)

[Пример №4: Запаковка строки 14](#_Toc82357160)

[Пример №5: Дубы 16](#_Toc82357161)

[Динамика по поддеревьям 16](#_Toc82357162)

[Пример №6: Логическое дерево 16](#_Toc82357163)

[Динамика по подмножествам 18](#_Toc82357164)

[Динамика по профилю 20](#_Toc82357165)

[Пример №8: Замощение доминошками 21](#_Toc82357166)

[Динамика по изломанному профилю 23](#_Toc82357167)

[Небольшие оптимизации 25](#_Toc82357168)

[Пример №9: Разложение числа 27](#_Toc82357169)

[Динамічне програмування (не дуже доступно, але структуровано) 30](#_Toc82357170)

[Принцип оптимальности на префиксе 31](#_Toc82357171)

[Принцип оптимальности на подотрезках 32](#_Toc82357172)

[Принцип оптимальности на подмножествах 34](#_Toc82357173)

[Вариант 1 34](#_Toc82357174)

[Вариант 2 35](#_Toc82357175)

[Вариант 3 36](#_Toc82357176)

[Вариант 4 37](#_Toc82357177)

[Вариант 5 38](#_Toc82357178)

[Вариант 6 39](#_Toc82357179)

[Вариант 7 39](#_Toc82357180)

[Вариант 8 40](#_Toc82357181)

[Вариант 9 41](#_Toc82357182)

[Вариант 10 42](#_Toc82357183)

[Вариант 11 42](#_Toc82357184)

[Вариант 12 43](#_Toc82357185)

[Вариант 13 44](#_Toc82357186)

[Вариант 14 45](#_Toc82357187)

[Вариант 15 45](#_Toc82357188)

[Вариант 16 46](#_Toc82357189)

[Вариант 17 47](#_Toc82357190)

[Вариант 18 48](#_Toc82357191)

[Вариант 19 48](#_Toc82357192)

[Вариант 20 49](#_Toc82357193)

[Вариант 21 50](#_Toc82357194)

[Вариант 22 51](#_Toc82357195)

[Вариант 23 51](#_Toc82357196)

[Вариант 24 52](#_Toc82357197)

[Вариант 25 52](#_Toc82357198)

[Вариант 26 53](#_Toc82357199)

[Вариант 27 54](#_Toc82357200)

[Вариант 28 54](#_Toc82357201)

[Вариант 29 55](#_Toc82357202)

[Вариант 30 56](#_Toc82357203)

[Вариант 31 57](#_Toc82357204)

[Вариант 32 57](#_Toc82357205)

[Вариант 33 58](#_Toc82357206)

[Вариант 34 59](#_Toc82357207)

[Вариант 35 60](#_Toc82357208)

[Вариант 36 61](#_Toc82357209)

[Вариант 37 61](#_Toc82357210)

[Вариант 38 63](#_Toc82357211)

[Вариант 39 64](#_Toc82357212)

[Вариант 40 65](#_Toc82357213)

[Вариант 41 66](#_Toc82357214)

[Вариант 42 67](#_Toc82357215)

[Вариант 43 68](#_Toc82357216)

[Вариант 44 69](#_Toc82357217)

[Вариант 45 69](#_Toc82357218)

# Теоретичні відомості

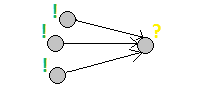
## Динамічне програмування (доступно)

*Динамическое программирование — это когда у нас есть задача, которую непонятно как решать, и мы разбиваем ее на меньшие задачи, которые тоже непонятно как решать. (с) А.Кумок*

Чтобы успешно решить задачу динамикой нужно:

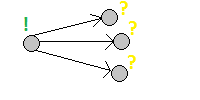
1. Состояние динамики: параметр(ы), однозначно задающие подзадачу.
2. Значения начальных состояний.
3. Переходы между состояниями: формула пересчёта.
4. Порядок пересчёта.
5. Положение ответа на задачу: иногда это сумма или, например, максимум из значений нескольких состояний.

### Порядок пересчёта

Существует три порядка пересчёта:  
1) Прямой порядок:  
Состояния последовательно пересчитывается исходя из уже посчитанных.

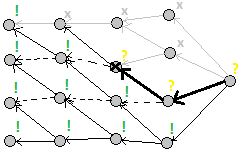
2) Обратный порядок:

Обновляются все состояния, зависящие от текущего состояния.



3) Ленивая динамика:

Рекурсивная [мемоизированная](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%BC%D0%BE%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) функция пересчёта динамики. Это что-то вроде [поиска в глубину](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B8%D1%81%D0%BA_%D0%B2_%D0%B3%D0%BB%D1%83%D0%B1%D0%B8%D0%BD%D1%83) по ацикличному графу состояний, где рёбра — это зависимости между ними.



Элементарный пример: [числа Фибоначчи](http://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%B0_%D0%A4%D0%B8%D0%B1%D0%BE%D0%BD%D0%B0%D1%87%D1%87%D0%B8). Состояние — номер числа.

Прямой порядок:

fib[1] = 1 # Начальные значения

fib[2] = 1 # Начальные значения

**for** i **in** range(3, n + 1):

fib[i] = fib[i - 1] + fib[i - 2] # Пересчёт состояния i

Обратный порядок:

fib[1] = 1 # Начальные значения

**for** i **in** range(1, n):

fib[i + 1] += fib[i] # Обновление состояния i + 1

fib[i + 2] += fib[i] # Обновление состояния i + 2

Ленивая динамика:

**def** **get\_fib**(i):

**if** (i <= 2): # Начальные значения

**return** 1

**if** (fib[i] != -1): # Ленивость

**return** fib[i]

fib[i] = get\_fib(i - 1) + get\_fib(i - 2) # Пересчёт

**return** fib[i]

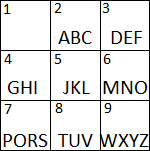
Все три варианта имеют права на жизнь. Каждый из них имеет свою область применения, хотя часто пересекающуюся с другими.

### Многомерная динамика

Пример одномерной динамики приведён выше, в «порядке пересчёта», так что я сразу начну с многомерной. Она отличается от одномерной, как вы уже наверно догадались, количеством измерений, то есть количеством параметров в состоянии. Классификация по этому признаку обычно строится по схеме «один-два-много» и не особо принципиальна, на самом деле.

Многомерная динамика не сильно отличается от одномерной, в чём вы можете убедиться взглянув на пару примеров:

### Пример №1: Количество СМСок

Раньше, когда у телефонов были кнопки, их клавиатуры выглядели примерно так:  
  
  
  
Требуется подсчитать, сколько различных текстовых сообщений множно написать используя не более k нажатий на такой клавиатуре.

**Решение**

1) Состояние динамики: dp[n][m] — количество различных сообщений длины n, использующих m нажатий.  
2) Начальное состояние: есть одно сообщение длины ноль, использующее ноль нажатий — пустое.  
3) Формулы пересчёта: есть по восемь букв, для написания которых нужно одно, два и три нажатия, а так же две буквы требующие 4 нажатия.  
  
Прямой пересчёт:

dp[n][m] = (dp[n - 1][m - 1] + dp[n - 1][m - 2] + dp[n - 1][m - 3]) \* 8 + dp[n - 1][m - 4] \* 2

Обратный пересчёт:

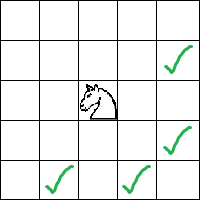
dp[n + 1][m + 1] += dp[n][m] \* 8

dp[n + 1][m + 2] += dp[n][m] \* 8

dp[n + 1][m + 3] += dp[n][m] \* 8

dp[n + 1][m + 4] += dp[n][m] \* 2

### Пример №2: Конь

Шахматный конь стоит в клетке (1, 1) на доске размера N x M. Требуется подсчитать количество способов добраться до клетки (N, M) передвигаясь четырьмя типами шагов:  
  


**Решение**

1) Состояние динамики: dp[i][j] — количество способов добраться до (i, j).

2) Начальное значение: В клетку (1, 1) можно добраться одним способом — ничего не делать.

3) Формула пересчёта:

Для прямого порядка:

dp[i][j] = dp[i - 2][j - 1] + dp[i - 2][j + 1] + dp[i - 1][j - 2] + dp[i + 1][j - 2]

Для обратного порядка:

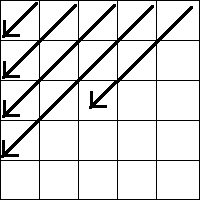
dp[i + 1][j + 2] += dp[i][j]

dp[i + 2][j + 1] += dp[i][j]

dp[i - 1][j + 2] += dp[i][j]

dp[i + 2][j - 1] += dp[i][j]

4) А теперь самое интересное в этой задаче: порядок. Здесь нельзя просто взять и пройтись по строкам или по столбцам. Потому что иначе мы будем обращаться к ещё не пересчитанным состояниям при прямом порядке, и будем брать ещё недоделанные состояния при обратном подходе.  
  
Есть два пути:  
1) Придумать хороший обход.  
2) Запустить ленивую динамику, пусть сама разберётся.  
  
Если лень думать — запускаем ленивую динамику, она отлично справится с задачей.  
Если не лень, то можно придумать обход наподобие такого:

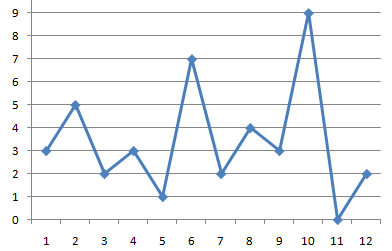


Этот порядок гарантирует обработанность всех требуемых на каждом шаге клеток при прямом обходе, и обработанность текущего состояния при обратном.  
  
5) Ответ просто лежит в dp[n][m].

### Динамика и матрица переходов

Если никогда не умножали матрицы, но хотите понять этот заголовок, то стоит прочитать хотя бы [вики](http://ru.wikipedia.org/wiki/%D0%A3%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86).  
  
Допустим, есть задача, которую мы уже решили динамическим программированием, например, извечные числа Фибоначчи.  
Давайте немного переформулируем её. Пусть у нас есть вектор , из которого мы хотим получить вектор . Чуть-чуть раскроем формулы: . Можно заметить, что из вектора  можно получить вектор  путем умножения на какую-то матрицу, ведь в итоговом векторе фигурируют только сложенные переменные из первого вектора. Эту матрицу легко вывести, вот она: . Назовём её матрицей перехода.  
  
Это значит, что если взять вектор  и умножить его на матрицу перехода n - 1 раз, то получим вектор , в котором лежит fib[n] — ответ на задачу.  
  
А теперь, зачем всё это надо. Умножение матриц обладает свойством ассоциативности, то есть  (но при этом не обладает коммутативностью, что по-моему удивительно). Это свойство даёт нам право сделать так: .  
  
Это хорошо тем, что теперь можно применить [метод быстрого возведения в степень](http://en.wikipedia.org/wiki/Exponentiation_by_squaring), который работает за . Итого мы сумели посчитать N-ое число Фибоначчи за логарифм арифметических операций.  
  
А теперь пример посерьёзнее:

### Пример №3: Пилообразная последовательность

Обозначим пилообразную последовательность длины N как последовательность, у которой для каждого не крайнего элемента выполняется условие: он или меньше обоих своих соседей или больше. Требуется посчитать количество пилообразных последовательностей из цифр длины N. Выглядит это как-то так:  


**Решение**

Для начала решение без матрицы перехода:  
  
1) Состояние динамики: dp[n][last][less] — количество пилообразных последовательностей длины n, заканчивающихся на цифру last. Причём если less == 0, то последняя цифра меньше предпоследней, а если less == 1, значит больше.  
2) Начальные значения:

**for** last **in** range(10):

dp[2][last][0] = 9 - last

dp[2][last][1] = last

3) Пересчёт динамики:

**for** prev **in** range(10):

**if** prev > last:

dp[n][last][0] += dp[n - 1][prev][1]

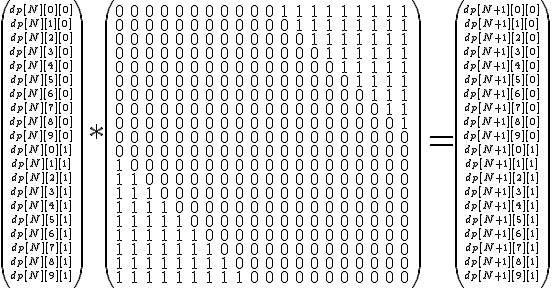
**if** prev < last:

dp[n][last][1] += dp[n - 1][pref][0]

4) Порядок пересчёта: мы всегда обращаемся к предыдущей длине, так что просто пара вложенных for'ов.  
5) Ответ — это сумма dp[N][0..9][0..1].  
  
Теперь надо придумать начальный вектор и матрицу перехода к нему. Вектор, кажется, придумывается быстро: все состояния, обозначающие длину последовательности N. Ну а матрица перехода выводится, смотря на формулы пересчёта.

.

**Вектор и матрица перехода**



### Динамика по подотрезкам

Это класс динамики, в котором состояние — это границы подотрезка какого-нибудь массива. Суть в том, чтобы подсчитать ответы для подзадач, основывающихся на всех возможных подотрезках нашего массива. Обычно перебираются они в порядке увеличения длины, и пересчёт основывается, соответственно на более коротких отрезках.

### Пример №4: Запаковка строки

Вот [Развернутое условие](http://acm.timus.ru/problem.aspx?space=1&num=1238). Я вкратце его перескажу:  
  
Определим сжатую строку:  
1) Строка состоящая только из букв — это сжатая строка. Разжимается она в саму себя.  
2) Строка, являющаяся конкатенацией двух сжатых строк A и B. Разжимается она в конкатенацию разжатых строк A и B.  
3) Строка D(X), где D — целое число, большее 1, а X — сжатая строка. Разжимается она в конкатенацию D строк, разжатых из X.  
Пример: “3(2(A)2(B))C” разжимается в “AABBAABBAABBC”.  
  
Необходимо по строке s узнать длину самой короткой сжатой строки, разжимающийся в неё.

**Решение**

Решается эта задача, как вы уже наверняка догадались, динамикой по подотрезкам.  
  
1) Состояние динамики: d[l][r] — сжатая строка минимальной длины, разжимающаяся в строку s[l:r]  
2) Начальные состояния: все подстроки длины один можно сжать только в них самих.  
3) Пересчёт динамики:  
У лучшего ответа есть какая-то последняя операция сжатия: либо это просто строка из заглавных букв, или это конкатенация двух строк, или само сжатие. Так давайте переберём все варианты и выберем лучший.

dp\_len = r - l

dp[l][r] = dp\_len # Первый вариант сжатия - просто строка.

**for** i **in** range(l + 1, r):

dp[l][r] = min(dp[l][r], dp[l][i] + dp[i][r]) # Попробовать разделить на две сжатые подстроки

**for** cnt **in** range(2, dp\_len):

**if** (dp\_len % cnt == 0): # Если не делится, то нет смысла пытаться разделить

good = True

**for** j **in** range(1, (dp\_len / cnt) + 1): # Проверка на то, что все cnt подстрок одинаковы

good &= s[l:l + dp\_len / cnt] == s[l + (dp\_len / cnt) \* j:l + (dp\_len / cnt) \* (j + 1)]

**if** good: # Попробовать разделить на cnt одинаковых подстрок и сжать

dp[l][r] = min(dp[l][r], len(str(cnt)) + 1 + dp[l][l + dp\_len / cnt] + 1)

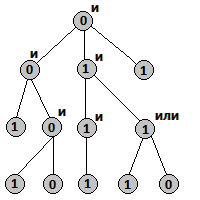
4) Порядок пересчёта: прямой по возрастанию длины подстроки или ленивая динамика.  
5) Ответ лежит в d[0][len(s)].

### Пример №5: [Дубы](http://habrahabr.ru/post/112386)

### Динамика по поддеревьям

Параметром состояния динамики по поддеревьям обычно бывает вершина, обозначающая поддерево, в котором эта вершина — корень. Для получения значения текущего состояния обычно нужно знать результаты всех своих детей. Чаще всего реализуют лениво — просто пишут поиск в глубину из корня дерева.

### Пример №6: Логическое дерево

Дано подвешенное дерево, в листьях которого записаны однобитовые числа — 0 или 1. Во всех внутренних вершинах так же записаны числа, но по следующему правилу: для каждой вершины выбрана одна из логических операций: «И» или «ИЛИ». Если это «И», то значение вершины — это логическое «И» от значений всех её детей. Если же «ИЛИ», то значение вершины — это логическое «ИЛИ» от значений всех её детей.  
  
  
  
Требуется найти минимальное количество изменений логических операций во внутренних вершинах, такое, чтобы изменилось значение в корне или сообщить, что это невозможно.

**Решение**

1) Состояние динамики: d[v][x] — количество операций, требуемых для получения значения x в вершине v. Если это невозможно, то значение состояния — +inf.  
2) Начальные значения: для листьев, очевидно, что своё значение можно получить за ноль изменений, изменить же значение невозможно, то есть возможно, но только за +inf операций.  
3) Формула пересчёта:  
Если в этой вершине уже значение x, то ноль. Если нет, то есть два варианта: изменить в текущей вершине операцию или нет. Для обоих нужно найти оптимальный вариант и выбрать наилучший.  
  
Если операция «И» и нужно получить «0», то ответ это минимум из значений d[i][0], где i — сын v.  
Если операция «И» и нужно получить «1», то ответ это сумма всех значений d[i][1], где i — сын v.  
Если операция «ИЛИ» и нужно получить «0», то ответ это сумма всех значений d[i][0], где i — сын v.  
Если операция «ИЛИ» и нужно получить «1», то ответ это минимум из значений d[i][1], где i — сын v.  
  
