

академия
больших
данных

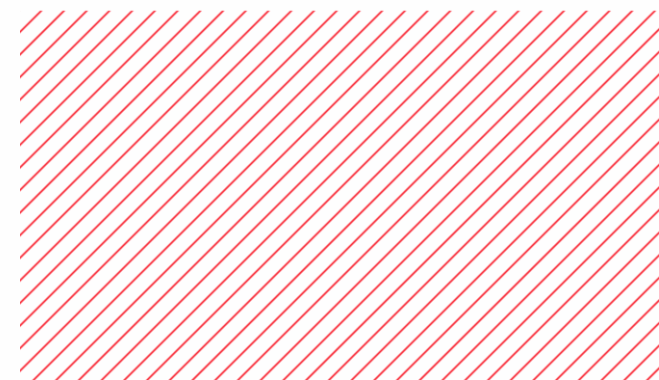
mail.ru
group

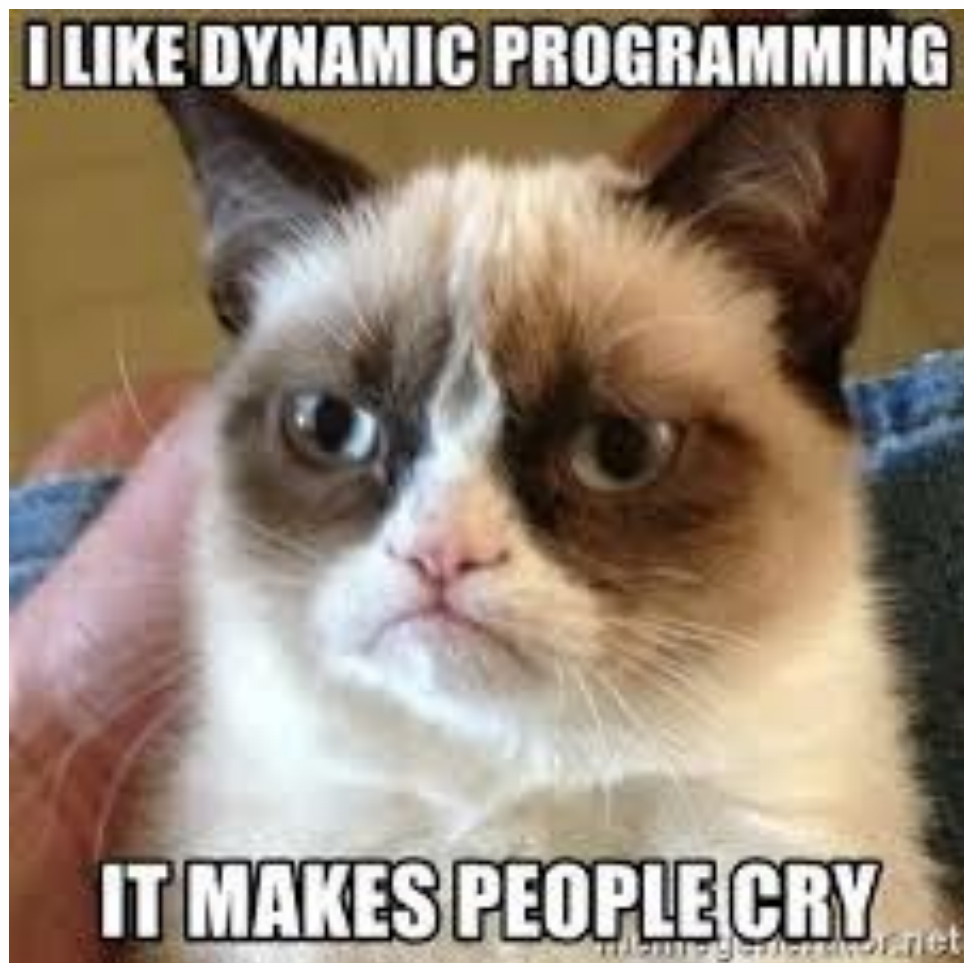


Базовое динамическое программирование

Шовкопляс Григорий

Введение в алгоритмы и структуры данных





Что такое
динамическое
программирование



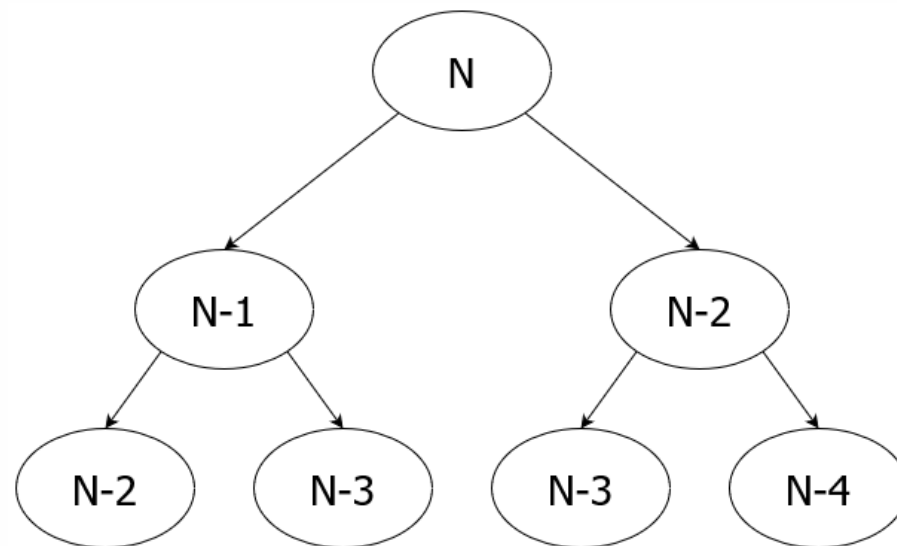
Введение

Динамическое программирование — это когда у нас есть одна большая задача, которую непонятно как решать, и мы разбиваем ее на меньшие задачи, которые тоже непонятно как решать.

А.С.Кумок

Числа Фибоначчи

$$F_n = F_{n-1} + F_{n-2}$$





Как решать задачи ДП

- Что храним в ДП?
- База
- Переход
- Порядок обхода
- Где ответ?



Поиск n -го числа Фибоначчи

- Что храним в ДП?
- База
- Переход
- Порядок обхода
- Где ответ?
- $F[i]$ — i -ое число Фибоначчи
- $F[0] = 0; F[1] = 1$
- $F[i] = F[i-1] + F[i-2]$
- По возрастанию i
- $F[n]$



Различные подходы ДП

- ДП назад
 - $F[i] = F[i-1] + F[i-2]$
- ДП вперед
 - $F[i+1] += F[i]$
 - $F[i+2] += F[i]$

Ленивое ДП

Значение ДП вычисляется
только тогда, когда к нему
обращаются

```
Fib(n)
    if n = 0
        return 0
    if n = 1
        return 1
    return Fib(n - 1) + Fib(n - 2)
```


Ленивое ДП

Значение ДП вычисляется
только тогда, когда к нему
обращаются + мемоизация

```
Fib(n)
```

```
    if n = 0
```

```
        return 0
```

```
    if n = 1
```

```
        return 1
```

```
    if F[n] != -1
```

```
        F[n] = Fib(n - 1) + Fib(n - 2)
```

```
    return F[n]
```

Базовые задачи ДП



Кузнечик

- На прямой есть n кочек
- Кузнечик прыгает на следующую кочку либо через одну
- Сколько способов у кузнечика есть добраться от 1 кочки до кочки n ?



Кузнечик

- Что храним в ДП?
- База
- Переход
- Порядок обхода
- Где ответ?
- $dp[i]$ – число способов добраться до кочки i
- $dp[1] = 1$
- $dp[i] = dp[i-1] + dp[i-2]$
- По возрастанию i
- $dp[n]$



Кузнечик

- На прямой есть n кочек
- Кузнечик прыгает на 1, 2, 3,... к кочек вправо
- Сколько способов у кузнечика есть добраться от 1 кочки до кочки n ?



