动态规划入门

主讲人：王凯祺
Africamonkey

2017 年 12 月 8 日

动态规划入门

主讲人介绍

我是谁？我不认识这个人！

动态规划入门

什么是动态规划

一个简单的例子：斐波那契数列

动态规划入门

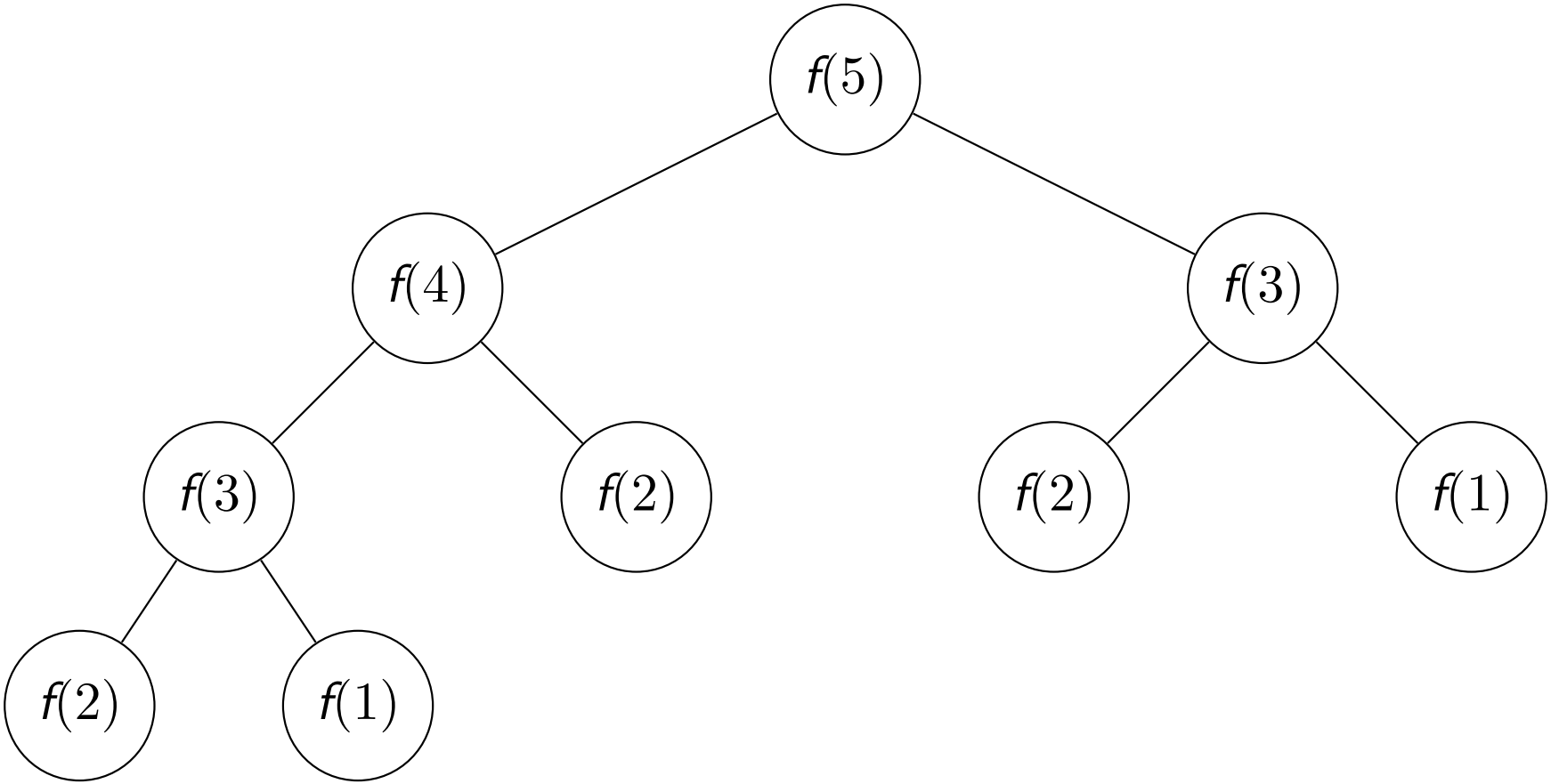
什么是动态规划

一个简单的例子：斐波那契数列

```
int f(int n) {
    if (n <= 2) return 1;
    else return f(n - 1) + f(n - 2);
}
```