4) Порядок пересчёта: легче всего реализуется лениво — в виде поиска в глубину из корня.  
5) Ответ — d[root][value[root] xor 1].

### Динамика по подмножествам

В динамике по подмножествам обычно в состояние входит маска заданного множества. Перебираются чаще всего в порядке увеличения количества единиц в этой маске и пересчитываются, соответственно, из состояний, меньших по включению. Обычно используется ленивая динамика, чтобы специально не думать о порядке обхода, который иногда бывает не совсем тривиальным.

Пример №7: Гамильтонов цикл минимального веса, или задача коммивояжера

Задан взвешенный (веса рёбер неотрицательны) граф G размера N. Найти [гамильтонов цикл](http://ru.wikipedia.org/wiki/%D0%93%D0%B0%D0%BC%D0%B8%D0%BB%D1%8C%D1%82%D0%BE%D0%BD%D0%BE%D0%B2_%D1%86%D0%B8%D0%BA%D0%BB) (цикл, проходящий по всем вершинам без самопересечений) минимального веса.

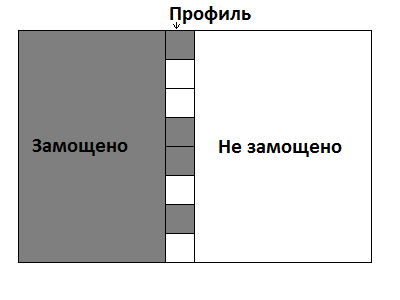
**Решение**

Так как мы ищем цикл, проходящий через все вершины, то можно выбрать за «начальную» вершину любую. Пусть это будет вершина с номером 0.  
  
1) Состояние динамики: dp[mask][v] — путь минимального веса из вершины 0 в вершину v, проходящий по всем вершинам, лежащим в mask и только по ним.  
2) Начальные значения: dp[1][0] = 0, все остальные состояния изначально — +inf.  
3) Формула пересчёта: Если i-й бит в mask равен 1 и есть ребро из i в v, то:

dp[mask][v] = min(dp[mask][v], dp[mask - (1 << i)][i] + w[i][v])

Где w[i][v] — вес ребра из i в v.  
4) Порядок пересчёта: самый простой и удобный способ — это написать ленивую динамику, но можно поизвращаться и написать перебор масок в порядке увеличения количества единичных битов в ней.  
5) Ответ лежит в d[(1 << N) - 1][0].

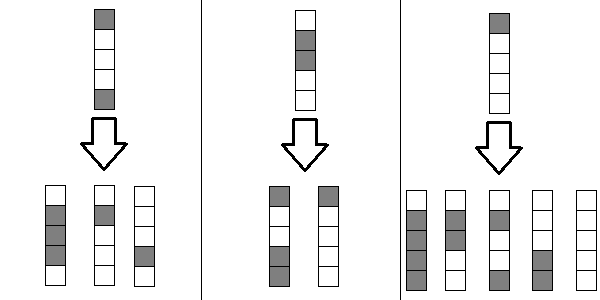
### Динамика по профилю

Классическими задачами, решающимися динамикой по профилю, являются задачи на замощение поля какими-нибудь фигурами. Причём спрашиваться могут разные вещи, например, количество способов замощения или замощение минимальным количеством фигур.  
  
Эти задачи можно решить полным перебором за , где a — количество вариантов замощения одной клетки. Динамика по профилю же оптимизирует время по одной из размерностей до линейной, оставив от себя в экспоненте только коэффициент. Получится что-то такое: .  
  
Профиль — это k (зачастую один) столбцов, являющиеся границей между уже замощённой частью и ещё не замощённой. Эта граница заполнена только частично. Очень часто является частью состояния динамики.  
  
  
  
Почти всегда состояние — это профиль и то, где этот профиль. А переход увеличивает это местоположение на один. Узнать, можно ли перейти из одного профиля в другой можно за линейное от размера профиля время. Это можно проверять каждый раз во время пересчёта, но можно и предподсчитать. Предподсчитывать будем двумерный массив can[mask][next\_mask] — можно ли от одной маски перейти к другой, положив несколько фигурок, увеличив положение профиля на один. Если предподсчитывать, то времени на выполнение потребуется меньше, а памяти — больше.

### Пример №8: Замощение доминошками

Найти количество способов замостить таблицу N x M с помощью доминошек размерами 1 x 2 и 2 x 1.

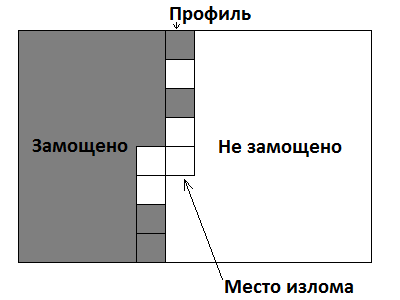
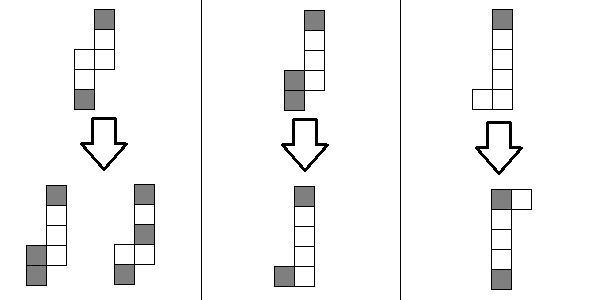
**Решение**

Здесь профиль — это один столбец. Хранить его удобно в виде двоичной маски: 0 — не замощенная клетка столбца, 1 — замощенная. То есть всего профилей .  
  
0) Предподсчёт (опционально): перебрать все пары профилей и проверить, что из одного можно перейти в другой. В этой задаче это проверяется так:  
  
Если в первом профиле на очередном месте стоит 1, значит во втором обязательно должен стоять 0, так как мы не сможем замостить эту клетку никакой фигуркой.  
  
Если в первом профиле на очередном месте стоит 0, то есть два варианта — или во втором 0 или 1.  
Если 0, это значит, что мы обязаны положить вертикальную доминошку, а значит следующую клетку можно рассматривать как 1. Если 1, то мы ставим вертикальную доминошку и переходим к следующей клетке.  
  
Примеры переходов (из верхнего профиля можно перейти в нижние и только в них):  
  
  
  
После этого сохранить всё в массив can[mask][next\_mask] — 1, если можно перейти, 0 — если нельзя.  
1) Состояние динамики: dp[pos][mask] — количество полных замощений первых pos - 1 столбцов с профилем mask.  
2) Начальное состояние: dp[0][0] = 1 — левая граница поля — прямая стенка.  
3) Формула пересчёта:

dp[pos][mask] += dp[pos - 1][next\_mask] \* can[mask][next\_mask]

4) Порядок обхода — в порядке увеличения pos.  
5) Ответ лежит в dp[pos][0].  
  
Полученная асимптотика — .

### Динамика по изломанному профилю

Это очень сильная оптимизация динамики по профилю. Здесь профиль — это не только маска, но ещё и место излома. Выглядит это так:  
  
  
  
Теперь, после добавления излома в профиль, можно переходить к следующему состоянию, добавляя всего одну фигурку, накрывающую левую клетку излома. То есть увеличением числа состояний в N раз (надо помнить, где место излома) мы сократили число переходов из одного состояния в другое с  до . Асимптотика улучшилась с  до .  
  
Переходы в динамике по изломанному профилю на примере задачи про замощение доминошками (пример №8):  
  


Восстановление ответа

Иногда бывает, что просто знать какую-то характеристику лучшего ответа недостаточно. Например, в задаче «Запаковка строки» (пример №4) мы в итоге получаем только длину самой короткой сжатой строки, но, скорее всего, нам нужна не её длина, а сама строка. В таком случае надо восстановить ответ.  
  
В каждой задаче свой способ восстановления ответа, но самые распространенные:

* Рядом со значением состояния динамики хранить полный ответ на подзадачу. Если ответ — это что-то большое, то может понадобиться чересчур много памяти, поэтому если можно воспользоваться другим методом, обычно так и делают.
* Восстанавливать ответ, зная предка(ов) данного состояния. Зачастую можно восстановить ответ, зная только как он был получен. В той самой «Запаковке строки» можно для восстановления ответа хранить только вид последнего действия и то, из каких состояний оно было получено.
* Есть способ, вообще не использующий дополнительную память — после пересчёта динамики пойти с конца по лучшему пути и по дороге составлять ответ.

### Небольшие оптимизации

Память

Зачастую в динамике можно встретить задачу, в которой состояние требует быть посчитанными не очень большое количество других состояний. Например, при подсчёте чисел Фибоначчи мы используем только два последних, а к предыдущим уже никогда не обратимся. Значит, можно про них забыть, то есть не хранить в памяти. Иногда это улучшает асимптотическую оценку по памяти. Этим приёмом можно воспользоваться в примерах №1, №2, №3 (в решении без матрицы перехода), №7 и №8. Правда, этим никак не получится воспользоваться, если порядок обхода — ленивая динамика.

Время

Иногда бывает так, что можно улучшить асимптотическое время, используя какую-нибудь структуру данных. К примеру, в [алгоритме Дейкстры](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B) можно воспользоваться [очередью с приоритетами](http://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%D1%81_%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BE%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) для изменения асимптотического времени.

Замена состояния

В решениях динамикой обязательно фигурирует состояние — параметры, однозначно задающие подзадачу, но это состояние не обязательно одно единственное. Иногда можно придумать другие параметры и получить с этого выгоду в виде снижения асимптотического времени или памяти.

### Пример №9: Разложение числа

Требуется найти количество разложений числа N на различные слагаемые. Например, если N = 7, то таких разложений 5:

* 7
* 3 + 4
* 2 + 5
* 1 + 7
* 1 + 2 + 4

**Два решения с различными состояниями**

Решение №1:

1) Состояние динамики: dp[n][k] — количество разложений числа n на числа, меньшие или равные k. Параметр k нужен, чтобы брать всегда только большие числа, чем уже имеющиеся.  
2) Начальные значения: dp[1][1] = 1, dp[1][i] = 0.  
3) Формула пересчёта:

**for** last\_summand **in** range(1, k + 1):

dp[n][k] += dp[n - last\_summand][last\_summand]

4) Порядок: прямой, в порядке увеличения n.  
5) Ответ — сумма dp[N][1..N].  
  
Состояний: , переходов: . Асимптотика: .

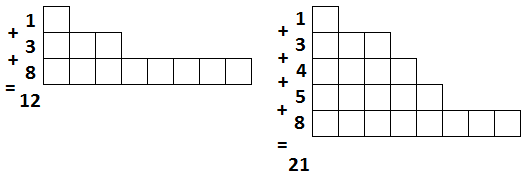
Решение №2:

1) Поменяем состояние. Теперь dp[n][k] — это количество разложений числа n на k различных чисел. Казалось бы незачем, но сейчас будет понятно.  
2) Начальные значения: dp[1][1] = 1, dp[1][i] = 0.  
3) Формула пересчёта:

dp[n][k] = dp[n - k][k] + dp[n - k][k - 1]

Теперь надо пояснить, что значит эта формула. Все состояния можно получить (причём единственным способом) делая поочерёдно два действия:

* Все уже имеющиеся числа увеличить на 1.
* Все уже имеющиеся числа увеличить на 1. Добавить число 1 в разложение.

Чтобы понять, почему это так можно посмотреть на [диаграммы Юнга](http://en.wikipedia.org/wiki/Young_tableau#Diagrams):  
  
  
Строки здесь обозначают слагаемые.  
  
Первое решение последовательно добавляет по одной строчке внизу таблицы, а второе — по одному столбцу слева таблицы. Вариантов размера следующей строчки много — главное, чтобы она была больше предыдущей, а столбцов — только два: такой же как предыдущий и на единичку больше.  
  
4) Порядок пересчёта: прямой, в порядке увеличения n.  
  
Невооруженным взглядом кажется, что у этого решения асимптотика , ведь есть два измерения в состоянии и  переходов. Но это не так, ведь второй параметр — k ограничен сверху не числом N, а формулой  (сумма чисел от 1 до k меньше или равна разлагаемого числа). Из этой формулы видно, что количество состояний .  
  
5) Ответ — это сумма dp[N][1..k\_max].

## Динамічне програмування (не дуже доступно, але структуровано)

В процессе составления алгоритмов динамического программирования, требуется следовать последовательности из четырёх действий:

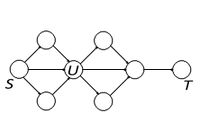
1. Описать структуру оптимального решения.
2. Рекурсивно определить значение оптимального решения.
3. Вычислить значение оптимального решения с помощью метода восходящего анализа.
4. Составить оптимальное решение на основе полученной информации.

Задача имеет оптимальную подструктуру, если её оптимальное решение может быть рационально составлено из оптимальных решений её подзадач.

Наличие оптимальной подструктуры в задаче используется для определения применимости динамического программирования и жадных алгоритмов для решения оной. Например, задача по нахождению кратчайшего пути между некоторыми вершинами графа содержит в себе оптимальное решение подзадач.

Многие задачи, решаемые динамическим программированием, можно определить как поиск в заданном ориентированном ациклическом графе [кратчайшего пути](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9A%D1%80%D0%B0%D1%82%D1%87%D0%B0%D0%B9%D1%88%D0%B8%D0%B9_%D0%BF%D1%83%D1%82%D1%8C_%D0%B2_%D0%B0%D1%86%D0%B8%D0%BA%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%BC_%D0%B3%D1%80%D0%B0%D1%84%D0%B5) от одной вершины к другой.

### Принцип оптимальности на префиксе

[](https://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%B0%D0%B9%D0%BB:ST.jpg)

Рассмотрим некий необратимый процесс производства и представим его в виде ориентированного и ациклического графа. Процесс проходит некий ряд состояний. Началом производства (первым состоянием) обозначим вершину графа SS, а конец производства (последнее состояние) TT. Процесс требует оптимизации, т.е. требуется найти оптимальный путь S⇝TS⇝T. Он проходит через вершину графа UU. Префикс оптимального пути S⇝US⇝U является оптимальным путём S⇝US⇝U. Теперь рассмотрим принцип оптимальности для динамического программирования на префиксе. Итак, имеем некоторый оптимальный путь S⇝TS⇝T, который проходит через UU. Пусть префикс ΔUΔU, т.е. путь от S⇝US⇝U, неоптимален. Тогда заменим неоптимальную часть S⇝US⇝U пути S⇝TS⇝T оптимальной, а путь U⇝TU⇝T добавим в конец. Получим оптимальный путь S⇝TS⇝T. Принцип оптимальности для подзадач выполняется. Т.е. чтобы получить оптимальный путь из одной вершины графа в другую, префиксы меньших путей должны быть оптимальными.

В качестве примера рассмотрим следующую задачу: пусть дан ациклический ориентированный взвешенный граф, требуется найти вес кратчайшего пути из u в v. Воспользуемся принципом оптимальности на префиксе.  
Пусть dd — функция, где d(i)d(i) — вес кратчайшего пути из uu в ii. Ясно, что d(u)d(u) равен 00. Пусть w(i,j)w(i,j) — вес ребра из ii в jj. Будем обходить граф в порядке [топологической сортировки](https://neerc.ifmo.ru/wiki/index.php?title=%D0%98%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BE%D0%B1%D1%85%D0%BE%D0%B4%D0%B0_%D0%B2_%D0%B3%D0%BB%D1%83%D0%B1%D0%B8%D0%BD%D1%83_%D0%B4%D0%BB%D1%8F_%D1%82%D0%BE%D0%BF%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B9_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8). Получаем следующие соотношения:

d(i)=minj:j⇝i(d(j)+w(j,i))d(i)=minj:j⇝i(d(j)+w(j,i))

Так как мы обходим граф в порядке [топологической сортировки](https://neerc.ifmo.ru/wiki/index.php?title=%D0%98%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BE%D0%B1%D1%85%D0%BE%D0%B4%D0%B0_%D0%B2_%D0%B3%D0%BB%D1%83%D0%B1%D0%B8%D0%BD%D1%83_%D0%B4%D0%BB%D1%8F_%D1%82%D0%BE%D0%BF%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B9_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8), то на ii-ом шаге всем d(j)d(j) (jj такие, что существует ребро из jj в ii) уже присвоены оптимальные ответы, и, следовательно, d(i)d(i) также будет присвоен оптимальный ответ.

