НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Динамическое программирование

Беркунский Е.Ю., кафедра ИУСТ, НУК
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua

Немного теории

Динамическое программирование — метод решения сложных задач путём разбиения их на более простые подзадачи.

- Применяется к задачам, выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной.
- В этом случае время вычислений, по сравнению с «наивными» методами, можно значительно сократить.

Это уже известно…

Последовательность Фибоначчи: $F_n = F_{n-1} + F_{n-2}$

- Каждое число в последовательности вычисляется как сумма двух предыдущих
- «Наивным» будет решение «в лоб», через рекурсию…

Числа Фибоначчи

```
int fib(int n) {
    if (n<2) return 1;
    return fib(n-1) + fib(n-2);
}
```

Числа Фибоначчи

```
int fib(int n) {
    if (n<2) return 1;
    return fib(n-1) + fib(n-2);
}
```

fib(5)=fib(4)+fib(3)

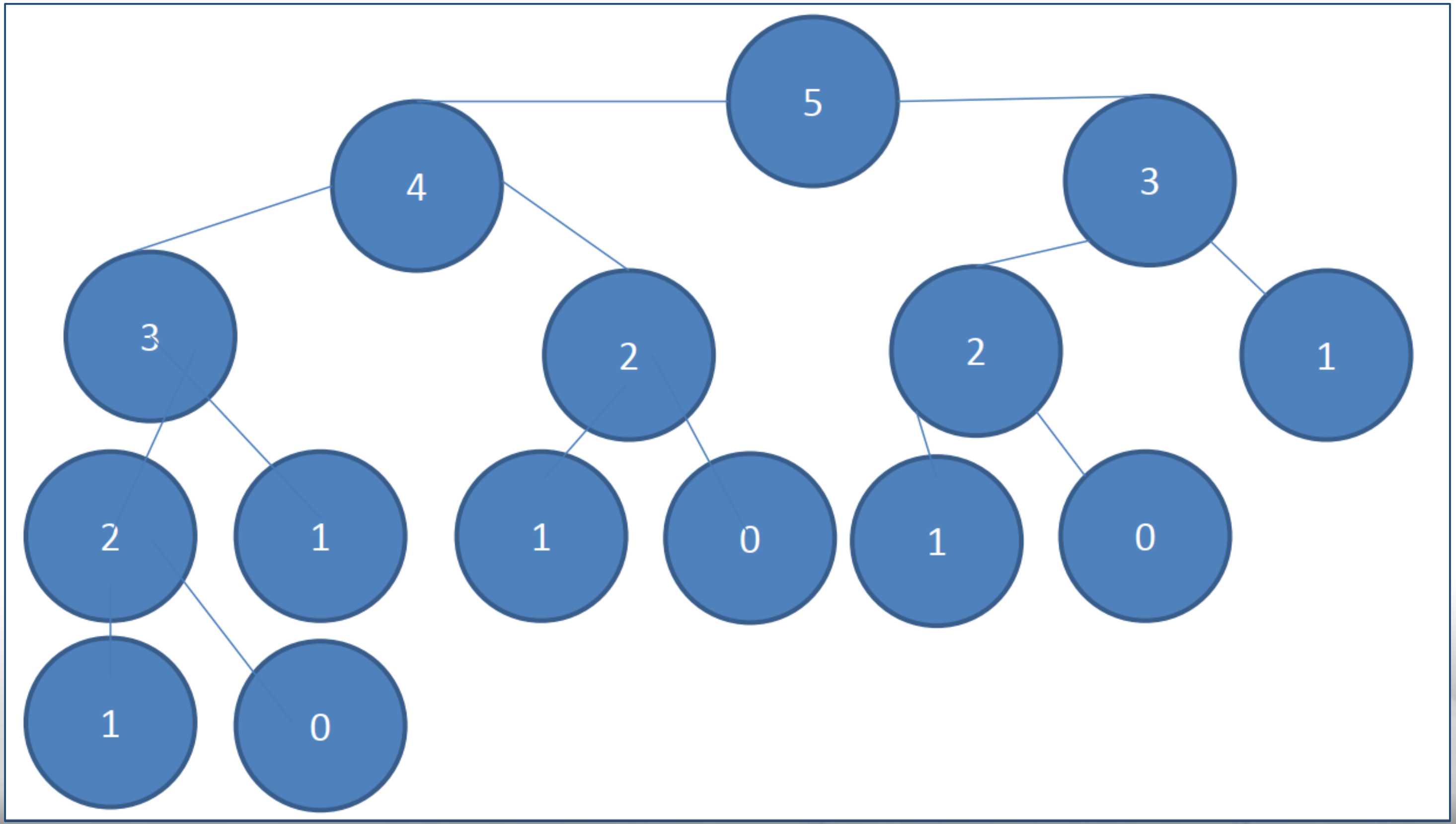
fib(4)=fib(3)+fib(2)

fib(3)=fib(2)+fib(1)

fib(2)=fib(1)+fib(0)

Числа Фибоначчи

```
int fib(int n) {
    if (n<2) return 1;
    return fib(n-1) + fib(n-2);
}
```



Минимальный путь в таблице

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.

Минимальный путь в таблице

1	1	1	1
5	2	2	100
9	4	2	1

Минимальный путь в таблице

1	1	1	1
5	2	2	100
9	4	2	1

Для такой таблицы сумма будет равна 8.