动态规划入门

什么是动态规划



我们发现，这样计算 $f(n)$ 需要执行 $f(n)$ 次加法运算。

关键点：减少重复计算

动态规划入门

什么是动态规划

把计算过的 $f(n)$ 保存起来

那么计算 $f(n)$ 将从原来的 $f(n)$ 次加法运算变为 n 次加法运算。

如何修改呢？

动态规划入门

什么是动态规划

把计算过的 $f(n)$ 保存起来

那么计算 $f(n)$ 将从原来的 $f(n)$ 次加法运算变为 n 次加法运算。

如何修改呢？

```
int a[N];
int f(int n) {
    if (a[n] > 0) return a[n];
    if (n <= 2) return 1;
    else {
        a[n] = f(n - 1) + f(n - 2);
        return a[n];
    }
}
```

动态规划入门

什么是动态规划

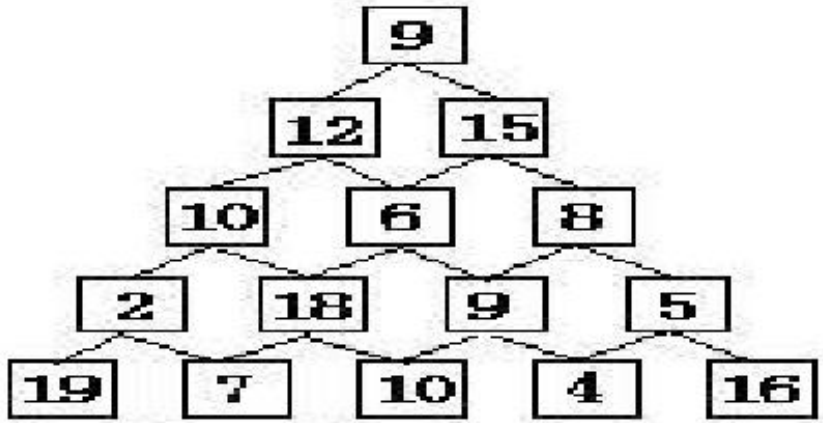
那么动态规划的概念就很清晰啦！

动态规划是一种**以空间换时间**的做法。动态规划保存了**子问题的结果**，并利用子问题的结果求出原问题的结果。

动态规划入门

用动态规划法解题

先看一个问题：数字三角形问题。



图中给出一个数字三角形宝塔。数字三角形中的数字为不超过 100 的正整数。现在规定从最顶层走到最底层，每一步可沿左斜线向下走或者沿右斜线向下走。假设三角形行数不超过 100 ，求一条从最顶层走到最底层的一条路径，使得该路径经过的数字总和最大。

动态规划入门

用动态规划法解题

暴力搜索：

```
void dfs(int x, y, sum) {
    if (x == n) {
        ans = sum;
        return;
    }
    dfs(x + 1, y, sum + a[x + 1][y]);
    dfs(x + 1, y + 1, sum + a[x + 1][y + 1]);
}
```

时间复杂度 $O(2^n)$ ，无法承受 $n = 100$ 的数据。

动态规划入门

用动态规划法解题

我们还是可以用刚刚那种优化斐波那契数列的方法来优化上面这段程序。

我们记 $f[i][j]$ 为从第 i 行第 j 个数走到底端的最大数字总和。

那么 $f[i][j] = a[i][j] + \max(f[i+1][j], f[i+1][j+1])$

(初始化数组 f 为无穷小)

```
int dfs(int x, y) {
    if (f[x][y] >= 0) return f[x][y];
    if (x == n) return a[x][y];
    f[x][y] = a[x][y] + max(dfs(x + 1, y), dfs(x + 1, y + 1);
    return f[x][y];
}
```

