Оглавление

9-11 класс

Задача 1. Драгоценные камни

Имя входного файла:	gems.in
Имя выходного файла:	gems.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 Кб

Гномы, жители страны Мории, добывают драгоценные камни трех видов — поделочные (вид A), полудрагоценные (вид B) и драгоценные (вид C). Хранят их гномы в одинаковых коробках, каждую из которых наполняют только одним видом камней. Заполненные коробки гномы складывают в огромных залах Мории одну на другую так, что получаются вертикальные стопки.

Самые «плохие» стопки гномы отдают оркам. Стопка считается плохой, если в ней подряд лежат более одной коробки, заполненной камнями вида A. Стопка считается «хорошей», если она не является «плохой».

От Вас **требуется** написать программу, которая для заданного количества коробок N будет определять число возможных «хороших» стопок.

Формат входного файла

В единственной строке входного файла записано одно натуральное число $1 \le N \le 20$.

Формат выходного файла

В единственной строке входного файла записано одно натуральное число — количество «хороших» вариантов формирования стопки.

Пример входного и выходного файлов

gems.in	gems.out
2	8

Примечание к примеру:

В примере из условия среди стопок длины 2 бывают «хорошие» стопки видов AB, AC, BA, BB, BC, CA, CB и CC. Стопки типа AA являются «плохими».

Решение

Задача относится к теме «Динамическое программирование». Для решения можно использовать двумерный массив a[n][k], где будет храниться количество способов для последовательности длины n расставить элементы, причем последний элемент имеет тип k (т.е. равен 1, 2 или 3). Тогда a[1][1], a[1][2] и a[3][3] будут равны 1, а формула расчета следующих элементов будут такими:

a[n][1] = a[n][2] = a[n - 1][1] + a[n - 1][2] + a[n - 1][3]

a[n][3] = a[n - 1][1] + a[n - 1][2];

Однако, можно заметить, что для подсчета n-го элемента нужны всего 3 предыдущих, рассчитанных для n-1-го элемента: [n - 1][1] , [n - 1][2], [n - 1][3]. Это позволяет отказаться от использования массива и обойтись четырьмя k0, k1, k2 и k соответственно.

Решение на PascalABC:

```
var k0,k1,k2,k:integer;
i,n:integer;
begin
  assign(input, 'gems.in');
  reset(input);
  readln(n);
  k0:=1;
  k1:=1;
  k2:=1;
  k:=3;
  for i:=2 to n do
  begin
    k0:=k1+k2;
    k1:=k;
    k2:=k;
    k:=k0+k1+k2;
  end;
  assign(output,'gems.out');
  rewrite(output);
  write(k);
  close(input);
  close(output);
end.
```

Решение на Python:

```
n = int (input())
k0 = 1
k1 = 1
k2 = 1
k = 3
for i in range(2, n + 1):
    k0 = k1 + k2
    k1 = k
    k2 = k
    k = k0 + k1 + k2
print (k)
```

Задача 2. Город будущего

Имя входного файла:	city.in
Имя выходного файла:	city.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 Кб

В городе будущего все дома – это небоскребы. Всего в городе *M* улиц, которые пересекаются и дают всего *N* перекрестков. Для организации эффективного движения необходимо, чтобы перед каждым перекрестком вдоль улицы было размещено лазерное устройство, отслеживающее поток транспортных средств, движущихся к перекрестку. Каждое такое устройство (лазер) работает только в одну сторону – от перекрестка вдоль улицы. Движение по всем улицам двустороннее. Любые два перекрестка соединены только одной улицей. Нет улиц от k-го перекрестка до него самого.

Итак, от Вас **требуется** написать программу, вычисляющую для каждого перекрестка количество лазеров, которые необходимо на нем поставить властям города.

Формат входных данных

Первая строка входного файла содержит два натуральных числа *N* и *M* ($0 < N \le 100$, $0 \le M \le N*(N-1)/2$).

Каждая из следующих M строк содержит разделенные пробелом два натуральных числа *i* и *j* ($1 \le i,j \le N$), которые означают, что перекрестки *i* и *j* соединены улицей.

Формат выходных данных

В первой строке выходного файла нужно вывести *N* чисел: *i*-ое число означает количество лазеров на *i*-ом перекрестке.

Пример входного и выходного файлов

city.in	city.out
7 10 5 1 3 2 7 1 5 2 7 4 6 5 6 4 7 5 2 1 5 3	3 3 2 2 5 2 3

Решение

Данная задача относится к теме «Графы» и затрагивает основные понятия данной темы. Представим город графом, в котором перекрестки – это вершины, ребра – это улицы, их содиняющие. Заведем в программе массив размером максимум 50 000 элементов. Каждый элемент этого

массива будет хранить количество лазеров, которые нужно установить на перекрестке так, чтобы каждый лазер контролировал движение к перекрестку на «своей» улице.