**Примеры задач**

* [Кратчайший путь в ациклическом графе](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9A%D1%80%D0%B0%D1%82%D1%87%D0%B0%D0%B9%D1%88%D0%B8%D0%B9_%D0%BF%D1%83%D1%82%D1%8C_%D0%B2_%D0%B0%D1%86%D0%B8%D0%BA%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%BC_%D0%B3%D1%80%D0%B0%D1%84%D0%B5)
* [Задача о числе путей в ациклическом графе](https://neerc.ifmo.ru/wiki/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D1%87%D0%B8%D1%81%D0%BB%D0%B5_%D0%BF%D1%83%D1%82%D0%B5%D0%B9_%D0%B2_%D0%B0%D1%86%D0%B8%D0%BA%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%BC_%D0%B3%D1%80%D0%B0%D1%84%D0%B5)

### Принцип оптимальности на подотрезках

Требуется посчитать функцию f(1,n)f(1,n). Принцип состоит в следующем: пусть для всех отрезков ii, jj (где u⩽i⩽j⩽vu⩽i⩽j⩽v) известен оптимальный ответ для функции f(i,j)f(i,j). Тогда мы будем вычислять f(u,v)f(u,v) через такие f(i,j)f(i,j). В качестве примера рассмотрим следующую классическую задачу: дана строка длины n, нужно найти максимальный подпалиндром (подпоследовательность максимальной длины, которая является палиндромом). Пусть d(i,j)d(i,j) - ответ на задачу для подстроки, начинающаяся с символа ii и заканчивающаяся в символе jj. Ясно, что d(i,j)=0d(i,j)=0 для всех i,j,i,j, таких что i>ji>j и d(i,i)=1d(i,i)=1 для всех ii. Пусть нам нужно посчитать значение для d(i,j)d(i,j), причем значение dd для всех l,rl,r, таких что i⩽l⩽r⩽ji⩽l⩽r⩽j уже посчитаны и они оптимальны. Рассмотрим два случая:

1. s(i)≠s(j)s(i)≠s(j), тогда d(i,j)=max(d(i,j−1),d(i+1,j))d(i,j)=max(d(i,j−1),d(i+1,j))
2. s(i)=s(j)s(i)=s(j), тогда d(i,j)=d(i+1,j−1)+2d(i,j)=d(i+1,j−1)+2

Доказательство:

1. Так s(i)≠s(j)s(i)≠s(j), символы s(i)s(i) и s(j)s(j) не могут входить в максимальный подпалиндром одновременно, то есть либо s(i)s(i) входят в максимальный подпалиндром(тогда его длина d[i,j−1]d[i,j−1]), либо s(j)s(j) входит в максимальный подпалиндром (тогда его длина d[i+1,j]d[i+1,j]), либо оба не входят в максимальный подпалиндром (тогда его длина =d[i,j−1]=d[i+1,j]=d[i,j−1]=d[i+1,j]).
2. Данное равенство следует из факта, что выгодно включить в максимальный подпалиндром символы s(i)s(i) и s(j)s(j).

**Примеры задач**

* [Задача о расстановке знаков в выражении](https://neerc.ifmo.ru/wiki/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D1%80%D0%B0%D1%81%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B5_%D0%B7%D0%BD%D0%B0%D0%BA%D0%BE%D0%B2_%D0%B2_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B8)
* [Задача о порядке перемножения матриц](https://neerc.ifmo.ru/wiki/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BA%D0%B5_%D0%BF%D0%B5%D1%80%D0%B5%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86)
* [Задача о выводе в контекстно-свободной грамматике, алгоритм Кока-Янгера-Касами](https://neerc.ifmo.ru/wiki/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4%D0%B5_%D0%B2_%D0%BA%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BD%D0%BE-%D1%81%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B9_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B5,_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9A%D0%BE%D0%BA%D0%B0-%D0%AF%D0%BD%D0%B3%D0%B5%D1%80%D0%B0-%D0%9A%D0%B0%D1%81%D0%B0%D0%BC%D0%B8)
* [Задача об оптимальном префиксном коде с сохранением порядка. Монотонность точки разреза](https://neerc.ifmo.ru/wiki/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE%D0%B1_%D0%BE%D0%BF%D1%82%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC_%D0%BF%D1%80%D0%B5%D1%84%D0%B8%D0%BA%D1%81%D0%BD%D0%BE%D0%BC_%D0%BA%D0%BE%D0%B4%D0%B5_%D1%81_%D1%81%D0%BE%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%D0%BC_%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BA%D0%B0._%D0%9C%D0%BE%D0%BD%D0%BE%D1%82%D0%BE%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D1%82%D0%BE%D1%87%D0%BA%D0%B8_%D1%80%D0%B0%D0%B7%D1%80%D0%B5%D0%B7%D0%B0)
* [Задача о наибольшей общей подпоследовательности](https://neerc.ifmo.ru/wiki/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%BD%D0%B0%D0%B8%D0%B1%D0%BE%D0%BB%D1%8C%D1%88%D0%B5%D0%B9_%D0%BE%D0%B1%D1%89%D0%B5%D0%B9_%D0%BF%D0%BE%D0%B4%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8)
* [Задача о редакционном расстоянии, алгоритм Вагнера-Фишера](https://neerc.ifmo.ru/wiki/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D1%80%D0%B5%D0%B4%D0%B0%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%BC_%D1%80%D0%B0%D1%81%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B8,_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%92%D0%B0%D0%B3%D0%BD%D0%B5%D1%80%D0%B0-%D0%A4%D0%B8%D1%88%D0%B5%D1%80%D0%B0)
* [Задача о расстоянии Дамерау-Левенштейна](https://neerc.ifmo.ru/wiki/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D1%80%D0%B0%D1%81%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B8_%D0%94%D0%B0%D0%BC%D0%B5%D1%80%D0%B0%D1%83-%D0%9B%D0%B5%D0%B2%D0%B5%D0%BD%D1%88%D1%82%D0%B5%D0%B9%D0%BD%D0%B0)

### Принцип оптимальности на подмножествах

Требуется посчитать функцию f(A)f(A), где AA — некоторое множество. Принцип состоит в следующем: пусть для всех множеств BB (где B∈AB∈A) известен оптимальный ответ для функции f(B)f(B). Тогда будем вычислять f(A)f(A) через такие f(B)f(B). В качестве примера рассмотрим задачу о коммивояжере.

Обозначим d[i][mask]d[i][mask] как наименьшую стоимость пути из вершины ii в вершину 00, проходящую (не считая вершины ii) единожды по всем тем и только тем вершинам jj, для которых maskj=1maskj=1 (т.е. d[i][mask]d[i][mask] уже найденный оптимальный путь от ii-ой вершины до 00-ой, проходящий через те вершины, где maskj=1maskj=1. Если maskj=0maskj=0,то эти вершины еще не посещены). Тогда воспользуемся принципом оптимальности на подмножествах. Стоимостью минимального гамильтонова цикла в исходном графе будет значение d[0][2n−1]d[0][2n−1] — стоимость пути из 00-й вершины в 00-ю, при необходимости посетить все вершины.

**Примеры задач**

* [Задача коммивояжера, ДП по подмножествам](https://neerc.ifmo.ru/wiki/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BC%D0%B8%D0%B2%D0%BE%D1%8F%D0%B6%D0%B5%D1%80%D0%B0,_%D0%94%D0%9F_%D0%BF%D0%BE_%D0%BF%D0%BE%D0%B4%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B0%D0%BC)

# Вариант 1

2127 год. Прошло уже много лет с тех пор, как состоялась первая олимпиада по информатике. Как и многие другие соревнования, наши олимпиады теперь проводятся в несколько дней. Теперь даже задача выбора подходящего времени для олимпиады представляет определенные трудности. Ведь на разных планетах, используются разные способы отсчета времени: длина месяца, количество дней в неделе и те дни, по которым невозможно проведение олимпиады, могут различаться. Возникла необходимость написания программы, которая поможет решить эту задачу. И тогда в жюри вспомнят, что уже сейчас мы предвидели такую ситуацию и предложили вам решить подобную задачу.

В качестве первого шага найдите количество способов выбрать время проведения олимпиады.

В первой строке входного файла INPUT.TXT содержатся два целых числа n и k (1 ≤ k ≤ n ≤ 100000) - количество дней месяца и продолжительность олимпиады соответственно. Во второй строке задаются количество дней в неделе w, количество дней, запрещенных еженедельно, dw и день недели, на который приходится первый день месяца s (1 ≤ s ≤ w ≤ n, 0 ≤ dw ≤ w). Третья строка содержит dw номеров дней недели (например, выходных), в которые нельзя проводить олимпиаду. В четвертой строке записано количество дней месяца dm, не подходящих для проведения олимпиады по причинам отличным от еженедельного распорядка (например, такими днями являются государственные праздники). Последняя строка содержит dm целых чисел - номера этих дней. Дни месяца так же нумеруются начиная с 1. Заметим, что некоторые дни могут быть запрещенными сразу по обеим причинам.

В выходной файл OUTPUT.TXT выведите единственное целое число - количество способов выбрать k подряд идущих дней, в которые возможно проведение олимпиады.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 31 3 7 1 7 7 2 1 9 | 15 |

# Вариант 2

Для проведения эксперимента надо выбрать из N имеющихся приборов только три. Для этого выполняют следующую операцию - если в группе приборов больше трех, то их нумеруют и выбирают одну из групп: с четными или нечетными номерами. Операцию повторяют до тех пор, пока в группе не останется три или менее приборов. Если их остается ровно три, то они и берутся для эксперимента.

Требуется написать программу, которая подсчитает количество способов такого выбора приборов.

В единственной строке входного файла INPUT.TXT записано число N (1 ≤ N ≤ 2147483647).

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно число - найденное количество способов выбора приборов.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 | 1 |
| 2 | 6 | 2 |

# Вариант 3

- Ты врешь, Коля! На Марсе жизни нет! Кто тебе такую чушь сказал?

- Петя. А ему сказал Саша.

- Да от Пети я в жизни правдивого слова не слышал! Ему что ни скажут, он все переврет. Да и Саше откуда знать?

- А Саше рассказал про это Владимир Алексеевич, наш учитель биологии.

- Ну, Владимиру Алексеевичу-то можно верить... Только вряд ли он так сказал, это либо Саша, либо Петя придумал. А может, это ты меня разыгрываешь?..

- Минуточку, ребята, - вмешался подошедший к спорящим учитель математики, Глеб Тимофеевич, - давайте подойдем к проблеме формально. Допустим, что все диалоги - Владимира Алексеевича с Сашей, Саши с Петей и Пети с Колей - действительно имели место. Пронумеруем ребят числами 1, 2 и 3. Предположим также, что каждый из ребят независимо друг от друга передал полученную информацию относительно жизни на Марсе верно с вероятностью pi, а соврал с вероятностью qi = 1 - pi для i = 1, 2, 3. Вероятности – это вещественные числа от нуля до единицы; событие, имеющее вероятность 0, никогда не произойдет, событие же с вероятностью 1 произойдет без всякого сомнения. Зная, что Коля после этого объявил, что жизнь на Марсе все-таки есть, найдите по данным pi вероятность того, что так действительно сказал Владимир Алексеевич.

- А как искать эту вероятность? И что значит независимо друг от друга? – растерялись ребята.

- Независимость означает, что действие одного из ребят никак не отражается на том, как поступят другие. К примеру, Пете неважно, соврал ли Саша - в любом случае он передаст сказанное Сашей правильно с вероятностью ровно p2. Задача несложная, и можно рассмотреть все восемь возможных случаев. Первый случай - все ребята говорили правду, и вероятность этого случая равна p1∙p2∙p3. В этом случае жизнь на Марсе, без сомнения, есть - Владимиру Алексеевичу мы верим, а ребята передали его слова правильно. Второй случай, когда соврал только Саша, имеет место с вероятностью q1∙p2∙p3, и в этом случае жизни на Марсе нет. Далее переберем остальные шесть случаев, каждый раз перемножая соответствующие вероятности, а потом просуммируем вероятности тех случаев, в которых слова учителя переданы правильно. То, что вероятности для отдельных ребят в каждом случае надо перемножить - это и есть формальное определение независимости. Ну, в скольких случаях будет передано именно то, что говорил Владимир Алексеевич?

- В одном …

- А вот и нет. Например, если Петя и Коля соврали, а Саша сказал правду, то истина, дважды исказившись, дойдет до нас в неизменном виде. И вообще, четное количество отрицаний, примененных к утверждению, дает само утверждение. В нашей задаче случаев с четным количеством отрицаний - четыре, и итоговая вероятность равна p1∙p2∙p3+q1∙q2∙p3+q1∙p2∙q3+p1∙q2∙q3.

- То есть если Петя и Коля точно соврут, а Саша точно скажет правду, то от Коли мы услышим в точности то, что говорил учитель?

- Совершенно верно. А теперь решите-ка задачу для общего случая, когда ребят не трое, а n. Первому, кто решит - пятерка на следующей контрольной!

Входной файл INPUT.TXT содержит целое число n (1 ≤ n ≤ 100). Во второй строке через пробел записаны n вещественных чисел - это числа p1, p2, ..., pn (0 ≤ pi ≤ 1). Числа даны с не более чем шестью десятичными знаками после запятой.

В выходной файл OUTPUT.TXT выведите одно вещественное число, округленное до шести знаков после запятой - вероятность существования жизни на Марсе.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 1 0.1 0.9 | 0.18 |

# Вариант 4

Вова стоит перед лесенкой из N ступеней. На каждой из ступеней написаны произвольные целые числа. Первым шагом Вова может перейти на первую ступень или, перепрыгнув через первую, сразу оказаться на второй. Также он поступает и дальше, пока не достигнет N-ой ступени. Посчитаем сумму всех чисел, написанных на ступенях через которые прошел Вова.

Требуется написать программу, которая определит оптимальный маршрут Вовы, при котором, шагая, он получит наибольшую сумму.

Входной файл INPUT.TXT содержит в первой строке натуральное число N – количество ступеней лестницы (2 ≤ N ≤ 1000). Во второй строке через пробел заданы числа, написанные на ступенях лестницы, начиная с первой. Числа, написанные на ступенях, не превосходят по модулю 1000.

Выходной файл OUTPUT.TXT должен содержать в первой строке наибольшее значение суммы. Во второй строке должны быть записаны через пробел номера ступеней по возрастанию, по которым должен шагать Вова. Если существует несколько различных правильных маршрутов, то можно вывести любой из них.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 1 2 1 | 4 1 2 3 |
| 2 | 3 1 -1 1 | 2 1 3 |

# Вариант 5

Найдите n-й элемент строго возрастающей последовательности, которая описывается следующими правилами:

1. число 1 является элементом последовательности;
2. если a – элемент последовательности, то 2a, 3a, 5a тоже являются элементами последовательности;
3. последовательности принадлежат только элементы, заданные правилами 1 и 2.

Входной файл INPUT.TXT содержит одно натуральное число n (n ≤ 10000).

В выходной файл OUTPUT.TXT выведите одно число – n-й элемент последовательности.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 2 | 2 |
| 2 | 9 | 10 |

# Вариант 6

Цепочкой слов длины N назовем последовательность слов W1, W2, ..., WN такую, что для 1 ≤ i < N слово Wi является собственным префиксом слова Wi+1.

Напомним, что слово U длины K называется собственным префиксом слова V длины L, если K < L и первые K букв слова V совпадают со словом U.

Задан набор слов S = {S1, S2, ..., SM}. Найдите максимальную длину цепочки слов, которую можно построить, используя (возможно, не все) слова этого набора.

Входной файл INPUT.TXT содержит целое число M (1 ≤ M ≤ 255). Каждая из последующих M строк содержит по одному слову из набора S. Все слова не пусты, имеют длину, не превосходящую 255 символов, и состоят только из строчных букв английского алфавита.

В выходной файл OUTPUT.TXT выведите ответ на задачу.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 a ab abc | 3 |
| 2 | 5 a ab bc bcd add | 2 |

# Вариант 7

Карта лабиринта представляет собой квадратное поле размером N×N. Некоторые квадраты этого поля запрещены для прохождения. Шаг в лабиринте – перемещение из одной разрешенной клетки к другой разрешенной клетке, смежной с первой по стороне. Путь – это некоторая последовательность таких шагов. При этом каждую клетку, включая начальную и конечную, можно посещать несколько раз.

Требуется написать программу, которая подсчитает количество различных путей из клетки (1, 1) в клетку (N, N) ровно за K шагов (то есть оказаться в клетке (N, N) после K-го шага).

Входной файл INPUT.TXT содержит в первой строке числа N и K, разделенные пробелом (1 < N ≤ 15, 0 < K ≤ 30). Следующие N строк, по N символов в каждой, содержат карту лабиринта, начиная с клетки (1, 1). Символ «0» означает не запрещенную для прохождения клетку, а символ «1» - запрещенную. Начальная и конечная клетки всегда разрешены для прохождения.

Выходной файл OUTPUT.TXT должен содержать количество возможных различных путей длины K. Во всех тестах это значение не будет превышать 2147483647.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 6 000 101 100 | 5 |
| 2 | 2 8 01 10 | 0 |

# Вариант 8

Вы можете вспомнить хоть одного своего знакомого до двадцатилетнего возраста, который в детстве не играл в компьютерные игры? Если да, то может быть вы и сами не знакомы с этим развлечением? Впрочем, трудностей при решении этой задачи это создать не должно.

Во многих старых играх с двумерной графикой можно столкнуться с подобной ситуацией. Какой-нибудь герой прыгает по платформам (или островкам), которые висят в воздухе. Он должен перебраться от одного края экрана до другого. При этом при прыжке с одной платформы на соседнюю, у героя уходит |y2-y1| единиц энергии, где y1 и y2 – высоты, на которых расположены эти платформы. Кроме того, у героя есть суперприем, который позволяет перескочить через платформу, но на это затрачивается 3\*|y3-y1| единиц энергии. Конечно же, энергию следует расходовать максимально экономно.

Предположим, что вам известны координаты всех платформ в порядке от левого края до правого. Сможете ли вы найти, какое минимальное количество энергии потребуется герою, чтобы добраться с первой платформы до последней?

В первой строке входного файла INPUT.TXT записано количество платформ n (1 ≤ n ≤ 30000). Вторая строка содержит n натуральных чисел, не превосходящих 30000 – высоты, на которых располагаются платформы.

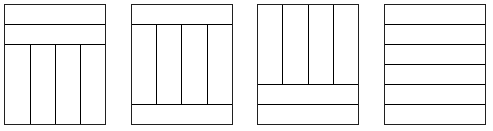
В выходной файл OUTPUT.TXT запишите единственное число – минимальное количество энергии, которую должен потратить игрок на преодоление платформ (конечно же в предположении, что cheat-коды использовать нельзя).