Кузнечик

- Что храним в ДП?
 - База
 - Переход
 - Порядок обхода
 - Где ответ?
- $dp[i]$ – число способов добраться до кочки i
 - $dp[1] = 1$
 - $dp[i] = \sum_{j=1}^k dp[i-j]$
 - По возрастанию i
 - $dp[n]$



Черепашка

- Есть поле $N \times M$
- Черепашка ходит на одну клетку вверх или вправо
- В некоторых клетках есть яблочки
- Какое наибольшее число яблочек черепашка сможет собрать по пути из левой нижней клетки в правую верхнюю



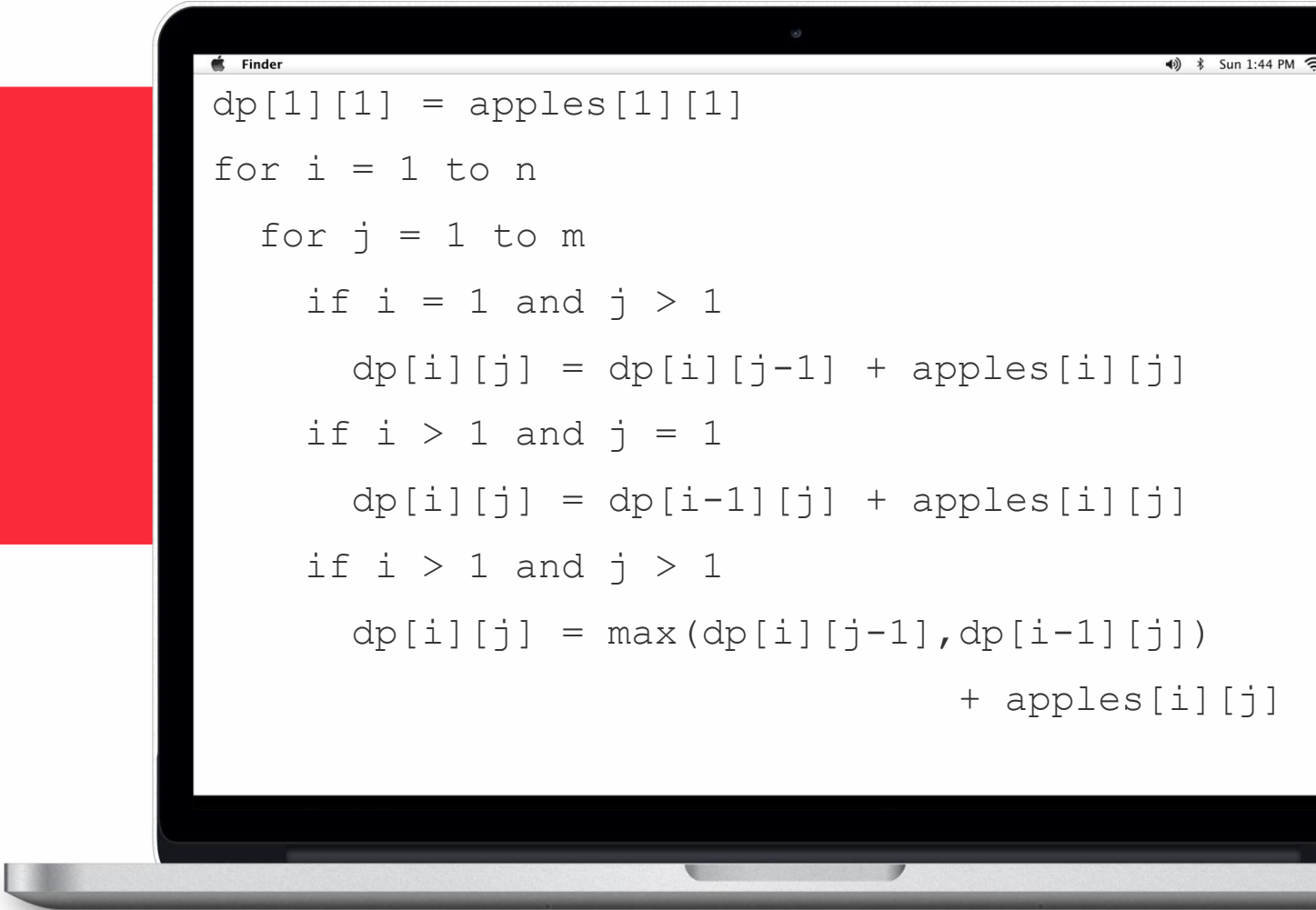
Черепашка

- Что храним в ДП?
 - База
 - Переход
 - Порядок обхода
 - Где ответ?
- $dp[i][j]$ — максимальное число яблок на пути в клетку (i, j)
