动态规划专题

手把手一节课教会你动态规划

侯卫东

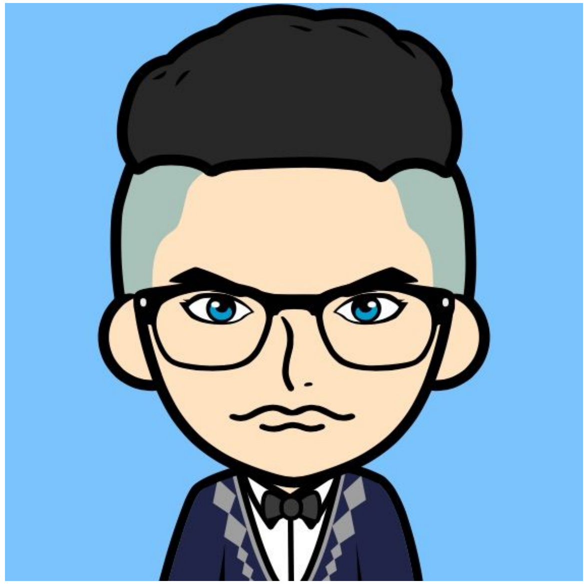
扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
知乎专栏: http://zhuanlan.zhihu.com/jiuzhang
微博: http://www.weibo.com/ninechapter
官网: www.jiuzhang.com

版权声明

九章的所有课程均受法律保护，不允许录像与传播录像
一经发现，将被追究法律责任和赔偿经济损失

讲师&助教介绍



讲师:侯卫东

清华大学毕业，全国算法竞赛金牌得主，参加过ACM国际大学生程序设计竞赛全球总决赛。斩获Google, Facebook, Microsoft, Uber, Dropbox等多家offer。拥有丰富的面试和面试官经验。

助教:

均获得过算法竞赛金奖

刷题超过1000道

新学员问题

- 第一节课错过了怎么办
  - 报名下一期的《动态规划专题》第一节课免费试听即可
- 学员QQ群是什么，怎么加
  - 缴费后，九章账号**我的课程**里有QQ群号
  - 我也会在QQ群里
- 新学员必读常见问题解答
  - http://www.jiuzhang.com/qa/3/

如何使用Webinar?

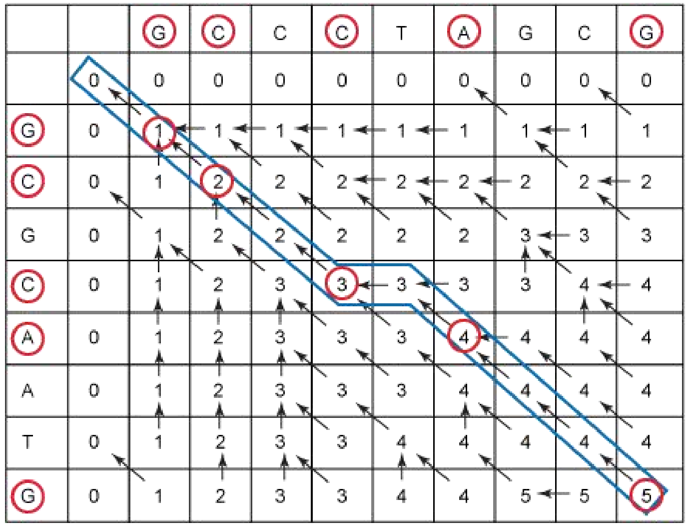
- 可以提问
- 我和助教能看到所有的问题
- 每个同学只能看到自己提到的问题
- 我和助教会选择一些问题让大家都看见

