ОЛИМПИАДНЫЕ ЗАДАЧИ

Динамическое программирование

Григорьева А.В.

Задача «Возрастающая подпоследовательность»

Даны N целых чисел $X_1, X_2, ..., X_N$. Требуется вычеркнуть из них минимальное количество чисел так, чтобы оставшиеся шли в порядке возрастания.

Ограничения: $1 \leq N \leq 10\,000$, $1 \leq X_i \leq 60\,000$, время 4 с.

Ввод из файла incseq.in. В первой строке находится число N. В следующей строке — N чисел через пробел.

Вывод в файл incseq.out. В первой строке выводится количество невычеркнутых чисел, во второй — сами невычеркнутые числа через пробел в исходном порядке. Если вариантов несколько, вывести любой.

Пример

Ввод

```
6
2 5 3 4 6 1
```

Вывод

```
4
2 3 4 6
```

1 5 3 7 1 4 10 15

Пример

Для каждого члена исходной последовательности нужно вычислить максимальную длину возрастающей подпоследовательности, оканчивающейся этим элементом. Для примера

2 5 3 4 6 1

эта характеристика будет выглядеть так:

1 2 2 3 4 1

Решение

Будем вычислять характеристики для всех членов последовательности от 1-го до N-го по порядку и заносить полученные результаты в массив.

Пусть известны характеристики для всех членов последовательности от 1-го до $(i - 1)$-го и нужно узнать ее для i-го. Последовательность длиной 1 из одного (i-го) элемента всегда можно построить. Пусть можно построить последовательность длиной не меньше двух. Тогда какой-то из элементов от 1-го до $(i - 1)$-го будет предпоследним. Очевидно, что предпоследним может быть любой элемент, меньший i-го. А наилучшая характеристика у i-го элемента получится, если взять предыдущий элемент с максимальной характеристикой.

Итак, чтобы получить максимальную длину подпоследовательности, кончающейся i-м элементом, нужно выбрать максимум из длин подпоследовательностей элементов от 1-го до $(i - 1)$-го, меньших i-го, и добавить единицу. Если меньших элементов не существует, получится длина 1.

Очевидно, длина искомой подпоследовательности равна максимуму из найденных длин каждого элемента. А соответствующий максимуму элемент является последним в последовательности, которую нужно вывести. Предыдущие элементы восстанавливаются следующим образом. Идем по массивам от индекса максимума к 1. Находим элемент, меньший максимума, с длиной последовательности, кончающейся им, на 1 меньше максимальной длины. Продолжаем идти к меньшим индексам, находим еще меньший элемент с длиной последовательности, меньшей еще на 1, и так до тех пор, пока не закончится массив.

Детали реализации

- Для хранения исходных значений нужен массив из элементов типа word. В переменные типа integer (возможные значения от –32 768 до 32 767) исходные данные не поместятся. Диапазон чисел, представимых типом word, — от 0 до 65 535, то есть для этой задачи подходит. Почему не рекомендуется использовать тип longint? Диапазон значений, представимых типом longint, — от –2 147 483 648 до 2 147 483 647 — подходит. Но переменные типа longint занимают 4 байта (в отличие от 2 байтов для integer и word), поэтому работа с ними в Турбо Паскале происходит медленнее, чем с переменными типа integer. А работа с данными типа word происходит со скоростью работы с данными типа integer.
- Согласно приведенному алгоритму, элементы искомой подпоследовательности будут найдены в порядке, обратном тому, в котором их нужно вывести. Чтобы не объявлять дополнительный массив, при последнем проходе можно числа, не входящие в последовательность, заменять нулями (во входных данных 0 встретиться не может, там числа только от 1 до 60 000). А потом пройти по массиву от 1-го до N-го элемента и вывести все ненулевые числа.

Модификация решения

Чтобы избежать дополнительного прохода по массиву, можно находить другую величину — максимальную длину возрастающей подпоследовательности, *начинающейся* этим элементом. Тогда эта характеристика находится для i-го элемента на основании характеристик элементов с $(i + 1)$-го по N-й.