 - $dp[1][1] = apples[1][1]$
 - $dp[i][j] = \max(dp[i-1][j], dp[i][j-1]) + apples[i][j]$
 - $dp[i][j] = dp[i-1][j] + apples[i][j]$,
если $j = 1$
 - $dp[i][j] = dp[i][j-1] + apples[i][j]$,
если $i = 1$
 - По возрастанию i , По возрастанию j
 - $dp[n][m]$



Черепашка

Псевдокод



```
dp[1][1] = apples[1][1]
for i = 1 to n
  for j = 1 to m
    if i = 1 and j > 1
      dp[i][j] = dp[i][j-1] + apples[i][j]
    if i > 1 and j = 1
      dp[i][j] = dp[i-1][j] + apples[i][j]
    if i > 1 and j > 1
      dp[i][j] = max(dp[i][j-1], dp[i-1][j])
                      + apples[i][j]
```

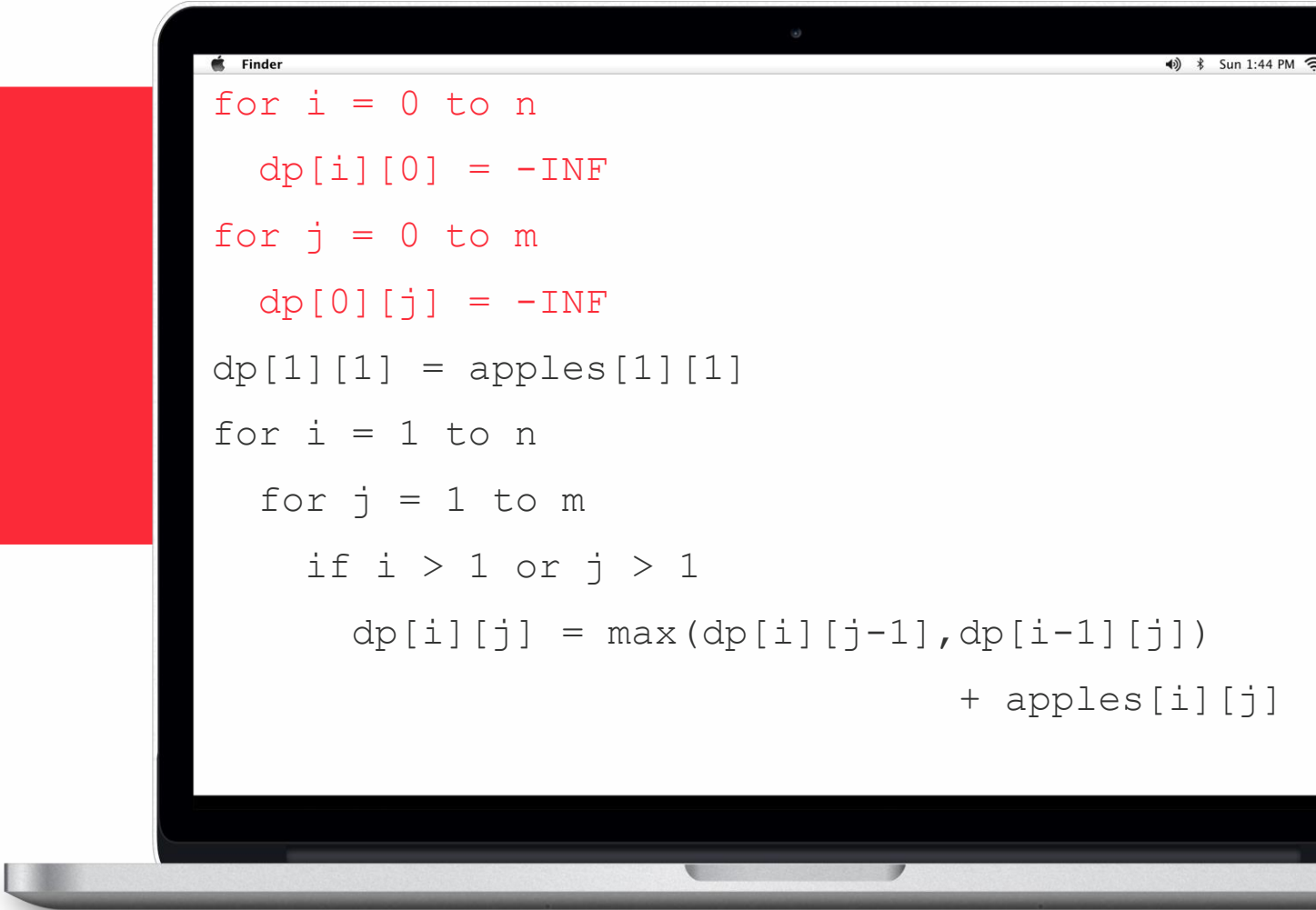
Черепашка + «Каемочка»