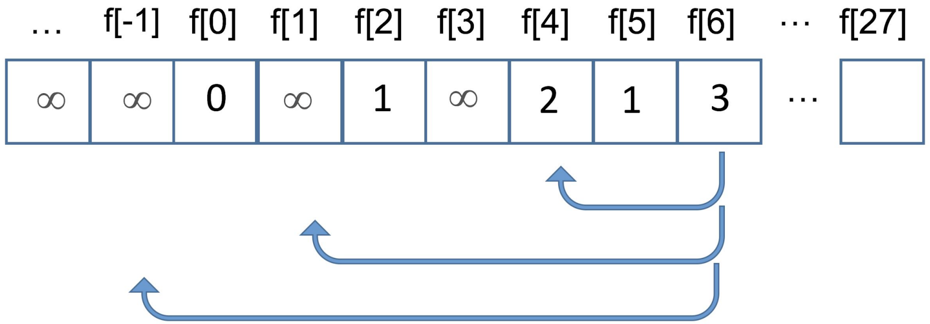
在LintCode上解题

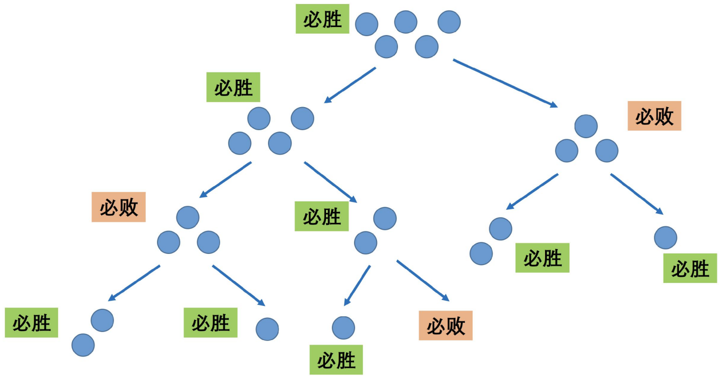
- 网址: www.lintcode.com
- LintCode需要单独先注册一个账户，不要使用九章的账号密码登录
- LintCode解题训练必须先完成上一节课的题目，才能继续下一节课