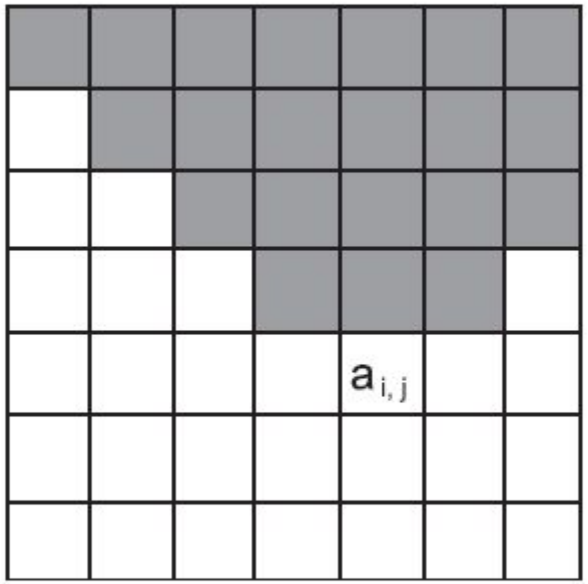
Сдать можно как задачу №613

- http://informatics.mccme.ru/mod/statements/view3.php?chapterid=613#1

Задача «Таблица»

Рассмотрим прямоугольную таблицу размером $n \times m$. 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 a_{ij} число, стоящее на пересечении i-ой строки и j-ого столбца. Первая строка таблицы заполняется заданными числами — a_{11}, a_{12}, $\ldots$, a_{1m}. Затем заполняются строки с номерами от 2 до n. Число a_{ij} вычисляется как сумма всех чисел таблицы, находящихся в «треугольнике» над элементом a_{ij}. Все вычисления при этом выполняются по модулю r.



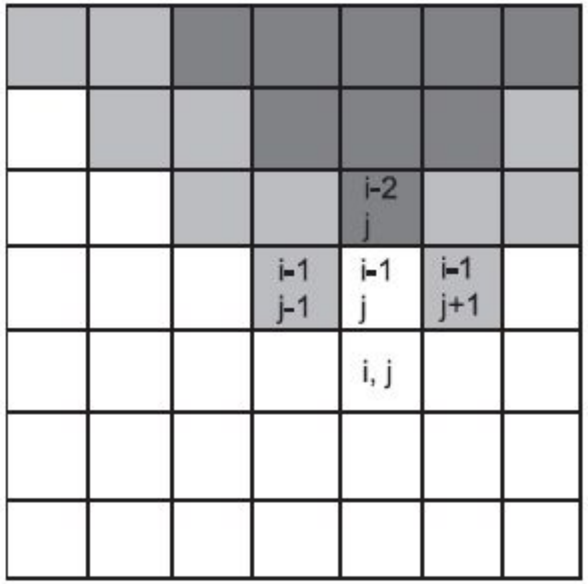
Например, если таблица состоит из трёх строк и четырёх столбцов, и первая строка состоит из чисел $2, 3, 4, 5$, а $r = 40$ то для этих исходных данных таблица будет выглядеть следующим образом (взятие по модулю показано только там, где оно приводит к изменению числа):

2	3	4	5
$5 = 2 + 3$	$9 = 2 + 3 + 4$	$12 = 3 + 4 + 5$	$9 = 4 + 5$
$23 = 2 + 3 + 4 + 5 + 9$	$0 = (2 + 3 + 4 + 5 + 5 + 9 + 12) \bmod 40 = 40 \bmod 40$	$4 = (2 + 3 + 4 + 5 + 9 + 12 + 9) \bmod 40 = 44 \bmod 40$	$33 = 3 + 4 + 5 + 12 + 9$

Требуется написать программу, которая по заданной первой строке таблицы (a_{11}, a_{12}, ..., a_{1m}), вычисляет последнюю строку, как описано выше.

Первый способ

База динамики (первая строка) нам явно задана. Рассмотрим два варианта вычисления значения в ячейке с использованием уже вычисленных данных.



