1.3. Наибольший квадрат

> Рассказывают, что Резерфорд, принимая в свою лабораторию нового сотрудника, давал ему задание. Если по выполнении сотрудник приходил к Резерфорду и просил второе задание, тот его увольнял. Великий исследователь понимал: второе задание должно проистекать из первого, хорошее исследование должно само рождать новые вопросы.

Дана матрица A из $0 < n \le 1000$ строк и $0 < m \le 1000$ столбцов. Элементы матрицы A равны 0 или 1. Найти «наибольший квадрат» — квадратную подматрицу максимального размера, состоящую из одних единиц, которая есть в A.

Пример:

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Наибольший квадрат — это подматрица размером 3×3, левый верхний угол которой находится в позиции $A[3,3]$, $A[3,4]$ или $A[3,5]$.

Полный перебор — это решение со сложностью $O(n \cdot m^4)$ (если $n \ge m$), заключающееся в том, что, начиная с каждой позиции $A[i,j]$, с помощью конструкции из двух вложенных циклов проверяются всевозможные квадраты и проверяется, состоит квадрат или нет из единиц.

Понятие подзадачи в данном случае сводится к определению наибольшего квадрата в подматрице $A[1..i,1..j]$, заканчивающегося в позиции (i,j). Другими словами, наибольшего квадрата, правый нижний угол которого находится в позиции (i,j). Для хранения результатов решения подзадач требуется двумерный массив B той же размерности. Как связаны подзадачи? Другими словами, как осуществляется переход от задач меньшей размерности к задачам большей размерности? Предположим, что наибольшие квадраты для подматриц $A[1..i-1,1..j]$, $A[1..i,1..j-1]$ и $A[1..i-1,1..j-1]$ уже вычислены и их размеры являются значениями переменных $B[i-1,j]$, $B[i,j-1]$, $B[i-1,j-1]$. Значение $A[i,j]$ равно единице. Очевидно, что в позицию (i,j) может быть «продолжен» только минимальный из трех построенных квадратов, и его размер (длина стороны) будет на единицу больше минимального (рис. 1.3).

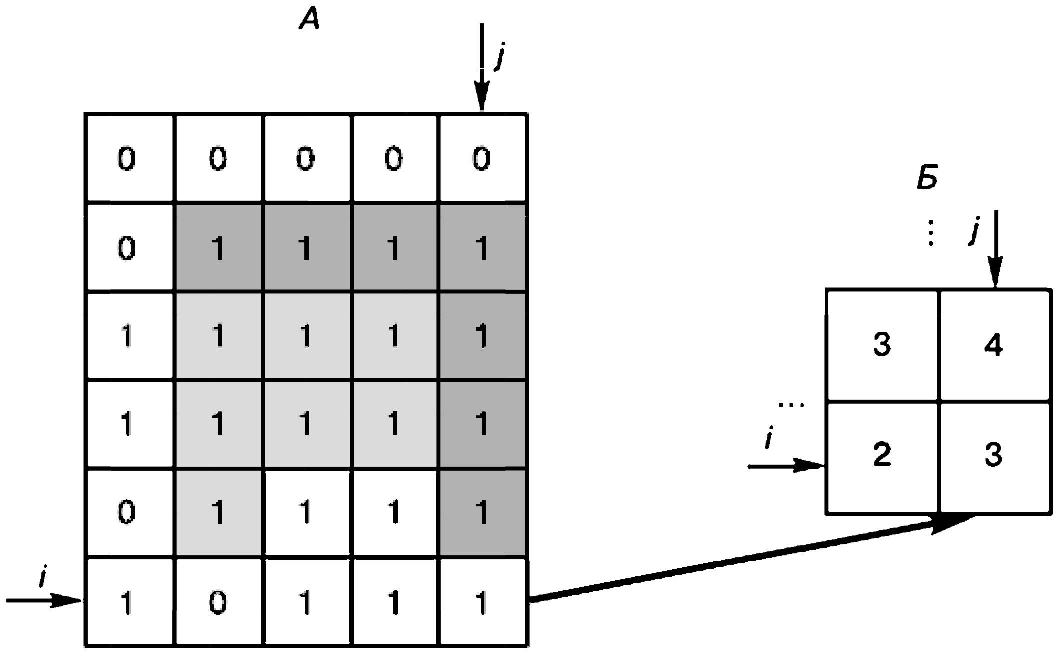

Рис. 1.3. Пример вычисления $B[i,j]$

Таким образом, связь подзадач описывается следующим рекуррентным соотношением:

$$B[i,j]=\begin{cases} A[i,j], & \text{при } i=1 \text{ или } j=1; \\ 0, & \text{при } A[i,j]=0; \\ \min(B[i-1,j],B[i,j-1],B[i-1,j-1])+1, & \text{при } A[i,j]=1. \end{cases}$$

Формализованная запись логики выглядит так:

```
Procedure Solve;
  Var i, j, l, k, max: LongInt;
  Begin
    For i:=1 To n Do B[i,1]:=A[i,1];
    For j:=1 To m Do B[1,j]:=A[1,j];
    For i:=2 To n Do
      For j:=2 To m Do
        If A[i,j]=0 Then B[i,j]:=A[i,j]
          Else B[i,j]:=Min(B[i-1,j],B[i,j-1],
                 B[i-1,j-1])+1;
    {Функция нахождения минимального из трех
     чисел}
```

```
max:=0;
For i:=1 To n Do
   For j:=1 To m Do
      If B[i,j]>max Then Begin
         max:=B[i,j]; l:=i; k:=j;
      End;
WriteLn('левые верхние координаты
      максимального квадрата', l-max+1,
       ' ', k-max+1);
End;
```

Временна́я сложность решения — $O(n^2)$.

Итак, опять вернемся к понятию динамического программирования, не формулируя полностью, что это такое. Есть задача, решение которой требуется найти. Определили понятие подзадачи и поняли, как решать подзадачи. Показали, что решение исходной задачи «складывается» из решения подзадач, и установили, как происходит процесс «сложения». То, о чем мы говорим, можно определить как некое свойство, некую характеристику исходной задачи, а именно её аддитивность[1].

А если требуется найти не квадрат, а наибольший прямоугольник, состоящий из одних единиц? Для простоты предположим, что находится его «площадь». В этом случае рассмотренное понятие подзадачи «не работает». Нельзя найти прямоугольник с наибольшей «площадью» по трем соседним максимальным прямоугольникам. Причина — нет прямой зависимости между «площадью» и длиной стороны, как в случае квадрата. «Площадь» прямоугольника определяют две стороны, и поэтому «пространство» возможных значений, кандидатов на решение, значительно больше.

Задание

Покажите на примере неработоспособность введенной подзадачи при поиске прямоугольника с наибольшей площадью.

Преобразуем матрицу A. В каждой строке, для каждой позиции, подсчитаем длину «полоски» (одномерная матрица — строка) из единиц, заканчивающейся в данной позиции. Результаты запомним в матрице B. Это нетрудно сделать с помощью следую-

[1] Аддитивность — понятие латинского происхождения, смысл которого означает некое свойство, имеющее отношение к операции сложения.