动态规划

- 科技公司面试必考算法
- 题目类型多，没有固定模板
- 难度属于中上
- 必须掌握

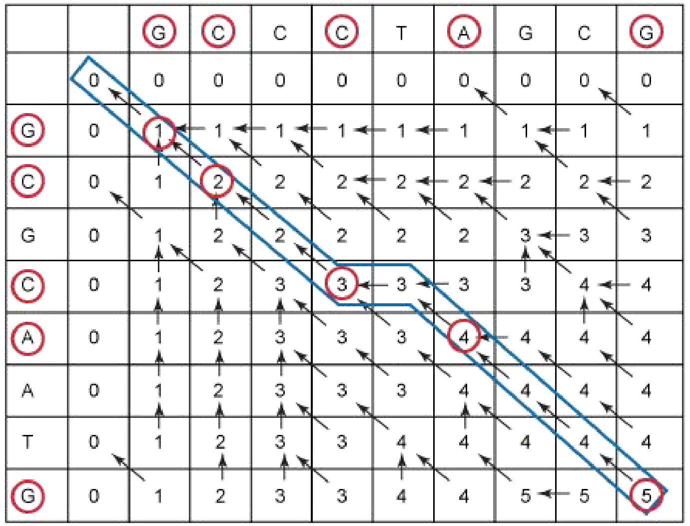


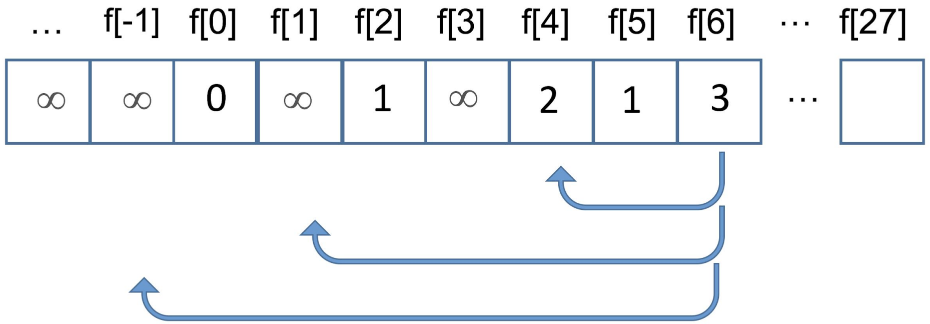


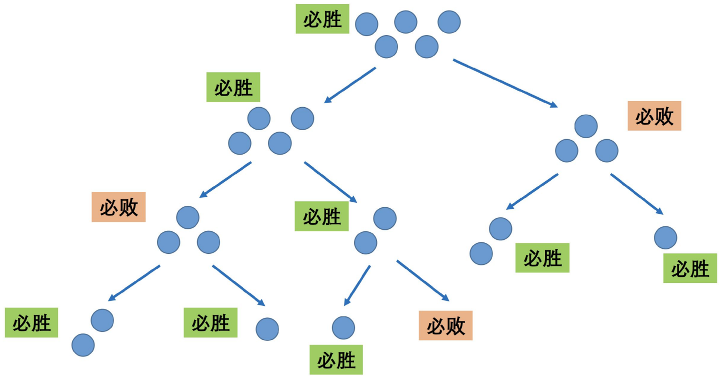


动态规划

- 并没有那么可怕
- 有规律可循
- 掌握其中的思想，举一反三







什么是动态规划

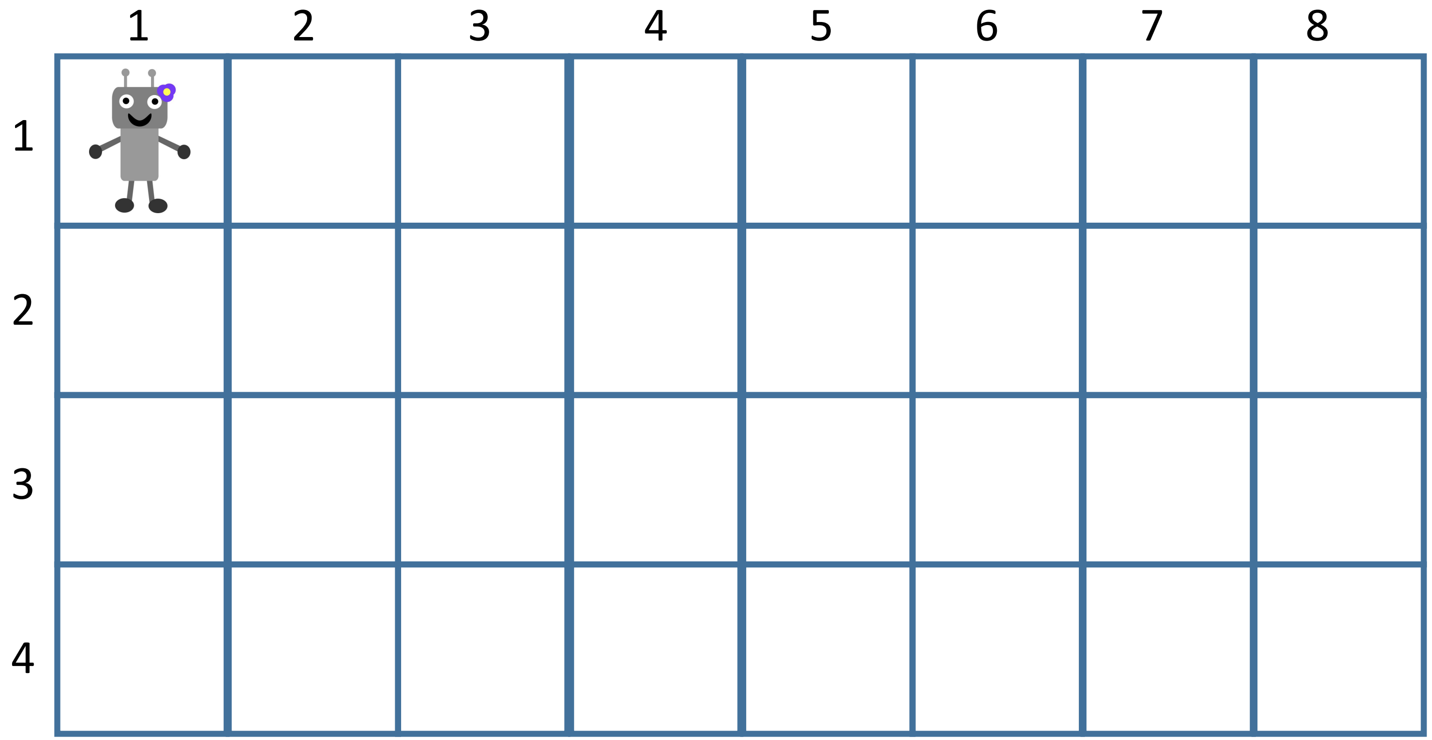
给定一个矩阵网格，一个机器人从左上角出发，每次可以向下或向右走一步

题A：求有多少种方式走到右下角

题B：输出所有走到右下角的路径

动态规划

递归



动态规划题目特点

1. **计数**
   - 有多少种方式走到右下角
   - 有多少种方法选出k个数使得和是Sum

2. **求最大最小值**
   - 从左上角走到右下角路径的最大数字和
   - 最长上升子序列长度

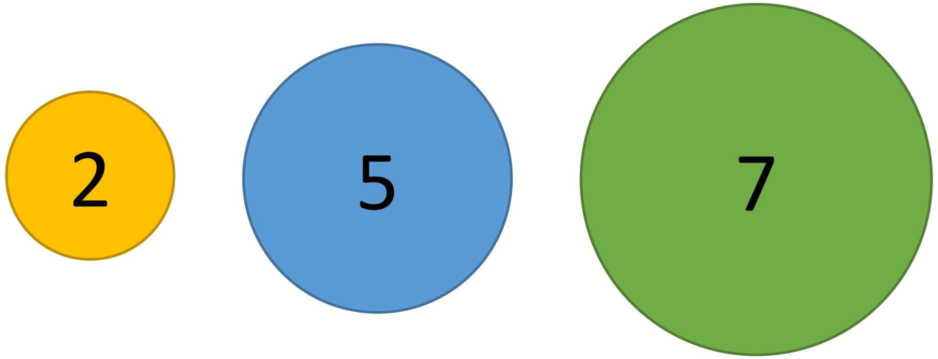
3. **求存在性**
   - 取石子游戏，先手是否必胜
   - 能不能选出k个数使得和是Sum

例题：硬币组合