- Что храним в ДП?
 - База
 - Переход
 - Порядок обхода
 - Где ответ?
- $dp[i][j]$ — максимальное число яблок на пути в клетку (i, j)
 - $dp[i][0] = -INF, dp[0][j] = -INF, dp[1][1] = apples[1][1]$
 - $dp[i][j] = \max(dp[i-1][j], dp[i][j-1]) + apples[i][j]$
 - По возрастанию i , По возрастанию j
 - $dp[n][m]$



Черепашка

Псевдокод + «Каемочка»



```
Finder
for i = 0 to n
    dp[i][0] = -INF
for j = 0 to m
    dp[0][j] = -INF
dp[1][1] = apples[1][1]
for i = 1 to n
    for j = 1 to m
        if i > 1 or j > 1
            dp[i][j] = max(dp[i][j-1], dp[i-1][j])
                                + apples[i][j]
```



Черепашка. Восстановление ответа

- Есть поле $N \times M$
- Черепашка ходит на одну клетку вверх или вправо
- В некоторых клетках есть яблочки
- Найти такой путь из левой нижней клетки в правую верхнюю, что черепашка соберет наибольшее возможное число яблочек

Черепашка. Восстановление ответа

Восстановление ответа
просмотром предыдущих
состояний

...

```
if i > 1 and j > 1
```

```
    dp[i][j] = max(dp[i][j-1], dp[i-1][j])  
                  + apples[i][j]
```

```
i = n, j = m, ans = []
```

```
while i != 1 or j != 1
```

```
    ans.add((i, j))
```

```
    if dp[i][j-1] > dp[i-1][j]
```

```
        j = j - 1
```

```
    else
```

```
        i = i - 1
```

```
reverse(ans)
```

Черепашка. Восстановление ответа

Восстановление ответа через
массив предков p

...

```
for i = 1 to n
```

```
  for j = 1 to m
```

```
    if i > 1 or j > 1
```

```
      if dp[i-1][j] > dp[i][j-1]
```

```
        dp[i][j] = dp[i-1][j] + apples[i][j]
```

```
        p[i][j] = (i-1, j)
```

```
    else
```

```
      dp[i][j] = dp[i][j-1] + apples[i][j]
```

```
      p[i][j] = (i, j-1)
```