Поскольку улицы с двухсторонним движением, то каждое ребро дает по два лазера – на каждом перекресте, завершающем участок, навстречу движению транспорта. Сразу при вводе будем увеличивать на 1 соответствующие элементы массива.

Решение на PascalABC:

```
program city;
var
  n, i, m, x, y: integer;
  ans: array[1..50000] of integer;
begin
  assign(input, 'city.in');
  reset(input);
  readln(n, m);
  for i := 1 to m do
  begin
    readln(x,y);
    ans[x] += 1;
    ans[y] += 1;
  end;
  assign(output, 'city.out');
  rewrite(output);
  for i := 1 to n do write(ans[i],' ');
  close(input);
  close(output);
end.
```

Решение на Python:

```
n, m = map(int, input().split())
a = [0 for i in range(n)]
for i in range(m):
    x = list(map(int, input().split()))
    a[x[0]-1] += 1
    a[x[1]-1] += 1
for j in a:
    print(j, end=' ')
```

Задача 3. Площадка перед домом

Имя входного файла:	tile.in
Имя выходного файла:	tile.out
Максимальное время работы на одном тесте:	2 секунды
Максимальный объем используемой памяти:	64 Кб

Иван Петрович купил для укладки дорожек на своем участке некоторое количество тротуарной плитки. После того, как все дорожки были готовы, осталось еще n плиток одинакового размера: w — ширина и h — высота. Иван Петрович решил уложить на своем участке перед домом еще и квадратную площадку так, чтобы она занимала как можно меньше места на участке. Для того, чтобы площадка была красивая, каждая плитка должна быть размещена строго в прямоугольнике размером w на h. Плитка нельзя поворачивать на 90 градусов и размещать так, чтобы они накладывались одна на другую, а то площадка будет с ухабами.

От Вас **требуется** написать программу, которая вычислит минимальный размер стороны площадки, которая получится перед домом Ивана Петровича.

Формат входных данных

Входной файл содержит одну строку, в которой находятся разделенный пробелом три натуральных числа: w, h, n ($1 \le w, h, n \le 10^9$).

Формат выходных данных

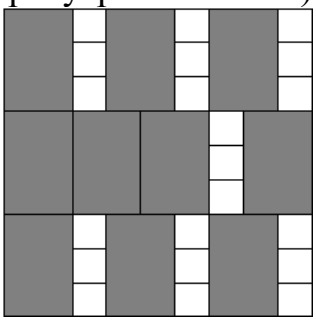
Одно натуральное число – минимальный размер стороны площадки.

Примеры входных и выходных файлов

tile.in	tile.out
2 3 10	9

tile.in	tile.out
1 1 1	1

Рисунок, поясняющий первый пример (закрашенные прямоугольники – это тротуарные плитки):



Решение

Данная задача относится к теме «Бинарный поиск по ответу». Из-за больших ограничений на n,w,h линейный перебор возможной длины стороны квадратной площадки не будет проходить по времени. Используем бинарный поиск по ответу.

Анализ условия показывает, что размеры площадки лежат в пределах от min(w,h) до n * max(w,h). За O(1) легко проверить, поместятся ли все плитки в квадрат со стороной a (n <= (a / w) * (a / h)). Считаем, что искомый ответ лежит в интервале от Min до Max. Теперь проверим, удовлетворяет ли условию квадрат со стороной Mid = (Min + Max) / 2. Если да, то Max = Mid, иначе Min = Mid+1.

Выполняем эту процедуру до тех пор, пока не сойдутся Min и Max. Мы будем всё точнее получать возможный диапазон ответа (каждый раз область поиска уменьшается в 2 раза). Когда Min станет равно Max, выведем значение Min и прекратим работу программы. Сложность такого решения: O(log(n*max(w,h))).

Решение на PascalABC:

```
var
    l,r,w,h,n,m,xx:int64;
begin
  assign(input,'tile.txt');
  reset(input);
  readln(w,h,n);
  close(input);
  assign(output,'tile.txt');
  rewrite(output);
  l:=min(w,h);
  r:=n*max(w,h);
  while (l<>r) do
  begin
    m:=(l+r) div 2;
    xx:=(m div w)*(m div h);
    if (n<=xx)
    then r:=m
    else l:=m+1;
  end;
  writeln(l);
  close(output);
end.
```

Решение на Python:

```
w, h, n = map(int, input().split())
if (w > h):
    l = h
    r = w
else:
    l = w
    r = h
r *= n
while (l != r):
    m = (l+r) // 2
```

```
    xx = (m // w) * (m // h)
    if (n<=xx): r = m
    else: l = m + 1
print(l)
```

Задача 4. Сложный шифр

Имя входного файла:	cipher.in
Имя выходного файла:	cipher.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 Кб

Для передачи сообщения по открытому каналу связи шифровальщики придумали сложный шифр. Шифр состоит из N различных чисел, записанных в одну строку через пробелы. Дешифрованное сообщение состоит из K чисел, которые являются номерами исходных чисел, разбивающих исходное сообщение на K частей (подпоследовательностей), так что сумма минимумов каждой подпоследовательности будет максимальна.