В первом варианте нам достаточно завести массив B в котором мы будем накапливать ответы для ячеек. Очевидно, первая строка массива B совпадает с первой строкой массива A. Рассмотрим, как можно вычислить значение ячейки $B[i, j]$. Треугольник с вершиной в точке (i, j) состоит из треугольников с вершинами в точках $(i-1, j-1)$, $(i-1, j+1)$ а также из точек $(i-1, j)$ и (i, j). В то же время, треугольник с вершиной в точке $(i-2, j)$ перекрывается дважды, что приводит к двукратному суммированию значений в этом треугольнике. Избежать этого очень просто — достаточно один раз вычесть уже подсчитанную для этого треугольника сумму. Таким образом, в итоге получим следующую формулу $B[i][j] = B[i-1][j-1] + B[i-1][j+1] - B[i-2][j] + A[i-1][j] + A[i][j]$. Пользуясь этой формулой легко подсчитать массив, содержащий ответ.

Второй способ

С диагоналями. Нужен, чтобы хранить не 3 строки одной таблицы (B), а по две строки трех таблиц (L, R, B)

можно создать два дополнительных массива L и R, таких, что ячейка $L[i][j]$ будет содержать сумму всех элементов на диагонали, проведённой от точки (i, j) вверх и влево (для массива R диагональ проводится вверх и вправо). Первая строка массивов L и R будет совпадать с первой строкой массива A.

L

2	3	4	5

R

2	3	4	5

B

2	3	4	5

Что должно получиться

2	3	4	5
5	9	12	9
23	40	44	33
…	…	…	…

Первую строку заполняем первой строкой из A

Заполняем вторую строку B по-честному

L

2	3	4	5

R

2	3	4	5

B

2	3	4	5
2+3	2+3+4	3+4+5	4+5

Что должно получиться

2	3	4	5
5	9	12	9
23	40	44	33
…	…	…	…

Первую строку заполняем первой строкой из A

Заполняем вторую строку B по-честному

L

2	3	4	5
5	11	15	13

R

2	3	4	5
8	13	17	9

B[i, j] = 2*B[i-1,j] + L[i-1,j-1] + R[i+1,j+1]
L[i, j] = L[i-1,j-1] + B[i,j]
R[i, j] = R[i-1,j+1] + B[i,j]

B

2	3	4	5
5	9	12	9

Что должно получиться

2	3	4	5
5	9	12	9
23	40	44	33
…	…	…	…

Заполняем вторую строку L и R по формулам

Теперь можно и третью строку B заполнить

L				
	2	3	4	5
L	5	11	15	13

R				
	2	3	4	5
R	8	13	17	9

B[i, j] = 2*B[i-1,j] + L[i-1,j-1] + R[i+1,j+1]
L[i, j] = L[i-1,j-1] + B[i,j]
R[i, j] = R[i-1,j+1] + B[i,j]

B				
	2	3	4	5
B	5	9	12	9
	2*5+ 13			

Что должно получиться

2	3	4	5
5	9	12	9
23	40	44	33
…	…	…	…

Теперь можно и третью строку B заполнить

L

2	3	4	5
5	11	15	13

R

2	3	4	5
8	13	17	9

B[i, j] = 2*B[i-1,j] + L[i-1,j-1] + R[i+1,j+1]
L[i, j] = L[i-1,j-1] + B[i,j]
R[i, j] = R[i-1,j+1] + B[i,j]

B

2	3	4	5
5	9	12	9
23	2*9 + 5 + 17		

Что должно получиться

2	3	4	5
5	9	12	9
23	40	44	33
…	…	…	…

Теперь можно и третью строку B заполнить

L

2	3	4	5
5	11	15	13

R

2	3	4	5
8	13	17	9

B[i, j] = 2*B[i-1,j] + L[i-1,j-1] + R[i+1,j+1]
L[i, j] = L[i-1,j-1] + B[i,j]
R[i, j] = R[i-1,j+1] + B[i,j]

B

2	3	4	5
5	9	12	9
23	40	12*2 + 11 + 9	

Что должно получиться

2	3	4	5
5	9	12	9
23	40	44	33
…	…	…	…

Теперь можно и третью строку B заполнить

L

2	3	4	5
5	11	15	13

R

2	3	4	5
8	13	17	9

$B[i, j] = 2*B[i-1,j] + L[i-1,j-1] + R[i+1,j+1]$
$L[i, j] = L[i-1,j-1] + B[i,j]$
$R[i, j] = R[i-1,j+1] + B[i,j]$