- 你有三种硬币，分别面值2元，5元和7元，每种硬币都有足够多
- 买一本书需要27元
- 如何用最少的硬币组合正好付清，不需要对方找钱

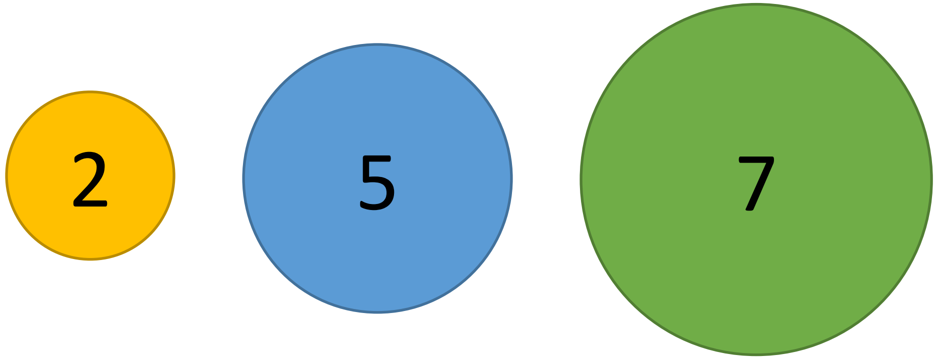
求最大最小值动态规划

直觉



- 最少硬币组合 → 尽量用面值大的硬币
- 7+7+7 = 21
- 21 + 5 = 26
- 呃。。。

直觉



- 改算法：尽量用大的硬币，最后如果可以用一种硬币付清就行
- 7+7+7 = 21
- 21 + 2 + 2 + 2 = 27
- 6枚硬币，应该对了吧。。。

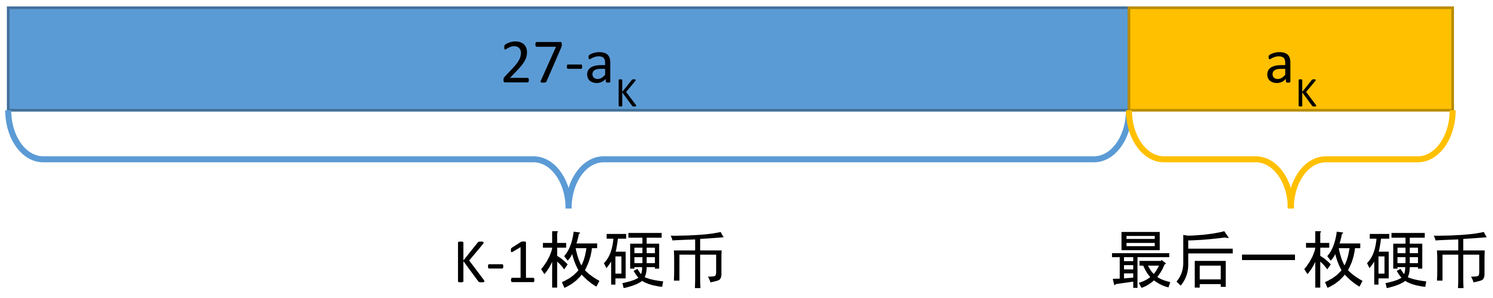
正确答案：7 + 5 + 5 + 5 + 5 = 27，5枚硬币

动态规划组成部分一：确定状态

- 状态在动态规划中的作用属于定海神针
- 简单的说，解动态规划的时候需要开一个数组，数组的每个元素f[i]或者f[i][j]代表什么
  - 类似于解数学题中，X，Y，Z代表什么
- 确定状态需要两个意识：
  - 最后一步
  - 子问题

最后一步

- 虽然我们不知道最优策略是什么，但是最优策略肯定是K枚硬币a_1, a_2,…, a_K 面值加起来是27
- 所以一定有一枚**最后的**硬币: a_K
- 除掉这枚硬币，前面硬币的面值加起来是27- a_K