只需要调用 f(1, 1) 即可得到答案。时间复杂度由 $O(2^n)$ 变成 $O(n^2)$。

动态规划入门

用动态规划法解题

我们还是可以用刚刚那种优化斐波那契数列的方法来优化上面这段程序。

我们记 $f[i][j]$ 为从第 i 行第 j 个数走到底端的最大数字总和。

那么 $f[i][j] = a[i][j] + \max(f[i+1][j], f[i+1][j+1])$

如果用非递归的方式来实现呢？

动态规划入门

用动态规划法解题

我们还是可以用刚刚那种优化斐波那契数列的方法来优化上面这段程序。

我们记 $f[i][j]$ 为从第 i 行第 j 个数走到底端的最大数字总和。

那么 $f[i][j] = a[i][j] + \max(f[i+1][j], f[i+1][j+1])$

如果用非递归的方式来实现呢？

```
for (int i = n; i >= 1; --i)
    for (int j = 1; j <= i; ++j)
        f[i][j] = a[i][j] + max(f[i + 1][j], f[i + 1][j + 1]);
ans = f[1][1];
```

动态规划入门

用动态规划法解题

记 $f[i][j]$ 为到达第 i 行第 j 个数的最大数字总和。

那么 $f[i][j] = a[i][j] + \max(f[i-1][j-1], f[i-1][j])$

答案就是 $\max_{1 \le i \le n} f[n][i]$

动态规划入门

用动态规划法解题

动态规划解题三步骤：

1. 设状态
2. 写状态转移方程
3. 写代码

动态规划入门

典型例题——最长不下降子序列

设有 n 个不相同的整数组成的数列 $a_1, a_2, \cdots, a_n$ 。若存在 $1 \leq i_1 < i_2 < \cdots < i_m \leq n$ 且有 $a_{i_1} \leq a_{i_2} \leq \cdots \leq a_{i_m}$ ，则把 $a_{i_1}, a_{i_2}, \cdots, a_{i_m}$ 称为 $a_1, a_2, \cdots, a_n$ 的长度为 m 不下降子序列。

例如：$3, 18, 7, 14, 55, 10, 12$ 中，最长不下降子序列为 $3, 7, 14, 55$ ，长度为 4 。

$n \leq 1000$

动态规划入门

典型例题——最长不下降子序列

设 f[i] 为必须包含 a_i 这一项的 $a_1, a_2, \cdots, a_i$ 的最长不下降子序列。

动态规划入门

典型例题——最长不下降子序列

设 f[i] 为必须包含 a_i 这一项的 $a_1, a_2, \cdots, a_i$ 的最长不下降子序列。

则：$f[i] = \max_{1 \le j < i, a_j \le a_i} f[j] + 1$

若没有满足条件的 j ，则 $f[i] = 1$ 。

动态规划入门

典型例题——最长不下降子序列

设 f[i] 为必须包含 a_i 这一项的 $a_1, a_2, \cdots, a_i$ 的最长不下降子序列。

则：$f[i] = \max_{1 \le j < i, a_j \le a_i} f[j] + 1$

若没有满足条件的 j，则 $f[i] = 1$。

代码也非常简单：

```
for (int i = 1; i <= n; ++i) {
    f[i] = 1;
    for (int j = 1; j < i; ++j)
        if (a[j] <= a[i]) f[i] = max(f[i], f[j] + 1);
}
```

答案为 $\max_{1 \le i \le n} f[i]$

动态规划入门

典型例题——背包问题

今天我们讲两种背包问题：

01 背包

完全背包

更多背包问题，请参阅《背包问题九讲》

动态规划入门

典型例题——01 背包

一个旅行者有一个最多能容纳 m 公斤的背包，现在有 n 件物品，它们的重量分别为 $w_1, w_2, \cdots, w_n$ ，它们的价值分别为 $c_1, c_2, \cdots, c_n$ 。若每种物品只有 1 件，求旅行者能获得的最大价值。

$n, m \leq 1000$

动态规划入门

典型例题——01 背包

一个旅行者有一个最多能容纳 m 公斤的背包，现在有 n 件物品，它们的重量分别为 $w_1, w_2, \cdots, w_n$ ，它们的价值分别为 $c_1, c_2, \cdots, c_n$ 。若每种物品只有 1 件，求旅行者能获得的最大价值。

$n, m \leq 1000$ 这个题显然可以用暴力搜索的方法对每件物品进行枚举（对每件物品选还是不选），时间复杂度为 $O(2^n)$ 。

动态规划入门

典型例题——01 背包

一个旅行者有一个最多能容纳 m 公斤的背包，现在有 n 件物品，它们的重量分别为 $w_1, w_2, \cdots, w_n$ ，它们的价值分别为 $c_1, c_2, \cdots, c_n$ 。若每种物品**只有 1 件**，求旅行者能获得的最大价值。