B

2	3	4	5
5	9	12	9
23	40	44	2*9 + 15 + 0

Что должно получиться

2	3	4	5
5	9	12	9
23	40	44	33
…	…	…	…

Теперь можно и третью строку B заполнить

L

2	3	4	5
5	11	15	13
23+0			

R

2	3	4	5
8	13	17	9

B[i, j] = 2*B[i-1,j] + L[i-1,j-1] + R[i+1,j+1]
L[i, j] = L[i-1,j-1] + B[i,j]
R[i, j] = R[i-1,j+1] + B[i,j]

B

2	3	4	5
5	9	12	9
23	40	44	33

Что должно получиться

2	3	4	5
5	9	12	9
23	40	44	33
…	…	…	…

Далее заполняем по формулам третьи строки L и R

L

2	3	4	5
5	11	15	13
23	40 + 5		

R

2	3	4	5
8	13	17	9

B[i, j] = 2*B[i-1,j] + L[i-1,j-1] + R[i+1,j+1]
L[i, j] = L[i-1,j-1] + B[i,j]
R[i, j] = R[i-1,j+1] + B[i,j]

B

2	3	4	5
5	9	12	9
23	40	44	33

Что должно получиться

2	3	4	5
5	9	12	9
23	40	44	33
...	...	...	...

Далее заполняем по формулам третьи строки L и R и т.д.

L

2	3	4	5
5	11	15	13
23	45	11+44	

R

2	3	4	5
8	13	17	9

B[i, j] = 2*B[i-1,j] + L[i-1,j-1] + R[i+1,j+1]
L[i, j] = L[i-1,j-1] + B[i,j]
R[i, j] = R[i-1,j+1] + B[i,j]

B

2	3	4	5
5	9	12	9
23	40	44	33

Что должно получиться

2	3	4	5
5	9	12	9
23	40	44	33
...	...	...	...

Далее заполняем по формулам третьи строки L и R и т.д.

Путешествия развивают ум, если, конечно, он у вас есть.

Гилберт Честертон

Задача «Черепашка»

9	8	6	2
10	11	13	11
3	7	12	8
5	9	13	9

Дана прямоугольная таблица (*n* строк, *m* столбцов), в клетках которой записаны целые числа Черепашка находится в левой нижней клетке, и ей необходимо попасть в правую верхнюю клетку. За один ход Черепашка может переместиться в соседнюю верхнюю или правую клетку. Требуется найти путь Черепашки с максимальной суммой элементов.

Решение задачи «Черепашка». П.П.

- Полный перебор вариантов – универсальный способ решения. Но рассмотрим его потенциальные возможности

- Пусть дана таблица 4x4. Любой путь состоит из трёх перемещений вверх и трех перемещений вправо, т.е. длина пути равна шести. Другими словами, дано 6 шагов, из них 3 выбираются для перемещений вверх, оставшиеся 3 – для перемещений вправо определяются однозначно. Т.о. количество способов выбора трех перемещений из шести

$$C_6^3 = \frac{6!}{3!\cdot 3!} = 20$$

В общем случае C_{n+m-2}^{n-1}

- При нахождении суммы (стоимости) пути потребуется 5 операци сложения, всего 100 операций. Оценим время решения задачи для компьютера с миллионным быстродействием (см. презентация предыдущих занятий о сложности алгоритмов и быстродействии на примере задачи о тупоугольном треугольнике)

Длительность вычислений

Размер таблицы	Длина пути	Количество путей	Количество операций сложения	Приблизительное время решения задачи
4 × 4	6	20	100	0,0001 с
8 × 8	14	3432	44616	0,045 с
31 × 31	60	~10^{17}	~59 × 10^{17}	~200 000 лет

Итак, возможности полного перебора вариантов ограничены.

Решение задачи «Черепашка». Д.П.