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 1 5 10 | 9 |
| 2 | 3 1 5 2 | 3 |

# Вариант 9

Прямоугольный коридор длиной N метров и шириной M метров решили застелить N прямоугольными плитками шириной 1 метр и длиной M метров, таким образом, чтобы не было не застеленной поверхности.

Требуется написать программу, которая найдет количество способов это сделать. Например, для коридора с размерами 6 на 4 существует четыре способа застелить плитками 1 на 4.



Входной файл INPUT.TXT содержит два целых числа – M (длина плитки и ширина коридора) и N (длина коридора). Для этих чисел верны неравенства 2 ≤ M ≤ N ≤ 50.

Выходной файл OUTPUT.TXT должен содержать одно число – количество способов.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 4 6 | 4 |
| 2 | 2 2 | 2 |

# Вариант 10

Дана числовая последовательность, требуется найти длину наибольшей возрастающей подпоследовательности.

В первой строке входного файла INPUT.TXT записано число N - длина последовательности (1 ≤ N ≤ 1000). Во второй строке записана сама последовательность (через пробел). Числа последовательности - целые числа, не превосходящие 10000 по модулю.

В выходной файл OUTPUT.TXT требуется вывести наибольшую длину возрастающей подпоследовательности.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 6 3 29 5 5 28 6 | 3 |

# Вариант 11

В таблице из N строк и N столбцов клетки заполнены цифрами от 0 до 9. Требуется найти такой путь из клетки (1, 1) в клетку (N, N), чтобы сумма цифр в клетках, через которые он пролегает, была минимальной; из любой клетки ходить можно только вниз или вправо.

В первой строке входного файла INPUT.TXT находится число N. В следующих N строках содержатся по N цифр без пробелов. (2 ≤ N ≤ 250)

В выходной файл OUTPUT.TXT выведите N строк по N символов. Символ «#» (решетка) показывает, что маршрут проходит через эту клетку, а «.» (точка) - что не проходит. Если путей с минимальной суммой цифр несколько, можно вывести любой.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 943 216 091 | #.. ### ..# |

# Вариант 12

Многие, вероятно, слышали песни про приключения лягушонка Crazy Frog. На этот раз неугомонное милое создание решило подкрепиться, но даже такое простое действие решило выполнить в виде игры. Итак, в каждой клетке квадратного игрового поля, разбитого на N×N (N ≤ 50) клеток, находится один комар весом aij (вес комара – натуральное число ≤ 50), i - номер строки, j - номер столбца. Лягушонок, прыгая с клетки на клетку, ест комаров. Правила игры таковы - в каждом столбце можно съесть не более одного комара. Всякий раз при съедании комара запоминаем номер строки, откуда съеден комар, и сумма номеров строк, в которых были съедены комары, в конце игры должна быть в точности равна N. Учтите, если из какой-то строки съедено несколько комаров, то номер данной строки участвует в суммировании более одного раза.

Определите максимальный вес комаров, который можно съесть при следовании приведённым правилам.

Первая строка входного файла INPUT.TXT содержит число N. Следующие N строк содержат по N чисел aij, разделенных пробелами.

В выходной файл OUTPUT.TXT выведите целое число – вес съеденных комаров.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 8 2 1 1 2 6 2 7 2 | 14 |
| 2 | 5 8 2 1 2 3 1 2 6 2 4 2 7 2 3 4 1 3 2 4 4 1 3 4 3 1 | 19 |

# Вариант 13

Энты были созданы в Первоначальную эпоху вместе с другими обитателями Средиземья. Эльфийские легенды гласят, что когда Варда зажгла звёзды и пробудились Эльфы, вместе с ними пробудились и Энты в Великих Лесах Арды.

Когда Энты пришли в Арду, они ещё не умели говорить — этому искусству их обучали Эльфы, и Энтам это ужасно нравилось. Им доставляло удовольствие изучать разные языки, даже щебетание Людей.

Эльфы выработали хорошую технику обучения энтов своему языку. Первый энт, которого обучили эльфы, выучил всего два слова — «tancave» (да) и «la» (нет). Обученный энт выбрал одного старого и одного молодого энта, не умеющих говорить, и обучил их всем словам, которые знал сам. Затем обучение этих двух энтов продолжили сами эльфы. Каждый обучившийся у эльфов энт снова выбирал из неговорящих сородичей одного старого и одного молодого, обучал их всем словам, которые знал, передавал эльфам и так далее.

Выяснилось, что более молодые энты выучивали у эльфов ещё ровно столько же слов, сколько они узнали от обучавшего их энта. А вот более старые, уже склонные к одеревенению энты, пополняли свой запас всего лишь одним словом. После обучения у эльфов энты до конца света уже не могли выучить ни одного нового слова.

Общее число энтов в Средиземье больше, чем вы думаете. Интересно, а сколько из них знают ровно 150 квенийских слов? Похожую задачу вам предстоит решить.

Входной файл INPUT.TXT содержит натуральные числа K и P (K ≤ 106; 1 ≤ P ≤ 109), записанные через пробел.

Мы понимаем, что число энтов, знающих в точности K слов, может быть слишком велико, поэтому просим вывести в выходной файл OUTPUT.TXT лишь количество энтов, знающих ровно K слов, по модулю P.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 4 10 | 2 |
| 2 | 8 10 | 5 |
| 3 | 360 1000 | 179 |

# Вариант 14

На прямой ветке железной дороги расположено несколько станций. Задана стоимость проезда между любыми двумя станциями.

Требуется написать программу нахождения минимальной стоимости проезда между крайними станциями. Двигаться по железной дороге можно только в одном направлении (от станции с меньшим номером до станции с большим номером.).

Входной файл INPUT.TXT содержит в первой строке натуральное число N, не большее 250. Всего на дороге расположено N+1 станций, пронумерованных от 0 до N. В следующих строках записано N(N+1)/2 чисел, задающих стоимости проезда между станциями: сначала стоимость проезда от станции 0 до станций 1, 2, 3, …, N, затем от станции 1 до станций 2, 3, …, N, …, от станции N-1 до станции N. Все стоимости проезда – неотрицательные целые числа, не превосходящие 10000.

Выходной файл OUTPUT.TXT должен содержать одно число – минимальную стоимость проезда от станции 0 до станции N с возможными пересадками.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 7 10 20 4 8 2 | 12 |

В приведенном примере всего 4 станции с номерами 0, 1, 2, 3. Оптимальный маршрут проходит через станции 0, 2 и 3. Его стоимость равна 10+2=12.

# Вариант 15

Слово называется палиндромом, если его первая буква совпадает с последней, вторая – с предпоследней и т.д. Например: "abba", "madam", "x".

Для заданного числа K слово называется почти палиндромом, если в нем можно изменить не более K любых букв так, чтобы получился палиндром. Например, при K = 2 слова "reactor", "kolobok", "madam" являются почти палиндромами (подчеркнуты буквы, заменой которых можно получить палиндром).

Подсловом данного слова являются все слова, получающиеся путем вычеркивания из данного нескольких (возможно, одной или нуля) первых букв и нескольких последних. Например, подсловами слова "cat" являются слова "c", "a", "t", "ca", "at" и само слово "cat" (а "ct" подсловом слова "cat" не является).

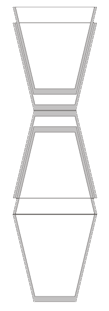
Требуется для данного числа K определить, сколько подслов данного слова S являются почти палиндромами.

В первой строке входного файла INPUT.TXT вводятся два натуральных числа: N (1 ≤ N ≤ 5 000) – длина слова и K (0 ≤ K ≤ N). Во второй строке записано слово S, состоящее из N строчных английских букв.

В выходной файл OUTPUT.TXT требуется вывести одно число – количество подслов слова S, являющихся почти палиндромами (для данного K).

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 5 1 abcde | 12 |
| 2 | 3 3 aaa | 6 |

# Вариант 16

Как известно, стакан – предмет весьма функциональный. Самый банальный способ применения – ёмкость для жидкости, самый оригинальный ещё не изобретён. А мальчик Слава строит из стаканов башни, пользуясь удивительным свойством стаканов ставиться друг на друга или вставляться друг в друга.

Слава строит башни из стаканов высотой 10 сантиметров, которых у него имеется бесконечное количество. Стакан можно поставить на уже имеющуюся конструкцию либо дном вниз, либо дном вверх. Если предыдущий стакан установлен аналогично новому, то конструкция вырастет на 1 сантиметр, так как стаканы надеваются друг на друга. В противном случае башня вырастет на 10 сантиметров.

Однажды Слава заметил, что ни в коем случае нельзя вставлять друг в друга более трёх стаканов, иначе один из стаканов обязательно разобьётся.

На рисунке показан пример башни высотой 32 сантиметра из 5 стаканов.

Слава умудрился построить красивую башню высотой k сантиметров. Но когда он пошёл за фотоаппаратом, чтобы запечатлеть это достижение, случайно задел конструкцию, и башня упала. Пытаясь восстановить своё творение, Слава понял, что есть несколько способов построить башню аналогичной высоты. Помогите Славе вычислить точное количество способов.

Входной файл INPUT.TXT содержит натуральное число k (k ≤ 105).

В выходной файл OUTPUT.TXT выведите количество способов построить башню заданной высоты, взятое по модулю 106.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 11 | 2 |
| 2 | 22 | 6 |
| 3 | 32 | 12 |

# Вариант 17

Будем называть натуральное число *трипростым*, если в нем любые подряд идущие 3 цифры образуют трехзначное простое число.

Требуется найти количество N-значных трипростых чисел.

Входной файл INPUT.TXT содержит натуральное число N (3 ≤ N ≤ 10000).

Выходной файл OUTPUT.TXT должен содержать количество N-значных трипростых чисел, которое следует вывести по модулю 109+9.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 4 | 204 |

# Вариант 18

Саша, не сделал домашнюю работу, зато купил шоколадку. И, по глупости, начал распечатывать ее прямо на уроке... Шелест золотинки услышала учительница. Она хотела вызвать в школу родителей, но Саша уговорил ее не вызывать их, а дать дополнительное задание.

Учительница внимательно посмотрела на шоколадку (она была размером 3х4 плиток), разделила на кусочки по две плитки и угостила всех, кто сделал домашнюю работу. А Сашу попросила написать программу, которая определяет, сколько существует способов деления шоколадки размером 3×N плиток на кусочки по две плитки.

Для выполнения задания Саше нужна помощь.

Входной файл INPUT.TXT содержит натуральное число N – размер плитки, (N < 33).

В выходной файл OUTPUT.TXT выведите одно целое число – количество возможных способов.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 2 | 3 |
| 2 | 4 | 11 |

# Вариант 19

Последовательность из английских букв строится следующим образом. Вначале она пуста. На каждом последующем шаге последовательность удваивается, после чего к ней слева дописывается очередная буква английского алфавита (a, b, c, …). Ниже приведены первые шаги построения по-следовательности:

Шаг 1. a

Шаг 2. baa

Шаг 3. cbaabaa

Шаг 4. dcbaabaacbaabaa

…………………………

Требуется написать программу, которая по заданному числу N находит символ, который стоит на N-ом месте в последовательности, получившейся после 26-го шага.

В единственной строке входного файла INPUT.TXT записано число N (1 ≤ N < 226).

В единственную строку выходного файла OUTPUT.TXT нужно вывести символ, стоящий в N-й позиции получившейся последовательности.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 4 | w |

# Вариант 20

Земельный комитет города принял решение о сдаче в аренду части муниципальной территории, имеющей форму прямоугольника размером H на W километров. Стоимость аренды каждого квадратного участка 1×1 км была определена с учётом локальных условий, и занесена в таблицу.

С целью организации открытого тендера на аренду, земельный комитет решил выставить на своём веб-сайте карту территории, и предоставить посетителям возможность узнавать суммарную стоимость аренды для произвольной прямоугольной группы соседних участков.

Данное предложение вызвало большой интерес у населения и предпринимателей, и нагрузка на сервер очень высока.

Требуется написать программу, позволяющую как можно более эффективно рассчитывать стоимость аренды для N запросов. В каждом запросе требуется определить общую стоимость участков внутри прямоугольной группы с противоположными углами, расположенными в элементах таблицы (ai, bi) и (ci, di).

В первой строке входного файла INPUT.TXT находятся числа H, W, N (1 ≤ H, W ≤ 100, 1 ≤ N ≤ 106). В следующих H строках содержится по W чисел (стоимости участков находятся в диапазоне от 0 до 10 000). Далее идут N строк с числами ai, bi, ci и di (1 ≤ ai ≤ ci ≤ H, 1 ≤ bi ≤ di ≤ W).

В выходной файл OUTPUT.TXT должен содержать N чисел, по одному числу в строке.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 2 3 1 5 1 2 6 7 3 2 1 2 3 | 16 |

# Вариант 21

Русский бизнесмен Иван Петров закупил в Китае большую партию наручных часов, чтобы продать их на родине за полцены (т.е. в 5 раз дороже, чем они стоили в Китае). Иван столкнулся с проблемой: китайские часы оказались некачественными. Мало того, что часы работали на протяжении всего нескольких часов, пока их не стукнешь, так еще и время подводить неудобно: вращать можно не минутную, а только секундную стрелку, причем, что самое ужасное, только в одну сторону в направлении увеличении времени. Например, для того, чтобы подвести часы на секунду назад, необходимо было сделать более 700 полных оборотов секундной стрелки, на что Иван бы потратил более 10 минут.

Чтобы продать эти часы оптом Ивану необходимо на момент сделки создать видимость того, что часы исправны. Для этого он собирается остановить все часы, установить их на одно и то же время. А перед сделкой ударить по чемодану с часами, чтобы они все дружно пошли.

Помогите Ивану выяснить: какое время на часах лучше установить для того, чтобы Иван потратил как можно меньше времени для того, чтобы подвести все часы.

В первой строке входного файла INPUT.TXT содержится натуральное число N – количество часов (N ≤ 50000). В последующих N строках располагаются показания всех часов в формате h:mm:ss, где h – показывает который час, mm – минуты, ss - секунды (1 ≤ h ≤ 12, 0 ≤ mm ≤ 59, 0 ≤ ss ≤ 59).

Выходной файл OUTPUT.TXT должен содержать время, которое нужно установить на всех часах, в формате, указанном выше. В случае неоднозначного ответа выведите наименьшее время.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 8:19:16 2:05:11 12:50:07 | 2:05:11 |

# Вариант 22

Задана матрица размером m×n из целых чисел. Путь начинается в любой строке первого столбца и состоит из последовательности шагов, обрывающихся в столбце n. Каждый шаг состоит в переходе из столбца i в столбец i+1 в соседнюю (по горизонтали или диагонали) ячейку. Весом пути называется сумма целых чисел, записанных в каждой из n посещенных ячеек.

Требуется написать программу, которая вычисляет путь с минимальным весом с левого края матрицы до правого.

Входной текстовый файл INPUT.TXT содержит в первой строке количество строк и столбцов матрицы, которые обозначаются m и n соответственно. Далее следует m строк по n чисел в каждой. Числа отделяются друг от друга пробелами. Число строк не превышает 10, столбцов – 100. Вес любого пути не будет превышать целого числа, для хранения которого потребуется больше 30 бит.

Выходной текстовый файл OUTPUT.TXT должен содержать две строки. Первая строка задает путь минимальной стоимости, а вторая – соответственно стоимость этого пути. Путь состоит из последовательности n целых чисел (разделенных одним или более пробелами), задающих номера строк, из которых состоит путь минимальной стоимости. Если путей минимальной стоимости больше одного, то должен быть выведен лексикографически минимальный путь.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 5 6 3 4 1 2 8 6 6 1 8 2 7 4 5 9 3 9 9 5 8 4 1 3 2 6 3 7 2 8 6 4 | 1 2 3 4 4 5 16 |

# Вариант 23

У вас имеется неограниченное количество монеток N разных достоинств. Определить, можно ли с их помощью разменять заданные К сумм денег.

В первой строке входного файла INPUT.TXT задано число N, далее во второй строке записаны N чисел, задающих достоинства монеток. В третьей строке задано число К – количество сумм. В четвертой строке располагаются К чисел, определяющих размеры сумм. Все числа во входном файле натуральны и не превосходят 1000.

В выходной файл OUTPUT.TXT выведите К чисел через пробел: для каждой суммы следует вывести 0, если ее разменять нельзя, и 1, если можно.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 2 3 5 5 3 6 7 11 12 | 1 1 0 1 1 |
| 2 | 1 10 1 3 | 0 |

# Вариант 24

Кубик, грани которого помечены цифрами от 1 до 6, бросают N раз. Требуется найти вероятность того, что сумма выпавших чисел будет равна Q.

Входной файл INPUT.TXT содержит натуральные числа N и Q (N ≤ 500, Q ≤ 3000).

В выходной файл OUTPUT.TXT выведите единственное вещественное число – искомую вероятность, которая должна отличаться от истинного значения не более чем на 10-6.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 1 6 | 0.166667 |
| 2 | 1 7 | 0 |
| 3 | 4 14 | 0.112654 |
| 4 | 100 100 | 1.530647E-78 |

# Вариант 25

Сколько N-значных чисел можно составить, используя цифры 5 и 9, в которых три одинаковые цифры не стоят рядом?

Во входном файле INPUT.TXT записано число N (1 ≤ N ≤ 30).