$n, m \leq 1000$

动态规划入门

典型例题——01 背包

一个旅行者有一个最多能容纳 m 公斤的背包，现在有 n 件物品，它们的重量分别为 $w_1, w_2, \cdots, w_n$ ，它们的价值分别为 $c_1, c_2, \cdots, c_n$ 。若每种物品**只有 1 件**，求旅行者能获得的最大价值。

$n, m \leq 1000$ 这个题显然可以用暴力搜索的方法对每件物品进行枚举（对每件物品选还是不选），时间复杂度为 $O(2^n)$ 。

动态规划入门

典型例题——01 背包

设 f[i][j] 表示选择了前 i 个物品，已选物品重量总和为 j 的最大价值。

动态规划入门

典型例题——01 背包

设 f[i][j] 表示选择了前 i 个物品，已选物品重量总和为 j 的最大价值。

$f[i][j] = max(f[i-1][j-w[i]] + c[i], f[i-1][j]), j \geq w[i]$

动态规划入门

典型例题——完全背包

一个旅行者有一个最多能容纳 m 公斤的背包，现在有 n 件物品，它们的重量分别为 $w_1, w_2, \cdots, w_n$ ，它们的价值分别为 $c_1, c_2, \cdots, c_n$ 。若每种物品有**无限多件**，求旅行者能获得的最大价值。

$n, m \leq 1000$

动态规划入门

典型例题——完全背包

一个旅行者有一个最多能容纳 m 公斤的背包，现在有 n 件物品，它们的重量分别为 $w_1, w_2, \cdots, w_n$ ，它们的价值分别为 $c_1, c_2, \cdots, c_n$ 。若每种物品有**无限多件**，求旅行者能获得的最大价值。

$n, m \leq 1000$ 这个题显然可以用暴力搜索的方法对每件物品进行枚举（对每件物品选还是不选），时间复杂度为 $O(2^n)$ 。

动态规划入门

典型例题——完全背包

设 f[i][j] 表示选择了前 i 个物品，已选物品重量总和为 j 的最大价值。

动态规划入门

典型例题——完全背包

设 f[i][j] 表示选择了前 i 个物品，已选物品重量总和为 j 的最大价值。

$f[i][j] = max(f[i-1][j-w[i]] + c[i], f[i-1][j], \mathbf{f[i][j - w[i]] + c[i]}), j \geq w[i]$

动态规划入门

练习题

https://vjudge.net/contest/203351

A 题（难度 0）：HDU 2070 Fibbonacci Number

B 题（难度 3）：HDU 2069 Coin Change

C 题（难度 2）：HDU 2563 统计问题

D 题（难度 1）：HDU 2084 数塔

E 题（难度 2）：HDU 1176 免费馅饼

动态规划入门

练习题

11888014	liuhanzhi	B	Accepted	0	1.9	356	C++	2017-12-08 15:16:54
11888010	liuhanzhi	B	Presentation Error			354	C++	2017-12-08 15:16:30
11887282	liuhanzhi	B	Time Limit Exceeded			111	C++	2017-12-08 13:35:11
11887277	liuhanzhi	B	Runtime			134	C++	2017-12-08 13:34:40
11887274	liuhanzhi	B	Compilation Error			153	C++	2017-12-08 13:34:10
11887266	liuhanzhi	B	Time Limit Exceeded			239	C++	2017-12-08 13:33:15
11887250	liuhanzhi	B	Time Limit Exceeded			316	C++	2017-12-08 13:30:48
11887239	liuhanzhi	B	Memory Limit			319	C++	2017-12-08 13:29:31
11887233	liuhanzhi	B	Compilation Error			320	C++	2017-12-08 13:28:46
11887225	liuhanzhi	B	Wrong Answer			321	C++	2017-12-08 13:27:37
11887220	liuhanzhi	B	Compilation Error			323	C++	2017-12-08 13:26:40
11887217	liuhanzhi	B	Time Limit Exceeded			321	C++	2017-12-08 13:25:39
11887215	liuhanzhi	B	Wrong Answer			345	C++	2017-12-08 13:24:55
11887210	liuhanzhi	B	Wrong Answer			360	C++	2017-12-08 13:24:09
11887191	liuhanzhi	B	Output Limit			359	C++	2017-12-08 13:20:34