Итак, от Вас **требуется** написать программу, которая выполняет дешифровку заданного шифра.

Формат входных данных

Первая строка входного файла содержит два разделенных пробелом натуральных числа N и K ($1 \le K < N \le 300$).

Вторая строка входного файла содержит зашифрованное сообщение – N разделенных пробелом целых различных натуральных чисел $a_1, a_2, a_3 \ldots a_n$ ($1 \le a_i \le 10^9$).

Формат выходных данных

Выходной файл в первой строке содержит максимальную сумму K слагаемых, которые являются минимумами каждой части разбиения последовательности на K частей.

Во второй строке выходного файла содержится дешифрованное сообщение из разделенных пробелом K натуральных чисел – номеров чисел в последовательности, которые завершают каждую из подпоследовательностей зашифрованного сообщения.

Пример входного и выходного файлов

cipher.in	cipher.out
10 5 1 10 2 8 9 3 5 4 7 6	27 3 4 5 8 10

Пояснения к примеру::

В примере зашифрованное сообщение из 10 чисел разбивается на 5 частей:

{1, 10,2}, {8},{9},{ 3 5 4}, {7 6}

Искомая сумма равна 1+8+9+3+6=27

Решение

Данная задача относится к теме «Динамическое программирование». Обозначим как rmq[i][j] минимальный элемент в массиве между i-й и j-й позицией, включительно. Для вычисления rmq[i][j] можно использовать следующие формулы: rmq[i][i] = 1 rmq[i][j] = min(rmq[i][j – 1], a[j]) Обозначим как d[r][c] максимальную сумму минимумов, которую можно получить на отрезке с первого по r-й элемент, если разбить его на с частей.

Тогда d[r][c] = min(d[r – k][c – 1] + rmq[r – k + 1][r]) для всех k от 1 до r Ответ содержится в элементе d[n][k]. Для восстановления самого разбиения необходимо также запоминать, для какого k достигается минимум (это позволит восстановить ответ). Следует также обратить внимание на возможность переполнения 32-битного типа данных в этой задаче. Использование 32-битный тип данных не позволит набрать полный баллза задачу.

Решение на PascalABC:

```
var
  i, j, n, k, l, r, cur, asize: int64;
  a, ans: array[1..300] of int64;
  rmq, d, pr: array[0..300, 1..300] of int64;
begin
  read(n, k);
  for i := 1 to n do read(a[i]);
  for l := 1 to n do
    for r := l to n do
    begin
      cur := a[l];
      for i := l to r do cur:= min(cur, a[i]);
      rmq[l, r] := cur
    end;
  for i := 1 to n do d[0, i]:=rmq[1, i];
  for i := 1 to k - 1 do
    for j := 1 to n do
      for l := 1 to j do
        if d[i, j] < d[i - 1, l] + rmq[l + 1, j]
        then
        begin
          d[i,j]:=d[i-1,l]+rmq[l+1,j];
          pr[i, j] := l
        end;
  writeln(d[k - 1][n]);
  i:=k-1;
  j := n;
  asize := 0;
  while pr[i, j] <> 0 do
  begin
    inc(asize);
    ans[asize] := pr[i, j];
    j := pr[i, j];
    dec(i)
  end;
  for i:=asize downto 1 do write(ans[i],' ');
  writeln(n)
```

```
end.
```

Решение на Python:

```
n, k = map(int, input().split())
a = list(map(int, input().split()))

rmq = [[-1] * n for i in range(n)]
for i in range(n):
    rmq[i][i] = a[i]

for i in range(n):
    for j in range(i + 1, n):
        rmq[i][j] = min(rmq[i][j - 1], a[j])

d = [[-1] * (k + 1) for i in range(n + 1)]
f = [[-1] * (k + 1) for i in range(n + 1)]
d[0][0] = 0
for i in range(n):
    for j in range(k):
        if d[i][j] != -1:
            for l in range(i + 1, n + 1):
                v = d[i][j] + rmq[i][l - 1]
                if v > d[l][j + 1]:
                    d[l][j + 1] = v
                    f[l][j + 1] = i
print(d[n][k])
a = []
c = n
p = k
while c > 0:
    c = f[c][p]
    p -= 1
    if c != 0:
        a.append(c)
a.reverse()
a.append(n)
print(" ".join([str(x) for x in a]))
```