В выходной файл OUTPUT.TXT нужно вывести одно число - количество чисел с указанным свойством.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 | 6 |

# Вариант 26

В околоземном космическом пространстве накопилось много мусора, поэтому ученые сконструировали специальный аппарат ловушку для космического мусора. Аппарат должен двигаться по достаточно сложной траектории, сжигая по пути мусор. Ловушка может передвигаться в пространстве по 6 направлениям: на север (N), на юг (S), на запад (W), на восток (E), вверх (U) и вниз (D). Движением ловушки управляет процессор. Программа движения задается шестью правилами движения, которые соответствуют каждому из указанных направлений. Каждое такое правило представляет собой строку символов из множества {N, S, W, E, U, D}.

Команда ловушки состоит из символа направления и целого положительного числа M. Если параметр больше 1, то ловушка перемещается на один метр в направлении, которое указано в команде, а затем последовательно выполняет команды, заданные правилом для данного направления, с параметром меньше на 1. Если же параметр равен 1, то просто перемещается на один метр в указанном направлении.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Пусть, например, заданы правила, отраженные в таблице справа. Тогда при выполнении команды S(3) мусорщик сначала переместится на 1 метр в направлении S, а потом выполнит последовательно команды N(2), U(2), S(2), D(2), D(2), U(2), S(2), E(2).  Если далее проанализировать действия мусорщика, получим, что в целом он совершит ровно 34 перемещения. | |  |  | | --- | --- | | **Направление** | **Правило** | | N | N | | S | NUSDDUSE | | W | UEWWD | | E |  | | U | U | | D | WED | |

Первые шесть строк входного файла INPUT.TXT задают правила для команд с направлением N, S, W, E, U и D соответственно. Каждая строка содержит не более 100 символов (и может быть пустой). Следующая строка содержит команду ловушки: сначала символ из множества {N, S, W, E, U, D}, затем пробел и параметр команды – целое положительное число, не превышающее 100.

Выведите в выходной файл OUTPUT.TXT единственное число - количество перемещений, которое совершит ловушка. Гарантируется, что ответ не превышает 109.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | N NUSDDUSE UEWWD  U WED S 3 | 34 |

# Вариант 27

На вершине лесенки, содержащей N ступенек, находится мячик, который начинает прыгать по ним вниз, к основанию. Мячик может прыгнуть на следующую ступеньку, на ступеньку через одну или через две. То есть, если мячик лежит на 8-ой ступеньке, то он может переместиться на 5-ую, 6-ую или 7-ую.

Требуется написать программу, которая определит число всевозможных "маршрутов" мячика с вершины на землю.

Входной файл INPUT.TXT содержит число N (0 < N ≤ 70).

Выходной файл OUTPUT.TXT должен содержать искомое число.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 1 | 1 |
| 2 | 4 | 7 |

# Вариант 28

Задан целочисленный прямоугольный массив M×N. Необходимо определить прямоугольную область данного массива, сумма элементов которого максимальна.

В первой строке входного файла INPUT.TXT записаны два натуральных числа N и M (1 ≤ N, M ≤ 100) – количество строк и столбцов прямоугольной матрицы. Далее идут N строк по M чисел, записанных через пробел – элементы массива, целые числа, не превосходящие 100 по абсолютной величине.

В выходной файл OUTPUT.TXT необходимо вывести целое число – сумму элементов найденного прямоугольного подмассива. Подмассив должен содержать хотя бы один элемент.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 2 3 5 0 9 1 2 7 | 24 |
| 2 | 4 5 -7 8 -1 0 -2 2 -9 2 4 -6 -7 0 6 8 1 4 -8 -1 0 -6 | 20 |

# Вариант 29

В одной военной части было принято революционное решение перейти от портянок к носкам. По такому случаю прапорщику Недалекому было поручено закупить n пар носков. Однако предложенная прапорщиком смета не удовлетворила начальство, и прапорщику было предложено очень-очень быстро переделать ее так, чтобы затраты были минимально возможными. Помогите бедному прапорщику составить такую смету.

Изучение рынка показало, что всего существует m различных поставщиков, которые продают носки разными пачками и по разным ценам. Пачка, содержащая ai пар носков, продается за bi рублей.

Разрешено покупать любое количество пачек у одного поставщика. Разрешено покупать пачки у нескольких поставщиков.

В первой строке входного файла INPUT.TXT содержатся числа n и m (1 ≤ n ≤ 10000, 1 ≤ m ≤ 100). Далее идут m пар чисел ai, bi (1 ≤ ai ≤ 10000, 1 ≤ bi ≤ 10000).

Выведите в выходной файл OUTPUT.TXT минимальную сумму денег, которую нужно потратить на покупку n пар носок.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 9 2 1 1 10 8 | 8 |

# Вариант 30

За билетами на премьеру нового мюзикла выстроилась очередь из N человек, каждый из которых хочет купить 1 билет. На всю очередь работала только одна касса, поэтому продажа билетов шла очень медленно, приводя «постояльцев» очереди в отчаяние. Самые сообразительные быстро заметили, что, как правило, несколько билетов в одни руки кассир продаёт быстрее, чем когда эти же билеты продаются по одному. Поэтому они предложили нескольким подряд стоящим людям отдавать деньги первому из них, чтобы он купил билеты на всех.

Однако для борьбы со спекулянтами кассир продавала не более 3-х билетов в одни руки, поэтому договориться таким образом между собой могли лишь 2 или 3 подряд стоящих человека.

Известно, что на продажу i-му человеку из очереди одного билета кассир тратит Ai секунд, на продажу двух билетов — Bi секунд, трех билетов — Ci секунд. Напишите программу, которая подсчитает минимальное время, за которое могли быть обслужены все покупатели.

Обратите внимание, что билеты на группу объединившихся людей всегда покупает первый из них. Также никто в целях ускорения не покупает лишних билетов (то есть билетов, которые никому не нужны).

Во входном файле INPUT.TXT записано сначала число N — количество покупателей в очереди (1≤N≤5000). Далее идет N троек натуральных чисел Ai, Bi, Ci. Каждое из этих чисел не превышает 3600. Люди в очереди нумеруются, начиная от кассы.

В выходной файл OUTPUT.TXT выведите одно число — минимальное время в секундах, за которое могли быть обслужены все покупатели.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 5 5 10 15 2 10 15 5 5 5 20 20 1 20 1 1 | 12 |
| 2 | 2 3 4 5 1 1 1 | 4 |

# Вариант 31

Любое натуральное число можно представить в виде суммы натуральных слагаемых, каждое из которых является степенью числа 2. Суммы, различающиеся лишь порядком слагаемых, считаются одинаковыми. Например, для числа 7 таких представлений 6 (4+2+1, 4+1+1+1, 2+2+2+1, 2+2+1+1+1, 2+1+1+1+1+1, 1+1+1+1+1+1+1).

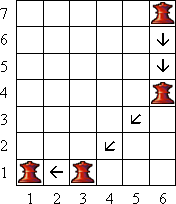
Требуется написать программу, которая найдет количество способов такого представления заданного числа N.

Входной файл INPUT.TXT содержит число N (1 ≤ N ≤ 1000).

В выходной файл OUTPUT.TXT выведите одно число – найденное количество способов представления числа N.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 4 | 4 |
| 2 | 7 | 6 |

# Вариант 32

Рассмотрим бесконечную вправо и вверх шахматную доску, на которой стоит ферзь. Двое по очереди двигают этого ферзя. Разрешается двигать ферзя только вниз, влево или по диагонали вниз влево на любое положительное количество клеток в выбранном направлении. Цель игры – задвинуть ферзя в угол, то есть клетку с координатами (1, 1). На рисунке показаны разрешенные движения ферзя.

Требуется написать программу, которая найдет номер игрока, который выиграет при правильной игре.

Входной файл INPUT.TXT содержит координаты ферзя перед первым ходом - два числа M и N, записанные через пробел (1 ≤ M, N ≤ 250). Гарантируется, что ферзь изначально не находится в клетке с координатами (1,1).

Выходной файл OUTPUT.TXT должен содержать найденный номер победителя.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 2 | 2 |
| 2 | 6 7 | 1 |

# Вариант 33

N гангстеров собираются в ресторан. i-й гангстер приходит в момент времени Ti и имеет богатство Pi. Дверь ресторана имеет K+1 степень открытости, они обозначаются целыми числами из интервала [0, K]. Степень открытости двери может изменяться на единицу в единицу времени, то есть дверь может открыться на единицу, закрыться на единицу или остаться в том же состоянии. В начальный момент времени дверь закрыта (степень открытости 0). i-й гангстер заходит в ресторан, только если дверь открыта специально для него, то есть когда степень открытости двери соответствует его полноте Si. Если в момент, когда гангстер подходит к ресторану, степень открытости двери не соответствует его полноте, он уходит и больше не возвращается. Ресторан работает в интервале времени [0, T].

Требуется собрать гангстеров с максимальным суммарным богатством в ресторане, открывая и закрывая дверь соответствующим образом.

В первой строке входного файла INPUT.TXT находятся числа N, K, T, во второй - T1, T2, ..., TN, в третьей - P1, P2, ..., PN. в четвёртой - S1, S2, ..., SN. Числа в строках разделены пробелами. Ограничения: 1 ≤ N ≤ 100, 1 ≤ K ≤ 100, 1 ≤ T ≤ 30 000, 0 ≤ Ti ≤ T, 1 ≤ Pi ≤ 300, 1 ≤ Si ≤ K .

В выходной файл OUTPUT.TXT выведите одно число - максимальное суммарное богатство гангстеров, попавших в ресторан. Если зайти не удалось никому, вывести 0.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 4 10 20 10 16 8 16 10 11 15 1 10 7 1 8 | 26 |
| 2 | 2 17 100 5 0 50 33 6 1 | 0 |

# Вариант 34

Когда Миша и Маша покупали подарок, возникла интересная ситуация. У них была в распоряжении только одна большая купюра, а у продавца – некоторое количество мелочи. Дело происходило утром, поэтому продавцу нужно было экономить мелочь, и он хотел отдать сдачу минимальным количеством монет. Подумав некоторое время, они точно определили, с каким количеством монет продавцу придется расстаться.

А вы сможете решить такую задачу?

В первой строке входного файла INPUT.TXT записано число N (1 ≤ N ≤ 10) – количество различных номиналов монет, содержащихся в кассе. Можно считать, что количество монет каждого номинала достаточно. На следующей строке содержится N целых чисел ai (0 < ai ≤ 2000) – номиналы монет. В третьей строке записано одно число K (1 ≤ K ≤ 106) – сумма, которую нужно набрать.

В выходной файл OUTPUT.TXT выведите минимальное количество монет, которое придется отдать продавцу, или -1, если продавец вообще не сможет дать им сдачу.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 1 5 7 19 | 3 |
| 2 | 5 7 8 9 10 11 13 | -1 |

# Вариант 35

В этом году Иван Иванович решил отметить приход осени субботником, чтобы убрать весь мусор во дворе дома номер 31 по улице Осенней. На субботник он пригласил N знакомых старушек, живущих в том же самом доме. Однако, в самом начале мероприятия выяснилось, что по одиночке старушки работают плохо, так как им хочется во время работы еще и поговорить друг с другом.

Иван Иванович подумал и принял волевое решение разбить старушек на группы так, чтобы в каждой группе было не менее 2 старушек. Старушки отличаются друг от друга уровнем разговорчивости, и если в одну группу попадут две старушки, у одной из которых маленький уровень разговорчивости, а у второй - большой, то они не могут поговорить друг с другом и работа будет стопориться.

Назовем разговорчивостью группы разность между максимальным и минимальным уровнями разговорчивости старушек в группе. Например, если уровни разговорчивости старушек в группе равны 7, 3 и 11, то разговорчивость группы равна 11 - 3 = 8. Разговорчивостью разбиения старушек на группы назовем максимальную из разговорчивостей групп, входящих в разбиение.

Требуется написать программу, которая поможет Ивану Ивановичу найти разбиение старушек на группы, разговорчивость которого минимальна.

Входной файл INPUT.TXT содержит в первой строке число N (2 ≤ N ≤ 1000) – количество старушек. Во второй строке записано N чисел от 1 до 109 – разговорчивости старушек.

Выходной текстовый файл OUTPUT.TXT должен содержать одно целое число, равное минимально возможной разговорчивости разбиения старушек на группы.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 2 1 1000000000 | 999999999 |
| 2 | 3 1 2 3 | 2 |
| 3 | 8 1 10 100 1000 1000 100 10 1 | 0 |
| 4 | 10 258 740 156 244 458 680 390 694 844 817 | 102 |

# Вариант 36

Напомним, что cтрока B = b1b2b3...bm, является подпоследовательностью строки A = a1a2a3...an, если существует строго возрастающая последовательность {i1, i2, i3, ... , im} индексов A, такая, что для всех j от 1 до m выполняется Aij=Bj. Например, B = ”aba” является подпоследовательностью строки A = ”abacaba”. Последовательность индексов в этом случае может быть такой: {1, 2, 3}.

Пусть Вам дана строка S, состоящая только из маленьких букв английского алфавита. Ваша задача заключается в том, чтобы посчитать количество ее подпоследовательностей “abc”.

Входной файл INPUT.TXT содержит строку S, длиной не более 105 символов.

В выходной файл OUTPUT.TXT выведите ответ на задачу.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | abc | 1 |
| 2 | ab | 0 |

# Вариант 37

Недавно на уроке информатики ученики одного из классов изучили булевы функции. Напомним, что булева функция f сопоставляет значениям двух булевых аргументов, каждый из которых может быть равен 0 или 1, третье булево значение, называемое результатом. Для учеников, которые выразили желание более подробно изучать эту тему, учительница информатики на дополнительном уроке ввела в рассмотрение понятие цепного вычисления булевой функции f.

Если задана булева функция f и набор из N булевых значений a1, a2, ..., aN , то результат цепного вычисления этой булевой функции определяется следующим образом:

* если N = 1, то он равен a1;
* если N > 1, то он равен результату цепного вычисления булевой функции f для набора из (N–1) булевого значения f(a1,a2), a3, …, aN, который получается путем замены первых двух булевых значений в наборе из N булевых значений на единственное булево значение – результат вычисления функции f от a1 и a2.

Например, если изначально задано три булевых значения: a1 = 0, a2 = 1, a3 = 0, а функция f – ИЛИ (OR), то после первого шага получается два булевых значения – (0 OR 1) и 0, то есть, 1 и 0. После второго (и последнего) шага получается результат цепного вычисления, равный 1, так как 1 OR 0 = 1.

В конце дополнительного урока учительница информатики написала на доске булеву функцию f и попросила одного из учеников выбрать такие N булевых значений ai, чтобы результат цепного вычисления этой функции был равен единице. Более того, она попросила найти такой набор булевых значений, в котором число единиц было бы как можно большим.

Требуется написать программу, которая решала бы поставленную учительницей задачу.

Первая строка входного файла INPUT.TXT содержит одно натуральное число N (2 ≤ N ≤ 100 000). Вторая строка содержит описание булевой функции в виде четырех чисел, каждое из которых – ноль или единица.

Первое из них есть результат вычисления функции в случае, если оба аргумента – нули, второе – результат в случае, если первый аргумент – ноль, второй – единица, третье – результат в случае, если первый аргумент – единица, второй – ноль, а четвертый – в случае, если оба аргумента – единицы.

В выходной файл OUTPUT.TXT необходимо вывести строку из N символов, определяющих искомый набор булевых значений ai с максимально возможным числом единиц. Если ответов несколько, требуется вывести любой из них. Если такого набора не существует, выведите в выходной файл фразу «No solution».

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 4 0110 | 1011 |
| 2 | 5 0100 | 11111 |
| 3 | 6 0000 | No solution |

# Вариант 38

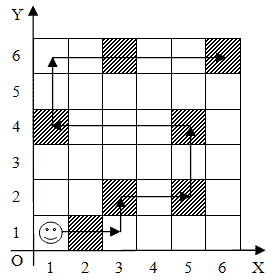
Вы любите играть в игры? Конечно, любите! Но про эту игру, возможно, ничего не знаете и не слышали даже. Что ж, расскажем о новой игре. На доске написана последовательность n целых чисел. Играют двое. На очередном ходе игрок выбирает число с правого или с левого края последовательности, затем это число стирается и последовательность становится на одно число меньше, а ход переходит к противнику. Выигрывает тот, кто наберет в сумме больше. Написать программу, определяющую победителя в конкретной игре, при условии, что игроки будут играть оптимально.

В первой строке входного файла INPUT.TXT записано целое число n (0 < n < 100). Во второй строке через пробел заданы n натуральных чисел, не превосходящих 1000.

В единственную строку выходного файла OUTPUT.TXT нужно вывести 1, если победит первый игрок, 2 – если победит второй игрок и 0 – в случае ничьей.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 4 3 2 5 4 | 1 |
| 2 | 6 5 5 5 5 5 5 | 0 |
| 3 | 9 2 1 3 2 9 1 2 3 1 | 2 |
| 4 | 10 2 5 3 12 4 6 13 7 1 3 | 1 |

# Вариант 39

Всем известно, что работа уборщицы становится все менее престижной, чем работа программиста. Поэтому, все больше становится программистов и все меньше уборщиц. И скоро, возможно, совсем некому будет делать уборку помещений, а чистота – она всегда актуальна и важна для работы, например, для тех же программистов.

Сотрудники одной из фирм разработали специальную машину «Мусорщик-001», которая предназначена для уборки прямоугольных пустых помещений. Машина не совершенна и может пока двигаться на 1 метр только влево, вправо и вперед (вдоль оси OY). Каждое помещение можно разбить на квадратные сектора со стороной в 1 метр и обозначить те, которые загрязнены. Для уборки помещения достаточно, чтобы машина-уборщик побывала в каждом из загрязненных секторов. Известно, что перед уборкой машина всегда находится в клетке (1,1) .

