Дистанционная подготовка к Всероссийской олимпиаде по информатике

Преподаватель:

к.ф.-м.н., заведующий кафедрой ВТиКГ ДВГУПС,

преподаватель программы IT-школа Samsung,

Пономарчук Юлия Викторовна

E-mail: yulia.ponomarchuk@gmail.com

Динамическое программирование

Идея метода

Метод динамического программирования используется для задач, обладающих следующим свойством:

имея решения некоторых подзадач (для меньшего числа *N*), можно найти решение исходной задачи, т.е. **оптимальное решение подзадачи большего размера можно построить из оптимальных решений подзадач**.

Одномерное ДП

Последовательность Фибоначчи задается формулами:
$F_1 = 1$, $F_2 = 1$, $F_n = F_{n-1}+F_{n-2}$ при $n>1$.
Необходимо найти F_n по номеру n.

Неэффективное рекурсивное решение:

```
function F(n: integer): integer;
begin
   if n<2 then F := 1
   else F := F(n-1) + F(n-2);
end;
```

Эффективное решение по методу ДП:

```
F[1] = 1;
F[2] = 1;
for i:=3 to n do
   F[i] := F[i-1] + F[i-2];
```

Одномерное ДП

Посчитать число последовательностей нулей и единиц длины n, в которых не встречаются две идущие подряд единицы.

- Обозначим K[i] – число таких последовательностей длины i
- Тривиальные случаи: K[1] = 2, K[2] = 3;
- Ответом является значение K[n]

Одномерное ДП

Проанализируем последовательность длины i.

- Если последний ее символ равен 0, то первые i-1 – любая правильная последовательность длины i-1 (не важно, заканчивается она нулем или единицей). Таких последовательностей K[i-1].
- Если последний ее символ равен 1, то предпоследний символ обязательно должен быть равен 0 (иначе будет две единицы подряд), а первые i-2 символа – любая правильная последовательность длины i-2. Таких последовательностей K[i-2].
- Таким образом, K[i] = K[i-1] + K[i-2], т.е. данная задача сводится к нахождению чисел Фибоначчи

Двумерное ДП

Дано прямоугольное поле размером n на m клеток. Можно совершать шаги длиной в одну клетку вправо и вниз. Посчитать, сколькими способами можно попасть из левой верхней клетки (с координатами (1,1)) в правую нижнюю (с координатами (n, m)).

Двумерное ДП

- В некоторую клетку с координатами (i,j) можно прийти только сверху или слева, т.е. из клеток с координатами (i-1, j) и (i, j-1).

- Таким образом, для клетки (i, j) число маршрутов

A[i,j] = A[i-1,j] + A[i,j-1], т.е. задача сводится к двум подзадачам.

- Необходимо последовательно пройти по строкам (или столбцам), находя число маршрутов для текущей клетки по формуле.
- Тривиальный случай: A[1,1] = 1
- Ответ находится в элементе A[n,m]

Двумерное ДП

(*Задача о черепашке*). Дано прямоугольное поле размером n на m клеток. Можно совершать шаги длиной в одну клетку вправо, вниз или по диагонали вправо-вниз.

В каждой клетке записано некоторое натуральное число.

Необходимо попасть из левой верхней клетки (с координатами (1,1)) в правую нижнюю (с координатами (n,m)).

Вес маршрута вычисляется как сумма чисел со всех посещенных клеток.

Необходимо найти маршрут с минимальным весом.

Двумерное ДП

- В некоторую клетку с координатами (i,j) можно прийти из клеток с координатами $(i-1, j)$, $(i, j-1)$ и $(i-1, j-1)$.
- Допустим, что для каждой из этих трех клеток уже найден маршрут минимального веса, а сами эти веса находятся в

$$W[i-1,j],\ W[i,j-1] \text{ и } W[i-1,j-1].$$

- Чтобы найти минимальный вес для (i,j), необходимо выбрать минимальный из весов $W[i-1,j]$, $W[i,j-1]$, $W[i-1,j-1]$ и прибавить к нему число, записанное в текущей клетке:

$$W[i,j] = W[i-1,j] + W[i,j-1] + W[i-1,j-1].$$

Задача о рюкзаке

Имеется судно грузоподъемностью *W* и *N* предметов. Известно, что *i*-й предмет имеет вес w_i и ценность c_i. Необходимо загрузить судно предметами так, чтобы получить максимальную прибыль.

Задача о рюкзаке

Обозначим через $f(i,w)$ максимальную стоимость, которую можно получить имея лишь первые i грузов и грузоподъемность w.

Если i-й груз не использовался при подсчете $f(i,w)$, то

$$f(i,w) = f(i-1,w).$$

Иначе $f(i,w) = f(i-1,w-w_i) + c_i$

Из двух вариантов выбираем максимальный:

$f(i,w) = \max(f(i-1,w), f(i-1,w-w_i) + c_i), i>1$

Начальные условия: $f(1,w_i) = c_i$

возрастающей подпоследовательности

Дана последовательность целых чисел. Необходимо найти длину ее самой длинной возрастающей подпоследовательности.

Пример

Последовательность 4, 1, 7, 5, 2, 5, 8, 3

Ответ: длина 4 (1, 2, 5, 8)

возрастающей подпоследовательности

Пусть $f[i]$ – длина наибольшей возрастающей подпоследовательности среди элементов $a[1], a[2], \ldots, a[i]$, где $a[i]$ – последний элемент возрастающей подпоследовательности.

Определим возрастающие подпоследовательности, к которым допустимо прибавление $a[i]$. Они обязаны заканчиваться на элемент $a[j]$ ($j<i$), меньший чем $a[i]$.

$f[i] = 1 + \max(f[j])$ для всех j, таких что $j=1\ldots i$ и $a[j]<a[i]$

Задача о палиндроме

Палиндром – это симметричная строка, т.е. строка, которая одинаково читается как слева направо, так и справа налево. Требуется по заданной строке определить минимальное количество символов, которые необходимо вставить в строку для преобразования ее в палиндром. Прописные и строчные буквы считаются различными.

Пример. После вставки двух символов строка **Ab3bd** может быть преобразована в палиндром **dAb3bAd** или **Adb3bdA**. Вставкой менее двух символов палиндром получить нельзя.

Задача о палиндроме

- $f[i,j]$ – минимальное количество символов, которые необходимо вставить в подстроку $S[i..j]$ (где i – номер крайнего левого символа исходной строки, j – крайнего правого) для того, чтобы получить искомый палиндром.

- Искомый результат – $f[1,n]$

Задача о палиндроме

- Рассмотрим строку S[i..j]
- Если символы S[i] и S[j] совпадают, то для преобразования строки в палиндром требуется столько же символов, сколько для преобразования строки S[i+1..j-1].
- При несовпадении символов требуется добавить один символ или к подстроке S[i+1..j] или к подстроке S[i+1..j] – к той из них, для которой преобразование в палиндром осуществляется за минимальное количество символов.

$$f[i,j]=\begin{cases} 0, & \text{если } i \ge j \\ f[i+1, j-1], & \text{если } S[i]=S[j],\ i<j \\ \min\left(f[i+1, j], f[i, j-1]\right)+1, & \text{если } S[i]\ne S[j],\ i<j \end{cases}$$