Дополнительная задача:
Сколько всего различных путей существует?

Подпоследовательность

Дана последовательность целых чисел.

Требуется найти длину наибольшей возрастающей подпоследовательности.

Пример:

3 29 5 5 28 6

Подпоследовательность

Дана последовательность целых чисел.

Требуется найти длину наибольшей возрастающей подпоследовательности.

Пример:
3 29 5 5 28 6

Ответ: 3

Дополнительно: найти саму возрастающую подпоследовательность

Подпоследовательность

Пример:

3 29 5 5 28 6

1 шаг: 3

2 шаг: 3 | 3 29

3 шаг: 3 | 3 5

4 шаг: 3 | 3 5

5 шаг: 3 | 3 5 | 3 5 28

6 шаг: 3 | 3 5 | 3 5 6

Подпоследовательность

Наибольшая возрастающая подпоследовательность имеет применения в физике, математике, теории представления групп, теории случайных матриц.

В общем случае известно решение этой задачи за время ***n log n*** в худшем случае.

Наибольшая общая подпоследовательность (НОПП)

Задача нахождения НОПП — задача поиска последовательности, которая является подпоследовательностью нескольких последовательностей (обычно двух).

- Часто задача определяется как поиск *всех* наибольших подпоследовательностей.
- Это классическая задача информатики, которая имеет приложения, в частности, в задаче сравнения текстовых файлов (утилита diff), а также в биоинформатике.

НОПП решение

«Наивное» решение – полный перебор.
Экспоненциальная сложность ☹

«Умное решение» – ДП:
Вначале найдём **длину** наибольшей подпоследовательности.
Допустим, мы ищем решение для случая (n1, n2),
где n1, n2 — длины первой и второй строк.

НОПП решение

Вначале найдём **длину** наибольшей подпоследовательности.
Допустим, мы ищем решение для случая (n_1, n_2),
где n_1, n_2 — длины первой и второй строк.
Пусть уже существуют решения для всех подзадач (m_1, m_2), меньших заданной.
Тогда задача (n_1, n_2) сводится к меньшим подзадачам следующим образом:

НОПП решение

задача (n_1, n_2) сводится к меньшим подзадачам следующим образом:

$$f(n_1, n_2) = \begin{cases} 0, & n_1 = 0 \,||\, n_2 = 0 \\ f(n_1 - 1, n_2 - 1) + 1, & s_1[n_1] = s_2[n_2] \\ \max\big(f(n_1 - 1, n_2), f(n_1, n_2 - 1)\big), & s_1[n_1] \neq s_2[n_2] \end{cases}$$

НОПП решение

- Для построения самой последовательности в существующий алгоритм добавим запоминание для каждой задачи той подзадачи, через которую она решается
- Следующим действием, начиная с последнего элемента, поднимаемся к началу по направлениям, заданным первым алгоритмом, и записываем символы в каждой позиции.

Задача об умножении матриц

- Как известно, матрицы умножаются по правилу «строка на столбец».
- При этом, количество столбцов в первой матрице должно равняться количеству строк во второй.
- В результате умножения получается матрица, у которой количество строк такое же, как в первой, а столбцов – как во второй
-

Задача об умножении матриц

A[m][k] * B[k][n] = C[m][n]

Операция умножения матриц некоммутативна,

т.е. $A*B \neq B*A$

Операция умножения матриц ассоциативна,

т.е. $A*(B*C) = (A*B)*C$

А теперь, собственно, задача…

Задача об умножении матриц

- Дана последовательность матриц $A_1, A_2, \ldots, A_n$

- Требуется минимизировать количество скалярных операций для вычисления их произведения.
- Когда матрицы велики по одному измерению и малы по другому, количество скалярных операций может серьёзно зависеть от порядка перемножений матриц

Задача об умножении матриц

Допустим, нам даны 3 матрицы A_1,A_2,A_3 размерами соответственно 10×100, 100×5 и 5×50.

Существует 2 способа их перемножения (расстановки скобок):
$((A_1A_2)A_3)$ и $(A_1(A_2A_3))$

При способе $(A_1A_2)A_3$ потребуется
10·100·5 + 10·5·50 = **7500** скалярных умножений
При способе $A_1(A_2A_3)$ потребуется
100·5·50 + 10·100·50 = **75000** скалярных умножений

Задача об умножении матриц

Обозначим через m[i, j] минимальное количество скалярных умножений для вычисления матрицы *Ai..j*.

Получаем следующее рекуррентное соотношение:

$$m[i,j] = \begin{cases} 0, & i = j \\ min(m[i,k] + m[k+1,j] + p_{i-1}p_kp_j \mid i <= k < j) & i < j \end{cases}$$

Задача об умножении матриц

Будем запоминать в двумерном массиве m результаты вычислений для подзадач, чтобы избежать пересчета для уже вычислявшихся подзадач.
После вычислений ответ будет в m[1,n]
(Сколько перемножений требуется для последовательности матриц от 1 до n — то есть ответ на поставленную задачу)

Задачи

Соревнование по ссылке

- https://www.e-olymp.com/ru/contests/12112

- 115,7447,1521,595,1228,4054,15,5062,1285,1740