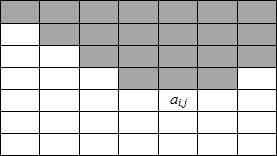
Одна из компаний, где в штате нет уборщицы, но имеется полный штат программистов, приобрела «Мусорщик-001». Пока программистам никак не удается написать программу, определяющую по заданному плану загрязнения помещения минимально возможную длину маршрута машины-уборщика, необходимого для уборки данной территории. Возможно, Вам удастся им помочь!

Первая строка входного файла INPUT.TXT содержит натуральное число n (n ≤ 1000). Следующие n строк содержат по два натуральных числа: xi и yi – координаты загрязненных секторов в заданном помещении (xi, yi ≤ 50).

В выходной файл OUTPUT.TXT выведите целое число, соответствующее минимальной длине маршрута в метрах, необходимого для уборки.

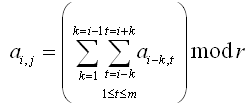
|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 7 2 1 3 2 5 2 5 4 1 4 3 6 6 6 | 18 |

# Вариант 40

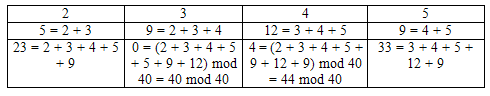
Рассмотрим прямоугольную таблицу размером n×m. Занумеруем строки таблицы числами от 1 до n, а столбцы – числами от 1 до m. Будем такую таблицу последовательно заполнять числами следующим образом.

Обозначим через aij число, стоящее на пересечении i-ой строки и j-ого столбца. Первая строка таблицы заполняется заданными числами – a11, a12, …, a1m. Затем заполняются строки с номерами от 2 до n. Число aij вычисляется как сумма всех чисел таблицы, находящихся в «треугольнике» над элементом aij. Все вычисления при этом выполняются по модулю r.

Более точно, значение aij вычисляется по следующей формуле:



Например, если таблица состоит из трех строк и четырех столбцов, и первая строка состоит из чисел 2,3,4,5, а r = 40 то для этих исходных данных таблица будет выглядеть следующим образом (взятие по модулю показано только там, где оно приводит к изменению числа):



Требуется написать программу, которая по заданной первой строке таблицы (a11, a12, …, a1m), вычисляет последнюю строку, как описано выше.

Первая строка входного файла INPUT.TXT содержит числа n, m и r (2 ≤ n, m ≤ 2000, 2 ≤ r ≤ 109) – число строк и столбцов таблицы соответственно, а так же число, по модулю которого надо посчитать ответ. Следующая строка содержит m целых чисел – первую строку таблицы: a11, a12, …, a1m. Все a1i неотрицательны и не превосходят 109.

В первой строке выходного файла OUTPUT.TXT необходимо вывести m чисел – последнюю строку таблицы: an1, an2, …, anm.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 2 3 10 1 2 3 | 3 6 5 |
| 2 | 3 3 10 1 1 1 | 8 0 8 |
| 3 | 3 4 40 2 3 4 5 | 23 0 4 33 |

# Вариант 41

Вася уже долго занимается K-расширениями. Недавно ему это надоело, и он стал изучать следующую операцию на строках: сначала удаляется каждый K-ый символ строки с начала и до конца (т. е. сначала K-й, потом 2K-й, и т. д., пока число iK не превосходит длины строки). Потом эта операция повторяется, и так далее, пока в строке есть хотя бы K символов.

Например, если дана строка длины 10, и K = 2, то сначала будут удалены 2, 4, 6, 8 и 10 символы. Затем будут удалены 2 и 4 символы новой строки, которые соответствуют 3 и 7 символам старой. Затем будет удален 2 символ получившейся строки - 5 символ старой. Последним будет удален 9 символ старой строки.

Собственно, задумал он это все для того, чтобы узнать, какой и когда символ исчезнет. Но когда ему надоело вручную удалять символы, он решил поручить это Вам. А именно, Вам придется отвечать на его вопросы: "А на какой секунде был удален i-ый символ строки?" На удаление каждого символа тратится одна секунда.

В первой строке входного файла INPUT.TXT находятся три числа: 1 ≤ N ≤ 5×106 - количество символов, которые написал Вася, число 1 ≤ K ≤ N, и 1 ≤ L ≤ 10 000 - количество вопросов, которые задает Вася. В последующих L строках указаны номера символов, для которых Вася хочет узнать, когда они были удалены. Все номера лежат в пределах от 1 до N и могут повторяться.

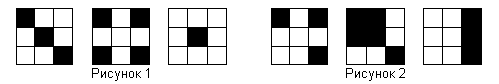
В выходной файл OUTPUT.TXT выведите L строк с ответами на вопросы в том порядке, в каком они были во входном файле. Если окажется, что символ из строки не будет удален никогда, выведите 0.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 10 2 6 1 2 3 5 10 5 | 0 1 6 8 5 8 |

# Вариант 42

Компания BrokenTiles планирует заняться выкладыванием во дворах у состоятельных клиентов узор из черных и белых плиток, каждая из которых имеет размер 1×1 метр. Известно, что дворы всех состоятельных людей имеют наиболее модную на сегодня форму прямоугольника M×N метров.

Однако при составлении финансового плана у директора этой организации появилось целых две серьезных проблемы: во первых, каждый новый клиент очевидно захочет, чтобы узор, выложенный у него во дворе, отличался от узоров всех остальных клиентов этой фирмы, а во вторых, этот узор должен быть симпатичным. Как показало исследование, узор является симпатичным, если в нем нигде не встречается квадрата 2×2 метра, полностью покрытого плитками одного цвета. На рисунке 1 показаны примеры различных симпатичных узоров, а на рисунке 2 – несимпатичных.



Для составления финансового плана директору необходимо узнать, сколько клиентов он сможет обслужить, прежде чем симпатичные узоры данного размера закончатся. Помогите ему!

В первой строке входного файла INPUT.TXT находятся два положительных целых числа, разделенные пробелом – M и N (1 ≤ M∙N ≤ 30).

Выведите в выходной файл OUTPUT.TXT единственное число – количество различных симпатичных узоров, которые можно выложить во дворе размера M×N. Узоры, получающиеся друг из друга сдвигом, поворотом или отражением считаются различными.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 2 2 | 14 |
| 2 | 3 3 | 322 |

# Вариант 43

Около Петиного университета недавно открылось новое кафе, в котором действует следующая система скидок: при каждой покупке более чем на 100 рублей покупатель получает купон, дающий право на один бесплатный обед (при покупке на сумму 100 рублей и меньше такой купон покупатель не получает).

Однажды Пете на глаза попался прейскурант на ближайшие N дней. Внимательно его изучив, он решил, что будет обедать в этом кафе все N дней, причем каждый день он будет покупать в кафе ровно один обед. Однако стипендия у Пети небольшая, и поэтому он хочет по максимуму использовать предоставляемую систему скидок так, чтобы его суммарные затраты были минимальны.

В первой строке входного файла INPUT.TXT записано целое число N (0 ≤ N ≤ 100). В каждой из последующих N строк записано одно целое число, обозначающее стоимость обеда в рублях на соответствующий день. Стоимость — неотрицательное целое число, не превосходящее 300.

В первой строке выходного файла OUTPUT.TXT выдайте минимальную возможную суммарную стоимость обедов. Во второй строке выдайте два числа K1 и K2 — количество купонов, которые останутся неиспользованными у Пети после этих N дней и количество использованных им купонов соответственно. Если существует несколько решений с минимальной суммарной стоимостью, то выдайте то из них, в котором значение K1 максимально (на случай, если Петя когда-нибудь ещё решит заглянуть в это кафе).

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 5 35 40 101 59 63 | 235 0 1 |

# Вариант 44

Изделие изготавливают из n блоков, каждый из которых имеет два технологических параметра – mi и ki. Известно, что ki=mi+1, i=1, 2, …, n-1. При этом условии два последовательных блока i и i+1 можно объединять в один новый, который будет иметь технологические параметры - mi и ki+1, и на это потребуется mi\*ki+1 технологических операций. Новый блок можно опять объединять с одним из соседних и так далее. Меняя порядок сборки блоков можно добиться уменьшения количества технологических операций.

Поясним это на примере трех блоков: 34 и 29, 29 и 4, 4 и 15. Если собрать вначале 2 и 3 блок, а затем присоединить собранное к первому, то потребуется 29\*15+34\*15=435+510=945 операций. Если собрать вначале блок из 1 и 2 исходных блоков, а затем присоединить 3 блок, то потребуется 34\*4+34\*15=136+510=646 операций.

Требуется написать программу, которая найдет минимальное число технологических операций для изготовления изделия.

Входной файл INPUT.TXT содержит в первой строке число n – количество блоков (1 ≤ n ≤ 100). Последующие n строк содержат пары чисел (разделенных пробелом) – технологические параметры блоков. Технологические параметры – целые неотрицательные числа, не превышающие 100.

Выходной текстовый файл OUTPUT.TXT должен содержать одно число – минимальное число технологических операций.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 3 34 29 29 4 4 15 | 646 |

# Вариант 45

В сообщении, состоящем из заглавных русских букв и пробелов, каждую букву заменили её порядковым номером в русском алфавите (А - 1, Б - 2, ..., Я - 33), а пробел - нулем.

Требуется по заданной последовательности цифр найти количество исходных сообщений, из которых она могла получиться.

Входной файл INPUT.TXT содержит последовательность цифр, состоящую не более чем из 100 цифр.

В выходной файл OUTPUT.TXT выведите ответ на задачу.

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 1025 | 4 |
| 2 | 21705 | 3 |
| 3 | 33222 | 8 |

# Вариант 46

Пицца – любимое лакомство Васи, он постоянно покупает и с удовольствием употребляет различные сорта этого великолепного блюда. Однажды, в очередной раз, разрезая круглую пиццу на несколько частей, Вася задумался: на какое максимальное количество частей можно разрезать пиццу за N прямых разрезов?

Помогите Васе решить эту задачу, определив максимальное число не обязательно равных кусков, которые может получить Вася, разрезая пиццу таким образом.

Входные данные

Входной файл INPUT.TXT содержит натуральное число N – число прямых разрезов пиццы (N ≤ 1000).

Выходные данные

В выходной файл OUTPUT.TXT выведите ответ на задачу.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 2 | 4 |
| 2 | 3 | 7 |

# Вариант 47

Садовник посадил N деревьев в один ряд. После посадки деревьев садовнику нужно их покрасить. В его распоряжении есть краска трех цветов: белая, синяя и оранжевая. Сколько способов покраски деревьев есть у него, если никакие два соседних дерева нельзя красить в одинаковый цвет?

Входные данные

В единственной строке входного файла INPUT.TXT записано одно натуральное число - количество деревьев N (1 ≤ N ≤ 50).

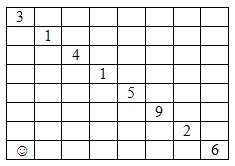
Выходные данные

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно число - количество способов покраски.

Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3 | 12 |

# Вариант 48



В левой нижней клетке шахматной доски размера N×N стоит пешка. Двое игроков по очереди двигают её, причём каждый может подвинуть её на одну клетку вверх или на одну клетку вправо. На диагонали доски написаны числа a1, a2, …, aN. Когда пешка попадает на диагональ, игра кончается и выигрыш первого игрока равен значению числа, написанного в клетке с остановившейся пешкой. Напишите программу определения гарантированного выигрыша первого игрока.

Входные данные

В первой строке входного файла INPUT.TXT записано число N (1 ≤ N ≤ 100). Во второй строке записаны натуральные числа a1, a2, …, aN (1 ≤ ai ≤ 100).

Выходные данные

В выходной файл OUTPUT.TXT выведите одно число – выигрыш первого игрока.

Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 8 3 1 4 1 5 9 2 6 | 5 |

# Вариант 49

A picture containing text, clock

Description automatically generated

В прямоугольной таблице N×M (в каждой клетке которой записано некоторое число) в начале игрок находится в левой верхней клетке. За один ход ему разрешается перемещаться в соседнюю клетку либо вправо, либо вниз (влево и вверх перемещаться запрещено). При проходе через клетку с игрока берут столько у.е., какое число записано в этой клетке (деньги берут также за первую и последнюю клетки его пути).

Требуется найти минимальную сумму у.е., заплатив которую игрок может попасть в правый нижний угол.

Входные данные

Во входном файле INPUT.TXT задано два числа N и M - размеры таблицы (1 ≤ N ≤ 20, 1 ≤ M ≤ 20). Затем идет N строк по M чисел в каждой - размеры штрафов в у.е. за прохождение через соответствующие клетки (числа от 0 до 100).

Выходные данные

В выходной файл OUTPUT.TXT выведите минимальную сумму, потратив которую можно попасть в правый нижний угол.

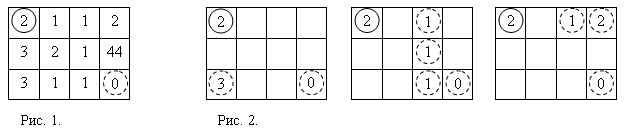
Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3 4 1 1 1 1 5 2 2 100 9 4 2 1 | 8 |
| 2 | 5 5 1 1 1 1 1 3 100 100 100 100 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 | 11 |

# Вариант 50

Игровое поле N×M заполняется целыми числами, одно неотрицательное целое число в каждой клетке. Цель игры состоит в том, чтобы пройти по любому разрешенному пути от верхнего левого угла до правого нижнего. Целое число в каждой клетке указывает, какой длины шаг должен быть из текущей клетки. Все шаги могут быть или направо или вниз. Если в результате какого-либо шага игрок покидает пределы поля, такой шаг запрещается.

На рис. 1 приведен пример игрового поля 3×4, где сплошная окружность показывает положение начала, а пунктирная окружность – цель. Рис. 2 показывает три возможных пути от начала до цели для рассматриваемого примера игрового поля, с удаленными промежуточными числами.



Требуется написать программу, которая определит число различных вариантов путей от верхнего левого угла до правого нижнего.

Входные данные

Входной файл INPUT.TXT содержит в первой строке размеры поля N (1 ≤ N ≤ 70) и M (1 ≤ M ≤ 70). В последующих N строках входного файла, каждая из которых описывает отдельную строку игрового поля, записаны через пробел по M целых чисел от 0 до 100 – длины шагов из клеток данной строки.

Выходные данные

Выходной файл OUTPUT.TXT должен содержать одно число - число различных вариантов путей от верхнего левого угла до правого нижнего. Для каждого поля будет менее чем 231 различных путей.

Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3 4 2 1 1 2 3 2 1 44 3 1 1 0 | 3 |

# Вариант 51

Агент Джеймс Бонд пошел на пенсию, но неугомонный характер требовал новых впечатлений. Поэтому Джеймс Бонд с удовольствием согласился провести мастер-класс в некоторых группах школы «Молодого агента». Тема одного из занятий – работа агента с напарником. В таком опасном деле, как разведка, важно иметь очень надёжного напарника, поэтому напарниками могут стать только агенты, которые максимально близки по возрасту (т.е. два агента не могут стать напарниками, если в группе существует третий агент, который старше одного и младше другого).

Задание Бонда состоит в том, чтобы агенты нашли друг другу напарников таким образом, чтобы у каждого агента был хотя бы один напарник (всего у агента может быть 2 напарника – один младше, и один старше него, но эти двое не считаются напарниками между собой). Очевидно, что группа из 4 и более агентов может поделиться несколькими способами.

После нескольких занятий Бонд узнал способности групп, обучающихся в школе «Молодого агента», и оценил риск раскрытия каждого агента в отдельности. Но специфика работы с напарником такова, что в паре риску подвергается только старший из двух агентов, поэтому группу надо распределить так, чтобы суммарный риск был минимален.

Входные данные

В первой строке входного файла INPUT.TXT находится одно целое число N – количество агентов в группе (2 ≤ N ≤ 10000). Во второй строке находятся N пар целых положительных чисел, разделенных пробелом. Первое число в паре – это возраст агента (в днях) из диапазона [5000, 16000], второе – риск раскрытия агента, число в диапазоне [1, 1000]. Известно, что в любой группе все агенты разного возраста.

Выходные данные

В выходной файл OUTPUT.TXT выведите единственное число – минимальное значение суммарного риска раскрытия группы.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3 6000 2 5500 3 5000 4 | 5 |
| 2 | 5 5005 1 5004 2 5003 3 5002 4 5001 5 | 7 |

# Вариант 52

На прямой дощечке вбиты гвоздики. Любые два гвоздика можно соединить ниточкой. Требуется соединить некоторые пары гвоздиков ниточками так, чтобы к каждому гвоздику была привязана хотя бы одна ниточка, а суммарная длина всех ниточек была минимальна.

Входные данные

В первой строке входного файла INPUT.TXT записано число N - количество гвоздиков (2 ≤ N ≤ 100). В следующей строке записано N чисел - координаты всех гвоздиков (неотрицательные целые числа, не превосходящие 10000).

Выходные данные

В выходной файл OUTPUT.TXT нужно вывести единственное число - минимальную суммарную длину всех ниточек.

Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 6 3 4 12 6 14 13 | 5 |

# Вариант 53

На расстоянии N шагов от магазина стоит человек. Каждую минуту он выбирает, куда сделать шаг: к магазину или в противоположном направлении.

Требуется написать программу, которая определит, сколькими способами он может попасть в магазин, пройдя ровно K шагов и оказавшись в магазине только после выполнения последнего шага.

Входные данные

Входной файл INPUT.TXT содержит два числа N и K, записанные через пробел. Известно, что 1 ≤ N ≤ K ≤ 37.