Рассмотрим другой способ решения задачи. Определим подзадачу как ту же самую задачу, но для таблицы меньшего размера, и «свяжем» решения подзадач с решением исходной задачи. Для таблицы размера 4 × 4 подзадача — это решение для таблиц с размерами 1 × 2, 2 × 1, 2 × 2, 1 × 3, 2 × 3, 3 × 2 и т. д. Для таблиц размеров 1 × 2, 1 × 3, 1 × 4 движение Черепашки происходит только вправо, Черепашка может попасть единственным образом в последние клетки таблиц, поэтому стоимость считается однозначно (сумма стоимости клеток). Результаты решений запоминаем в массиве *B* (рис. 1.5, *б*). Массив *B* формируется, начиная с нижней левой клетки — *B*(1,1). Аналогично для таблиц размеров 2 × 1, 3 × 1, 4 × 1 движение Черепашки — только вверх. Рассмотрим таблицу размера 2 × 2 (рис. 1.6, *а*). У Черепашки два способа попадания в правую верхнюю клетку такой таблицы — или справа, или снизу. Выбираем тот, который дает максимальную сумму, и фиксируем результат. Аналогично и для таблицы размера 2 × 3 (рис. 1.6, *б*).

9	8	6	2
10	11	13	11
3	7	12	8
5	9	13	9

а

27	40	58	**65**
18	32	**52**	**63**
8	21	**39**	47
5	**14**	**27**	36

б

max (*B*[1,2], *B*[2,1]) + *A*[2,2]

8	21
5	14

а

max (*B*[1,3], *B*[2,2]) + *A*[2,3]

8	21	39
5	14	27

б

Рис. 1.6. Принцип решения подзадач

Код (на паскале)

Формализованная запись логики иллюстрирует простоту решения:

```
Procedure Solve;
  Var i, j: LongInt;
  Begin
    B[1,1]:=A[1,1];
    For i:=2 To n Do B[i,1]:=B[i-1,1]+A[i,1];
    For j:=2 To m Do B[1,j]:=B[1,j-1]+A[1,j];
    For i:=2 To n Do
      For j:=2 To m Do B[i,j]:=Max(B[i-1,j],
                               B[i,j-1])+A[i,j];
      {Max - функция нахождения максимального
       из двух чисел.}
  End;
```

После полного вычисления *B* мы находим стоимость пути Черепашки (для рассматриваемого примера она равна 65)

Вычисление пути

После полного вычисления B мы находим стоимость пути Черепашки (для рассматриваемого примера она равна 65), но не сам путь (он выделен на рис. жирным шрифтом). Для нахождения пути Черепашки следует выполнить «обратный просмотр» массива B. Его суть: из значения $B[i,j]$ вычитаем $A[i,j]$ и смотрим, которое из двух чисел — $B[i-1,j]$ или $B[i,j-1]$ — равно полученному числу. Осуществляем переход по равенству и продолжаем до тех пор, пока не будет достигнут элемент $B[1,1]$.

27	40	58	**65**
18	32	**52**	**63**
8	21	**39**	47
5	**14**	**27**	36

Вычисление пути

Естественно, что следует предусмотреть ситуации наличия одной соседней клетки. Рекурсивный вариант реализации этой логики имеет следующий вид:

```
Procedure Way(i, j: LongInt);
  Begin
    If (i=1) And (j=1) Then Exit;
    If (i=1) And (j>1) Then Way(i,j-1)
      Else If (i>1) And (j=1) Then Way(i-1,j)
        Else
          If B[i,j]-A[i,j]=B[i-1,j]
            Then Way(i-1,j)
            Else Way(i,j-1);
    Write(i, ' ', j, '; ');
  End;
```

В рассмотренном варианте массив B формировался, начиная с элемента $B[1,1]$.

Временна́я сложность решения — $O(n \cdot m)$. Для вычисления каждого значения B требуется максимум две операции — сравнение и сложение. Для таблицы размером $n = 300$, $m = 300$ общее количество операций меньше 1 000 000, т. е. компьютер с миллионным быстродействием выполнит задачу менее чем за одну секунду.

Сдать можно как задачу №2965

- Там даже не требуется вывести путь
- И идет черепашка в другом направлении
- http://informatics.mccme.ru/mod/statements/view3.php?id=656&chapterid=2965#1