Черепашка. Восстановление ответа

Восстановление ответа через
массив предков p

...

```
i = n, j = m, ans = []  
while i > 0 and j > 0  
    ans.add((i, j))  
    i = p[i][j].first  
    j = p[i][j].second  
    ni = p[i][j].first  
    j = p[i][j].second  
    i = ni  
reverse(ans)
```



Наибольшая
возрастающая
подпоследовательность



Наибольшая возрастающая подпоследовательность

- Есть последовательность чисел $a_1, a_2, a_3 \dots a_n$
- Найти ее подпоследовательность наибольшей длины, такую что:
 - $\forall i < j: a_{k_i} < a_{k_j}$
- 2, 1, 3, 5, 7, 4, 3, 5, 1, 8
 - 1, 3, 4, 5, 8
 - 2, 3, 5, 7, 8



Наибольшая возрастающая подпоследовательность

- Что храним в ДП?
- База
- Переход
- Порядок обхода
- Где ответ?
- $dp[i]$ — длина НВП с концом в i
- $dp[0] = 1$
- $dp[i] = \max_{j=0}^{i-1} dp[j] + 1$
- По возрастанию i
- $\max(dp[i])$



Наибольшая общая
подпоследовательность



Наибольшая общая подпоследовательность

- Есть последовательность чисел $a_1, a_2, a_3 \dots a_n$
 - Есть последовательность чисел $b_1, b_2, b_3 \dots b_m$
 - Найти их общую подпоследовательность наибольшей длины
-
- $A = 1, 2, 3, 4, 5$
 - $B = 5, 1, 3, 2, 4$
 - $1, 3, 4$
 - $1, 2, 4$

Наибольшая общая подпоследовательность

- Что храним в ДП?
- База
- Переход
- Порядок обхода
- Где ответ?
- $dp[i][j]$ – длина НОП $A[0..i-1]$ и $B[0..j-1]$
- $dp[0][0] = 0$
- $dp[i][j] = \begin{cases} dp[i-1][j-1] + 1, & a[i-1]=b[j-1] \\ \max(dp[i-1][j], dp[i][j-1]) \end{cases}$
- По возрастанию i , По возрастанию j
- $dp[n][m]$

Задача о рюкзаке



Задача о рюкзаке

- Рюкзак грузоподъемностью W
- n предметов
 - вес w_i
- Набрать в рюкзак предметы, что:
 - $\sum w_{k_j} < W$
 - $\sum w_{k_j} \rightarrow \max$



Задача о рюкзаке

- Что храним в ДП?
- База
- Переход
- Порядок обхода
- Где ответ?
- $dp[i][j]$ — можно ли из первых i предметов набрать суммарный вес j
- $dp[0][j] = \text{False}, dp[0][0] = \text{True}$
- $dp[i][j] = dp[i-1][j] \text{ or } dp[i-1][j - w_i]$
- По возрастанию i , По возрастанию j
- $\max j: dp[n][j] = \text{True}$



Задача о рюкзаке

- Рюкзак грузоподъемностью W
- n предметов
 - **СТОИМОСТЬ c_i**
 - **ВЕС w_i**
- Набрать в рюкзак предметы, что:
 - $\sum w_{k_j} < W$
 - **$\sum c_{k_j} \rightarrow \max$**

Задача о рюкзаке

- Что храним в ДП?
- База
- Переход
- Порядок обхода
- Где ответ?
- $dp[i][j]$ — максимальная стоимость, с которой можно из первых i предметов набрать суммарный вес j
- $dp[0][j] = -INF, dp[0][0] = 0$
- $dp[i][j] = \max(dp[i-1][j], dp[i-1][j - w_i] + c_i)$
- По возрастанию i , По возрастанию j
- $\max(dp[n][j])$



Bce!