Выходные данные

Выходной файл OUTPUT.TXT должен содержать одно число – количество способов попадания в магазин.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 2 4 | 2 |
| 2 | 5 5 | 1 |

# Вариант 54

Дано натуральное число N. Над ним можно произвести следующий набор операций:

вычитать единицу;

делить на три, если число кратно трем;

делить на два, если число четное.

После выполнения одной из операций к полученному результату также можно применить указанные операции, и делается это до тех пор, пока результат не окажется равным 1.

Входные данные

Входной файл INPUT.TXT содержит натуральное число N (N ≤ 106).

Выходные данные

В выходной файл OUTPUT.TXT выведите наименьшее количество операций, в результате выполнения которых будет получена единица.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 5 | 3 |
| 2 | 1 | 0 |
| 3 | 10 | 3 |

# Вариант 55

Требуется вычислить количество N-значных чисел в системе счисления с основанием K, таких что их запись не содержит двух подряд идущих нулей.

Входные данные

Во входном файле INPUT.TXT записаны два натуральных числа N и K в десятичной системе счисления (2 ≤ K ≤ 10; 2 ≤ N; 4 ≤ N+K ≤ 18).

Выходные данные

В выходной файл OUTPUT.TXT необходимо вывести целое число в десятичной записи – ответ на задачу.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 2 10 | 90 |
| 2 | 4 2 | 5 |
| 3 | 6 3 | 328 |

# Вариант 56

Фермер решил на своем квадратном участке земли вспахать пашню квадратной формы максимальной площади, т.к. он посчитал, что именно квадратная форма пашни наиболее удобна для обработки. Но на его участке присутствуют деревья и хозяйственные постройки, которые он никуда не хочет переносить, а так же иные места, не пригодные для пашни. Для удобства он составил квадратную карту местности N×N в форме матрицы и пометил нулями непригодные для пашни зоны, в остальные зоны он поставил единицу.

Необходимо помочь фермеру определить максимальную площадь пашни.

Входные данные

В первой строке входного файла INPUT.TXT записано единственное натуральное число N (1 ≤ N ≤ 1000) – длина стороны квадратного участка фермы. Далее, следует N строк, в каждой из которых находится последовательность (без пробелов) нулей и единиц, описывающих ферму.

Выходные данные

В выходной файл OUTPUT.TXT необходимо вывести максимально возможную площадь пашни.

Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 7 1101101 1111110 1011100 0011100 1000010 1100111 1001110 | 9 |

# Вариант 57

Часто для пробного тура на различных олимпиадах по информатике предлагается задача «A + B», в которой по заданным целым числам A и B требуется найти их сумму.

При проведении городской олимпиады по информатике председатель жюри решил сам подготовить тесты для такой задачи. Для этого он использовал свою оригинальную методику, которая заключалась в следующем: сначала готовятся предполагаемые правильные ответы, а затем подбираются входные данные, соответствующие этим ответам.

Пусть председатель жюри выбрал число C, запись которого состоит из n десятичных цифр и не начинается с нуля. Теперь он хочет подобрать такие целые положительные числа A и B, чтобы их сумма была равна C, и запись каждого из них также состояла из n десятичных цифр и не начиналась с нуля. В дополнение к этому председатель жюри старается подобрать такие числа A и B, чтобы каждое из них было красивым. Красивым в его понимании является число, запись которого не содержит двух одинаковых подряд идущих цифр. Например, число 1272 считается красивым, а число 1227 — нет.

Требуется написать программу, которая для заданного натурального числа C вычисляет количество пар красивых положительных чисел A и B, сумма которых равна C. Поскольку количество пар красивых чисел может быть большим, необходимо вывести остаток от деления этого количества на число 109+7.

Входные данные

Входной файл INPUT.TXT содержит одно целое положительное число C. Число C не начинается с нуля. Количество цифр в записи числа С не превышает 10 000.

Выходные данные

В выходной файл OUTPUT.TXT должен содержать одно целое число — остаток от деления количества искомых пар красивых чисел A и B на число 109+7.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 22 | 2 |
| 2 | 200 | 0 |
| 3 | 1000 | 0 |
| 4 | 239 | 16 |

Пояснения к примерам

Число 22 можно представить в виде суммы двузначных чисел тремя способами: 10 + 12, 11 + 11, 12 + 10. Способ 11 + 11 не подходит, поскольку число 11 не является красивым. Следовательно, ответ для числа 22 равен 2.

Число 200 можно представить в виде суммы трехзначных чисел единственным способом: 100 + 100. Этот способ не подходит, поэтому ответ для числа 200 равен 0.

Число 1000 нельзя представить в виде суммы четырехзначных чисел, поэтому ответ для числа 1000 аналогично равен 0.

# Вариант 58

Мэр города Урюпинска решил посадить на главной аллее города, которая проходит с запада на восток, голубые ели. Причем сажать ели можно не во всех местах, а только на специально оставленных при асфальтировании аллеи клумбах.

Как оказалось, голубые ели бывают M различных сортов. Для ели каждого сорта известна максимальная длина ее тени в течение дня в западном и в восточном направлении (Wi и Ei соответственно). При этом известно, что ели растут гораздо лучше, если в течение дня они не оказываются в тени других елей.

Координатная ось направлена вдоль аллеи с запада на восток.

По заданным координатам клумб вычислите максимальное число елей, которое можно посадить, соблюдая условие о том, что никакая ель не должна попадать в тень от другой ели.

Входные данные

Во входном файле INPUT.TXT записано сначала натуральное число M — количество сортов елей (1 ≤ M ≤ 100). Затем идет M пар чисел Wi, Ei, описывающих максимальную длину тени в западном и восточном направлении в течение дня для каждого сорта ели (числа Wi, Ei — целые, из диапазона от 0 до 30000). Далее идет натуральное число N — количество клумб, в которых можно сажать ели (1 ≤ N ≤ 100). Далее идет N чисел, задающих координаты клумб (координаты — целые числа, по модулю не превышающие 30000). Клумбы перечислены с запада на восток (в порядке возрастания их координат).

Примечание

Если на клумбе с координатой X мы посадили ель, максимальная тень которой в восточном направлении равна E, то все клумбы с координатами от X+1 до X+E–1 попадают в тень от этой ели, а клумба с координатами X+E — уже нет. Аналогично для тени в западном направлении.

Выходные данные

В выходной файл OUTPUT.TXT выведите число A — максимальное количество елей, которые удастся посадить.

Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3 1000 3 1 200 128 256 3 1 2 4 | 2 |

# Вариант 59

Гриша и Дима играют в следующую игру: они разложили однокопеечные монетки в стопки (в разных стопках может быть различное количество монет), а стопки расположили на столе перед собой в ряд слева направо. Затем Гриша и Дима по очереди делают ходы. На каждом ходе один из игроков берет слева несколько стопок, не меньше одной, но и не больше, чем перед этим взял его соперник. Первый игрок своим первым ходом берет не более K стопок. Игра заканчивается, когда стопок не остается.

Требуется написать программу, позволяющую вычислить, какое максимальное число монет может получить первый участник после окончания игры, если второй – тоже старается ходить так, чтобы получить как можно больше монет.

Входные данные

Входной файл INPUT.TXT состоит из одной строки, в которой записаны: число стопок N (1 ≤ N ≤ 180), за ним идут N чисел, задающих количество монет в стопках слева направо (количество монет в стопке – не менее 1 и не более 20000), а затем число K, ограничивающее количество стопок, которые первый игрок может взять на первом ходе (1 ≤ K ≤ 80). Все числа в строке разделены пробелом.

Выходные данные

В выходной файл OUTPUT.TXT необходимо вывести одно число – максимальное количество монет, которое заведомо может получить первый игрок, как бы ни играл второй.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3  4 9 1  3 | 14 |
| 2 | 4  1 2 2 7  3 | 5 |
| 3 | 5  3 4 8 1 7  2 | 18 |

# Вариант 60

Дана матрица N×N, заполненная положительными числами. Путь по матрице начинается в левом верхнем углу. За один ход можно пройти в соседнюю по вертикали или горизонтали клетку (если она существует). Нельзя ходить по диагонали, нельзя оставаться на месте.

Требуется найти максимальную сумму чисел, стоящих в клетках по пути длиной K (клетку можно посещать несколько раз).

Входные данные

Входной файл INPUT.TXT в первой строке содержит разделенные пробелом числа N и K. Затем идут N строк по N чисел в каждой – данные таблицы. Элементы матрицы – целые числа от 1 до 9999, 2 ≤ N ≤ 100, 1 ≤ K ≤ 2000.

Выходные данные

В выходной файл OUTPUT.TXT выведите одно число – максимальную сумму.

Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 5 7 1 1 1 1 1 1 1 3 1 9 1 1 6 1 1 1 1 3 1 1 1 1 1 1 1 | 21 |

# Вариант 61

Рассмотрим последовательности длины N, состоящие из 0 и 1. Требуется написать программу, которая по заданному натуральному числу N определяет количество тех из них, в которых никакие две единицы не стоят рядом.

Входные данные

Входной файл INPUT.TXT содержит число N (1 ≤ N ≤ 1000).

Выходные данные

В выходной файл OUTPUT.TXT выведите ответ на задачу.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 2 | 3 |
| 2 | 3 | 5 |

# Вариант 62

Рассмотрим числовую последовательность, первоначально состоящую из двух единиц: 1, 1. Далее на каждом последующем шаге будем вставлять между соседними элементами их сумму. В примере добавляемые элементы выделены:

|  |  |
| --- | --- |
| Номер шага | Последовательность |
| 0 | 1, 1 |
| 1 | 1, 2, 1 |
| 2 | 1, 3, 2, 3, 1 |
| 3 | 1, 4, 3, 5, 2, 5, 3, 4, 1 |

Требуется написать программу, которая подсчитает сумму членов последовательности, построенной за K шагов.

Входные данные

Входной файл INPUT.TXT содержит одно натуральное число K (0 ≤ K ≤ 100) – номер последнего шага.

Выходные данные

Выходной файл OUTPUT.TXT должен содержать одно натуральное число – сумму элементов последовательности, построенной за K шагов.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3 | 28 |
| 2 | 10 | 59050 |

# Вариант 63

Дано N целых чисел A1, A2, ..., AN. Требуется найти количество различных значений сумм вида k1A1 + k2A2 + ... + kNAN.

Входные данные

Входной файл INPUT.TXT в первой строке содержит число N, во второй - A1, A2, ..., AN через пробел. Ограничения: все числа целые, 1 ≤ N ≤ 500, 0 ≤ Ai ≤100, 0 ≤ ki ≤ 1.

Выходные данные

В выходной файл OUTPUT.TXT выведите количество различных значений сумм.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3 1 1 2 | 5 |
| 2 | 3 1 3 2 | 7 |
| 3 | 5 49 100 98 49 0 | 10 |

# Вариант 64

Автомобильные пробки случаются везде, даже в нашем небольшом городе. Дороги у нас имеют по две полосы в одном направлении, а автомобили только двух видов: легковые (в пробке занимают квадратное место 1×1 от ширины одной полосы) и грузовые (занимают прямоугольное место 1×2). Автомобилисты очень дисциплинированы: не становятся поперек полосы, не занимают чужую площадь, но и не оставляют свободных мест.

Требуется написать программу, которая определит количество различных автомобильных пробок длины N.

Входные данные

Входной файл INPUT.TXT содержит одно натуральное число N (N ≤ 1000).

Выходные данные

Выходной файл OUTPUT.TXT должен содержать найденное количество автомобильных пробок.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 2 | 4 |
| 2 | 3 | 9 |

# Вариант 65

Будем называть пару строк (α, β) подпарой строки γ, если γ = γ1αγ2βγ3 для некоторых (возможно пустых) строк γ1, γ2 и γ3. Длиной пары строк будем называть сумму длин составляющих ее строк: |(α, β)| = |α| + |β|.

По заданным двум строкам ξ и η найдите их длиннейшую общую подпару, то есть такую пару строк (α, β), что она является подпарой как ξ, так и η, и ее длина максимальна.

Входные данные

Входной файл INPUT.TXT содержит две непустые строки ξ и η, состоящие из маленьких букв английского алфавита. Длина каждой из строк не превышает 2000.

Выходные данные

В выходной файл OUTPUT.TXT выведите α на первой строке выходного файла и β на второй строке.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | abacabadabacaba acabacadacabaca | acaba abaca |
| 2 | ab bc | b |

# Вариант 66

Участник игры в морской бой размещает на игровом поле свои корабли. По правилам этой разновидности игры корабли могут быть только прямоугольниками любых размеров, не могут пересекаться и иметь общих граничных точек. Количество уже размещённых кораблей равно K. Последний корабль он хочет сделать максимально большим.

Входные данные

В первой строке входного файла INPUT.TXT записаны три числа N, M и K – количество клеток по вертикали, количество клеток по горизонтали и число уже выстроенных кораблей соответственно (1 ≤ N, M ≤ 100, 1 ≤ K ≤ 10). Следующие K строк содержат координаты K размещённых кораблей – 4 числа в каждой строке. 1-е и 2-е число – вертикальная и горизонтальная координаты левой верхней угловой клетки корабля, 3-е и 4-е число – вертикальная и горизонтальная координаты правой нижней угловой клетки корабля. Клетки поля нумеруются сверху вниз (от 1 до N) и слева направо (от 1 до M).

Выходные данные

В выходной файл OUTPUT.TXT выведите одно число – количество клеток в последнем корабле.

Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 8 7 3 1 1 2 2 3 5 3 7 4 2 4 3 | 21 |

# Вариант 67

В поисках пропитания большая дружная семья кроликов добрела до морковного поля. К сожалению, чуть раньше сюда же прибыла большая дружная семья голодных хомяков. Во избежание конфликта было решено собирать урожай по очереди. Поле представляет собой N грядок по M кустов; на каждом кусте растет некоторое количество морковок. Очередной собирающий стартует от любого куста первой грядки и движется к последней, переходя от одного куста к другому по следующему правилу: от куста номер K на грядке L можно перейти только на грядку L+1 к одному из трех кустов с номерами K-1, K, K+1 (конечно, если кусты с такими номерами есть). Каждый посещенный куст очищается от моркови полностью. Первым на сбор урожая выходит один из кроликов, следом идет хомяк, потом снова кролик и так до тех пор, пока на поле есть хоть одна морковка.

Кролики суетливы, поэтому они выбирают путь наиболее выгодный внешне: стартуют от самого богатого куста первой грядки, а из трех последующих вариантов всегда выбирают самый большой куст (при наличии нескольких кустов с одинаковым числом морковок выбирается куст с наибольшим номером). Хомяки, прибыв на поле раньше, успели составить подробную карту поля и поддерживают её в актуальном состоянии на основе оперативных данных о сборе урожая, поэтому они для каждого хомяка выбирают путь, позволяющий собрать максимальное количество морковок из возможных (при наличии нескольких вариантов с максимально возможным количеством морковок выбирается тот, где лексикографически больше последовательность номеров кустов в порядке посещения).

По известной карте поля определите, сколько моркови удалось собрать кроликам и хомякам по отдельности.

Входные данные

В первой строке входного файла INPUT.TXT содержится два целых числа N и M (1 ≤ N, M ≤ 100). Следом идут N строк, в каждой из которых M чисел Xi,j (0 ≤ Xi,j ≤ 10). Xi,j - количество морковок на j-ом кусте i-ой грядки.

Выходные данные

В выходной файл OUTPUT.TXT выведите через пробел два числа: количество морковок, собранных кроликами и хомяками, соответственно.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3 3 1 1 2 1 1 1 10 1 1 | 7 12 |
| 2 | 4 4 1 1 1 2 1 1 1 1 1 1 1 1 10 10 1 1 | 18 17 |

# Вариант 68

Дана последовательность чисел a1, a2, …, aN. За одну операцию разрешается удалить любое (кроме крайних) число, заплатив за это штраф, равный произведению этого числа на сумму соседних. Требуется удалить все числа, кроме крайних, с минимальным суммарным штрафом.

Например:

Начальная последовательность: 1 50 51 50 1.

Удаляем четвёртое число, штраф 50(51+1)=2600, получаем 1 50 51 1.

Удаляем третье число, штраф 51(50+1)=2601, получаем 1 50 1.

Удаляем второе число, штраф 50(1+1)=100.

Итого штраф 5301.

Входные данные

В первой строке входного файла INPUT.TXT записано одно число N (1 ≤ N ≤ 100) - количество чисел в последовательности.

Во второй строке находятся N целых чисел a1, a2, …, aN; никакое из чисел не превосходит по модулю 100.

Выходные данные

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно число - минимальный суммарный штраф.

Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 5 1 50 51 50 1 | 5301 |

# Вариант 69

Непустая строка, содержащая некоторое слово, называется палиндромом, если это слово одинаково читается как слева направо, так и справа налево. Пусть задана строка, в которой записано слово S, состоящее из N букв английского алфавита. Путем вычеркивания из этого слова некоторого набора символов, можно получить строку, которая будет палиндромом.

Требуется написать программу, с помощью которой можно определить, сколько существует способов вычеркивания из заданного слова некоторого (возможно пустого) набора символов, чтобы образованная таким образом строка была палиндромом. Способы, отличающиеся порядком вычеркивания символов, считаются одинаковыми.

Входные данные

В первой и единственной строке входного файла INPUT.TXT записано слово S, состоящее из N символов (1 ≤ N ≤ 30).

Выходные данные

В выходной файл OUTPUT.TXT выведите найденное число способов.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | AAA | 7 |
| 2 | BAOBAB | 22 |

# Вариант 70

Задан вес E пустой копилки и вес F копилки с монетами. В копилке могут находиться монеты N видов, для каждого вида известна ценность Pi и вес Wi одной монеты. Найти минимальную и максимальную суммы денег, которые могут находиться в копилке.

Входные данные

В первой строке входного файла INPUT.TXT находятся числа E и F, во второй - число N, в следующих N строках - по два числа, Pi и Wi. (1 ≤ E ≤ F ≤ 10000, 1 ≤ N ≤ 500, 1 ≤ Pi ≤ 50000, 1 ≤ Wi ≤ 10000).

Выходные данные

В выходной файл OUTPUT.TXT выведите два числа через пробел - минимальную и максимальную суммы. Если копилка не может иметь точно заданный вес при условии, что она наполнена монетами заданных видов, следует вывести "This is impossible.".

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 1000 1100 2 1 1 5 2 | 100 250 |
| 2 | 1000 1010 2 6 3 2 2 | 10 16 |
| 3 | 1000 2000 1 10 3 | This is impossible. |

# Вариант 71

Художественная гимнастика – это вид спорта, где всё познаётся в сравнении, здесь нельзя, как в беге или плавании, измерить результат спортсмена с точностью до сотой доли секунды. Поэтому на выступлениях оценки выступлениям дают судьи. При выставлении оценок судьи ориентируются не только на текущее выступление, но, безусловно, сравнивают текущее выступление с выступлениями, показанными ранее. Кроме того, учитывается сложность показанного упражнения и рейтинг спортсмена.

Каждый выход спортсмена описывается двумя числами: номер спортсмена в рейтинге (наиболее профессиональные спортсмены имеют наибольший номер), и номер исполненного упражнения (упражнения нумеруются, начиная с самых простых). Судья сравнивает каждый выход спортсмена с каждым из выполненных ранее выходов. Если в результате сравнения получается, что спортсмен с большим номером показал более простое упражнение, чем спортсмен с меньшим номером, судья удивляется. Следует учитывать, что один выход может удивить судью несколько раз. Один спортсмен может выполнить несколько выходов, так же, как и одно упражнение может быть показано несколькими спортсменами – но такие выходы в сравнении судью не удивляют. Спортсмен не исполняет уже показанное упражнение повторно. Требуется подсчитать, сколько раз за время выступлений будет удивлён судья.

Входные данные

Первая строка входного файла INPUT.TXT содержит количество спортсменов N (0 < N ≤ 250), количество упражнений M (0 < M ≤ 250) и количество выходов P. Следующие P строк содержат по два числа, описывающие выход спортсмена – номер спортсмена и номер упражнения.

Выходные данные

В выходной файл OUTPUT.TXT выведите, сколько раз был удивлен судья.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3 3 4 3 1 1 2 1 3 3 2 | 3 |
| 2 | 2 2 2 1 1 2 2 | 0 |

# Вариант 72

Заклинание Ауэрса представляет собой набор звуков, записываемых малыми буквами английского алфавита без пробелов. Сам по себе этот набор не обладает ни секретностью, ни магической силой и приведен во всех учебниках по белой магии в соответствующем разделе. Там же указан способ его применения:

а) разбить набор букв на палиндромы, то есть непустые слова, которые читаются одинаково как справа налево, так и слева направо;

б) произносить их по порядку, после каждого, взмахивая умклайдетом (в крайнем случае, щелкая пальцами);

в) магическая сила заклинания обратно пропорциональна числу палиндромов, на которые разбит набор букв.

Даже домовые знают, что это заклинание можно разбить на отдельные буквы и очень забавно наблюдать, как они, щелкая пальцами и выкрикивая отдельные звуки заклинания, выводят клопов. Однако истинную мощь заклинание обретает только тогда, когда оно разбито на наименьшее возможное число палиндромов. Знание этого разбиения доступно только самым великим магам, например Кристобалю Хозевичу Хунте.

Желая поднять свою квалификацию мага, Александр Привалов решил использовать современные средства для нахождения кратчайшего разбиения заклинания Ауэрса на палиндромы. Уже неделю он бьется над составлением переборной программы для своего Алдана, но пока все без толку. У него уже начала расти шерсть на ушах, помогите ему.

Входные данные

В единственной строке файла INPUT.TXT содержится непустой исходный набор, состоящий из маленьких букв английского алфавита без пробелов (количество символов в наборе не больше 70).

Выходные данные

В первой строке выходного файла OUTPUT.TXT необходимо указать наименьшее число k палиндромов, на которые разбивается набор. Эти k палиндромов нужно привести в следующих k строках в том порядке, в котором они входят в исходный набор и должны произноситься при использовании заклинания. Если имеется несколько минимальных разбиений, достаточно вывести любое из них.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | baobab | 4 b a o bab |
| 2 | aaaaa | 1 aaaaa |
| 3 | rarabar | 3 rar aba r |

# Вариант 73

В школу бальных танцев профессора Падеграса записались n учеников — мальчиков и девочек. Профессор построил их в один ряд, и хочет отобрать из них для первого занятия группу стоящих подряд учеников, в которой количество мальчиков и девочек одинаково. Сколько вариантов выбора есть у профессора?

Входные данные

В первой строке входного файла INPUT.TXT задано число n (1 ≤ N ≤ 106). Во второй строке задается описание построенного ряда из мальчиков и девочек — строка из n символов a и b (символ a соответствует девочке, а символ b — мальчику).

Выходные данные

В единственной строке выходного файла OUTPUT.TXT должно содержаться единственное число — количество вариантов выбора требуемой группы.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3 bab | 2 |
| 2 | 8 abbababa | 13 |

# Вариант 74

Вася недавно узнал, что такое циклическое k-расширение строки S. Его можно получить следующим образом: склеить k экземпляров строки S, а потом взять первые k символов результата.

Узнав это, Вася обрадовался, взял некоторую строку, и начал к ней применять описанную операцию, не запоминая, какое он каждый раз брал k.

Вам дана часть строки, получившейся у Васи. Ваша задача определить, не ошибся ли Вася в своих сложных преобразованиях, т. е., мог ли у него из первоначальной строки получиться ответ, содержащий данную строку в качестве подстроки.

Входные данные

В первой строке входного файла INPUT.TXT находится изначальная строка, которую Вася бережно записал перед тем, как приступить к своим действиям. Во второй строке находится подстрока результата, полученного Васей. Обе строки не пусты и по длине не превышают 5 000 символов. Строки могут состоять из больших и маленьких английских букв (с учетом регистра), а также цифр.

Выходные данные

В выходной файл OUTPUT.TXT выведите "NO", если можно точно сказать, что Вася ошибся, и "YES", если мог и не ошибиться.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | abc abc | YES |
| 2 | abcd bcabc | YES |
| 3 | abcabc abcA | NO |

# Вариант 75

Задан шаблон, состоящий из круглых скобок и знаков вопроса. Требуется определить, сколькими способами можно заменить знаки вопроса круглыми скобками так, чтобы получилось правильное скобочное выражение.

Входные данные

Единственная строка входного файла INPUT.TXT содержит заданный шаблон длиной от 1 до 80 символов.

Выходные данные

Выведите в выходной файл OUTPUT.TXT искомое количество способов. Исходные данные будут таковы, что это количество не превзойдет 2×109.

Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | ????(? | 2 |

# Вариант 76

Играют двое. Задаётся какая-то дата 2008 года. Каждый игрок на своём ходе называет более позднюю дату, увеличивая на 1 или 2 либо день в месяце, либо месяц, но не то и другое сразу. При этом сочетание дня и месяца должно оставаться датой. Игрок, назвавший 31 декабря, проигрывает. Оба играют наилучшим образом. Исходя из заданной даты вывести, кто выиграет.

Входные данные

В первой строке входного файла INPUT.TXT находятся числа, обозначающие день и месяц. Месяц указывается от 1 до 12, день от 1 до числа дней в месяце, дата «31 декабря» отсутствует во входных данных.

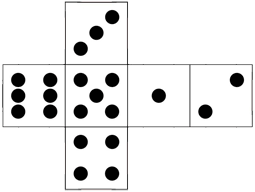
Выходные данные

В выходной файл OUTPUT.TXT выведите 1, если выигрывает первый (начинающий) игрок, или 2 – в противном случае.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 30 12 | 2 |
| 2 | 29 12 | 1 |
| 3 | 29 11 | 2 |

# Вариант 77



В последнее время настольные игры стали очень популярны. В нашу жизнь возвращаются как некогда забытые, так и новые увлекательные игры. К вам попала совершенно новая, уникальная настольная игра. Для игры нужен игральный кубик (его развертка приведена на рисунке) и прямоугольное игровое поле, разбитое на клетки. В каждой клетке поля написано целое число.

Играть могут несколько человек по следующим правилам:

Первым ходом игрок ставит кубик в левую верхнюю клетку поля на любую грань.

Каждым следующим ходом игрок переходит на соседнюю справа или снизу клетку, перекатывая кубик на любую из 4 соседних граней.

За каждый ход игрок получает K\*A очков, где A - число, записанное в текущей клетке поля, K - цифра на той грани кубика, которой он стоит на игровом поле.

Игрок ходит, пока не окажется в нижней правой клетке поля. Тогда очки за все его ходы суммируются.

Когда один игрок закончил ходить, начинает ходить второй игрок и т.д.

Победитель - игрок, набравший больше всех очков.

По заданному игровому полю определите максимальную сумму очков, которую можно получить, играя по указанным правилам.

Входные данные

Первая строка входного файла INPUT.TXT содержит разделенные пробелом числа N и M - размеры игрового поля (1 ≤ N×M ≤ 105). Далее идет N строк по M чисел, разделенных пробелами - числа, записанные в клетках игрового поля. Все числа по модулю не превышают 103.

Выходные данные

В выходной файл OUTPUT.TXT выведите целое число – наибольшее количество очков, которые можно получить при оптимальной игре.

Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3 3 1 -2 3 -4 5 -6 7 -8 9 | 78 |

# Вариант 78

Закончив университет, Петя решил заняться разработкой web-приложений. Получив первый заказ, Петя посчитал разумным на время выполнения заказа обеспечить себя доступом в Интернет. Провайдер, с которым Петя заключил контракт, обеспечивает доступ в Интернет только посредством сервис-карт. Когда Петя пришел в местное почтовое отделение за их покупкой, то оказалось, что в продаже имеется N сервис-карт.

Каждая сервис-карта характеризуется тремя числами Bi, Ei, Si, где Si – цена карты, а Bi и Ei – это начало и окончание промежутка времени её действия, т.е. сервис-карта позволяет получить доступ в Интернет в любой момент времени t такой, что Bi ≤ t ≤ Ei. Время отсчитывается от некоторого фиксированного момента в прошлом.

Петя решил, что будет работать над выполнением заказа, начиная с момента времени B и заканчивая моментом времени E. Основная проблема состоит в том, что Петя не богат, поэтому он хочет на время выполнения заказа обеспечить себя «стабильным» Интернетом и потратить на это как можно меньшую сумму денег. Будем говорить, что данное множество сервис-карт обеспечивает Пете «стабильный» Интернет, если для любого момента времени t, такого, что B ≤ t ≤ E, найдется хотя бы одна сервис-карта из данного множества, которая позволяет получить доступ в Интернет в этот момент времени.

Ваша задача состоит в том, чтобы определить минимальную сумму денег, необходимую Пете для приобретения множества карт, которое обеспечит ему «стабильный» Интернет на время выполнения заказа.

Входные данные

В первой строке входного файла INPUT.TXT находится одно число N. Во второй строке два числа B и E, разделенные пробелом, где B – момент начала промежутка времени, в который Петя будет работать над выполнением заказа, а E – момент окончания.

Следующие N строк описывают сервис-карты, i-ая из этих строк описывает i-ю сервис-карту и содержит три числа Bi, Ei и Si, разделенные одиночными пробелами.

Ограничения: все числа натуральные, 1 ≤ N ≤ 105, 1 ≤ Bi < Ei ≤ 109, 1 ≤ B < E ≤ 109, 1 ≤ Si ≤ 20 000. Существует хотя бы одно множество карт, которое обеспечивает «стабильный» Интернет на время выполнения заказа

Выходные данные

Выходной файл OUTPUT.TXT должен содержать одно число, равное минимальной сумме денег, необходимой Пете для приобретения множества сервис-карт, которое обеспечит ему «стабильный» Интернет на время выполнения заказа.

Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 7 10 30 9 14 10 13 19 18 14 18 16 18 24 14 24 30 9 17 2005 24 15 20 14 | 49 |

# Вариант 79

|  |  |
| --- | --- |
| В нашем зоопарке появился заяц. Его поместили в клетку, и чтобы ему не было скучно, директор зоопарка распорядился поставить в его клетке лесенку. Теперь наш зайчик может прыгать по лесенке вверх, перепрыгивая через ступеньки. Лестница имеет определенное количество ступенек N. Заяц может одним прыжком преодолеть не более К ступенек. Для разнообразия зайчик пытается каждый раз найти новый путь к вершине лестницы. Директору любопытно, сколько различных способов есть у зайца добраться до вершины лестницы при заданных значениях K и N. Помогите директору написать программу, которая поможет вычислить это количество. Например, если K=3 и N=4, то существуют следующие маршруты: 1+1+1+1, 1+1+2, 1+2+1, 2+1+1, 2+2, 1+3, 3+1. Т.е. при данных значениях у зайца всего 7 различных маршрутов добраться до вершины лестницы. |  |
|  |  |

Входные данные

В единственной строке входного файла INPUT.TXT записаны два натуральных числа K и N (1 ≤ K ≤ N ≤ 300). К - максимальное количество ступенек, которое может преодолеть заяц одним прыжком, N – общее число ступенек лестницы.

Выходные данные

В единственную строку выходного файла OUTPUT.TXT нужно вывести количество возможных вариантов различных маршрутов зайца на верхнюю ступеньку лестницы без ведущих нулей.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 1 3 | 1 |
| 2 | 2 7 | 21 |
| 3 | 3 10 | 274 |

# Вариант 80

Однажды глава семейства заказал фотографию своей большой семьи, состоящей из N человек, возраст которых 1 год, 2 года, …, N-1 лет и N лет. На фотографии должны присутствовать все родственники, и для этого они должны расположиться в один ряд. Сначала было решено расположить родственников по старшинству, начиная с самого младшего. Но фотограф сказал, что, возможно, на фото это будет выглядеть неестественно. Тогда было решено использовать следующее размещение:

слева сидит ребенок возрастом в 1 год

разность возрастов двух соседних родственников не превышает 2 года

Действительно, на фотографии, таким образом, все будут все равно выглядеть, будто расположенные по старшинству (ведь среди людей возрастом, к примеру, 25 и 27 лет не так легко определить старшего). Способов такой посадки существует, понятно, несколько. Фотограф снял все такие способы. Сколько же фотографий получилось в итоге?

Входные данные

Во входном файле INPUT.TXT содержится число N (1 ≤ N ≤ 55) – количество членов большой семьи.

Выходные данные

Выходной файл OUTPUT.TXT должен содержать искомое число фотографий.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 4 | 4 |
| 2 | 7 | 14 |

# Вариант 81

Математики – люди, весьма ценящие красоту. Они умеют находить ее даже в объектах, к которым, по мнению большинства, это понятие даже не может быть применимо. Например, недавно один математик, занимавшийся изучением последовательностей натуральных чисел, заметил, что некоторые из последовательностей отличаются необычайной красотой. Особенно красивыми он посчитал последовательности a1, ... , an длины n из чисел от 1 до k, обладающие следующими свойствами:

a1=1;

если ai=M , i > 1, то найдется j < i : aj=M-1, либо M=1.

Примером такой последовательности для n = 5, k = 3 является «1, 1, 2, 3, 1», а последовательность «1, 3, 1, 4» не является особенно красивой ни для каких n и k.

Ваша задача состоит в нахождении числа особенно красивых последовательностей для заданных n и k.

Входные данные

Входной файл INPUT.TXT содержит два целых числа: n и k (1 ≤ n, k ≤ 500).

Выходные данные

В выходной файл OUTPUT.TXT выведите ответ на задачу.

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3 3 | 5 |
| 2 | 7 6 | 876 |

# Вариант 82

Дана цепочка из N символов, состоящая из прописных английских букв. Необходимо пройти с первого символа цепочки до последнего символа, прыгая не более чем на K символов. Стоимость прыжка, при котором символ не меняется, равна 0, а стоимость прыжка на другой символ равна 1.

Требуется написать программу, которая вычислит наименьшую стоимость перехода с первого на последний символ.

Входные данные

Входной файл INPUT.TXT содержит в первой строке два целых числа: длина цепочки N (2 ≤ N ≤ 105) и максимальная длина прыжка K (1 ≤ K < N). Во второй строке содержится цепочка из N английских букв.

Выходные данные

Выходной файл OUTPUT.TXT должен содержать одно число – минимальную стоимость перехода.

Пример

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 10 2 ABABBCACBC | 2 |

# Вариант 83

Алеша Попович и Добрыня Никитич сражаются со стаей двух- и трехголовых драконов. Они по очереди взмахивают мечами, и одним махом могут отрубить любое натуральное число голов (по своему желанию), но только у одного дракона. Отрубивший последнюю голову у последнего дракона получает в жены прекрасную принцессу.

Кто из богатырей (начинающий или второй) может получить в жены принцессу независимо от действий другого?

Входные данные

Во входном файле INPUT.TXT записано два числа N и M — количество двух- и трехголовых драконов соответственно (оба числа целые из диапазона от 0 до 100, N+M>0).

Выходные данные

В выходной файл OUTPUT.TXT выведите число 1 или 2 определяющее, кто из богатырей имеет все шансы получить в жены принцессу (1 — тот, кто начинает, 2 — второй).

Примеры

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 2 2 | 2 |
| 2 | 1 2 | 1 |