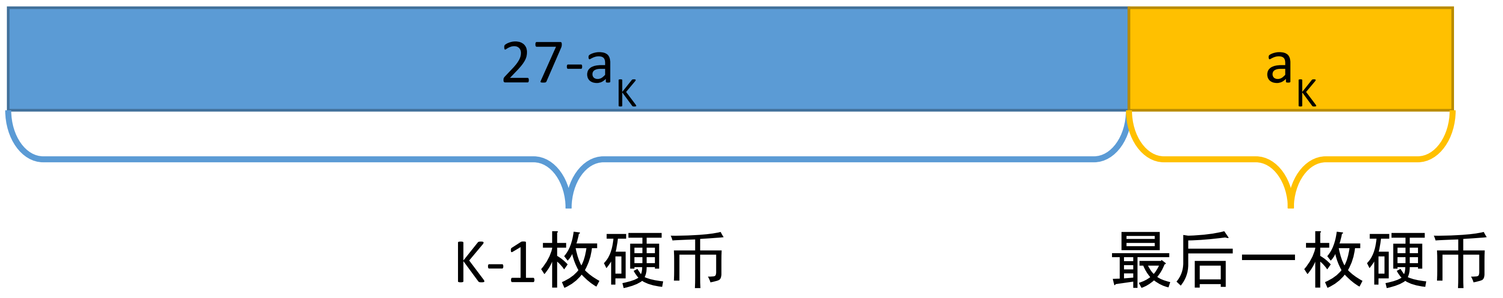
最后一步

关键点1

我们不关心前面的K-1枚硬币是怎么拼出27- a_K 的(可能有1种拼法，可能有100种拼法)，而且我们现在甚至还不知道a_K和K，但是我们确定前面的硬币拼出了27- a_K

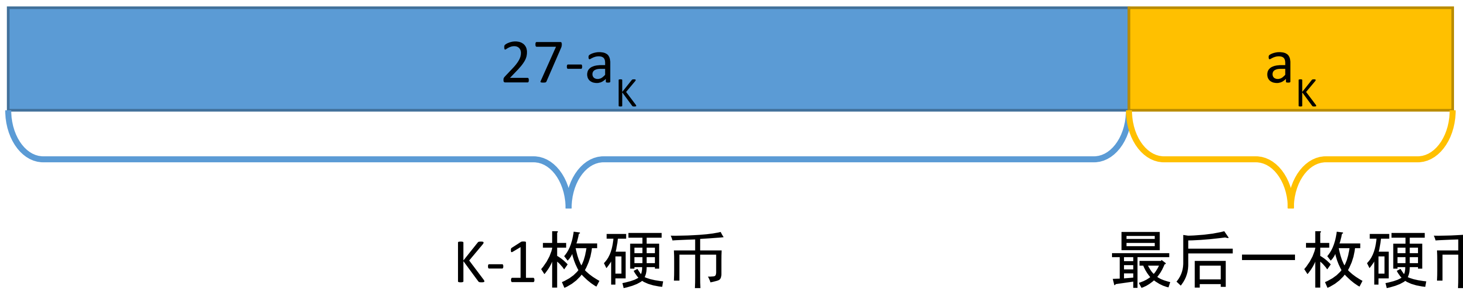
关键点2

因为是最优策略，所以拼出27- a_K 的硬币数一定要最少，否则这就不是最优策略了



子问题

- 所以我们就要求：最少用多少枚硬币可以拼出27- a_K
- 原问题是最少用多少枚硬币拼出27
- 我们将原问题转化成了一个子问题，而且规模更小：27- a_K
- 为了简化定义，我们设状态f(X)=最少用多少枚硬币拼出X



子问题

- 等等，我们还不知道最后那枚硬币a_K是多少
- 最后那枚硬币a_K只可能是2，5或者7
- 如果a_K是2，f(27)应该是f(27-2) + 1 (加上最后这一枚硬币2)
- 如果a_K是5，f(27)应该是f(27-5) + 1 (加上最后这一枚硬币5)
- 如果a_K是7，f(27)应该是f(27-7) + 1 (加上最后这一枚硬币7)
- 除此以外，没有其他的可能了
- 需要求最少的硬币数，所以：

$$f(27) = \min\{f(27-2)+1, f(27-5)+1, f(27-7)+1\}$$

拼出27所需最少的硬币数

拼出25所需最少的硬币数，加上最后一枚硬币2

拼出22所需最少的硬币数，加上最后一枚硬币5

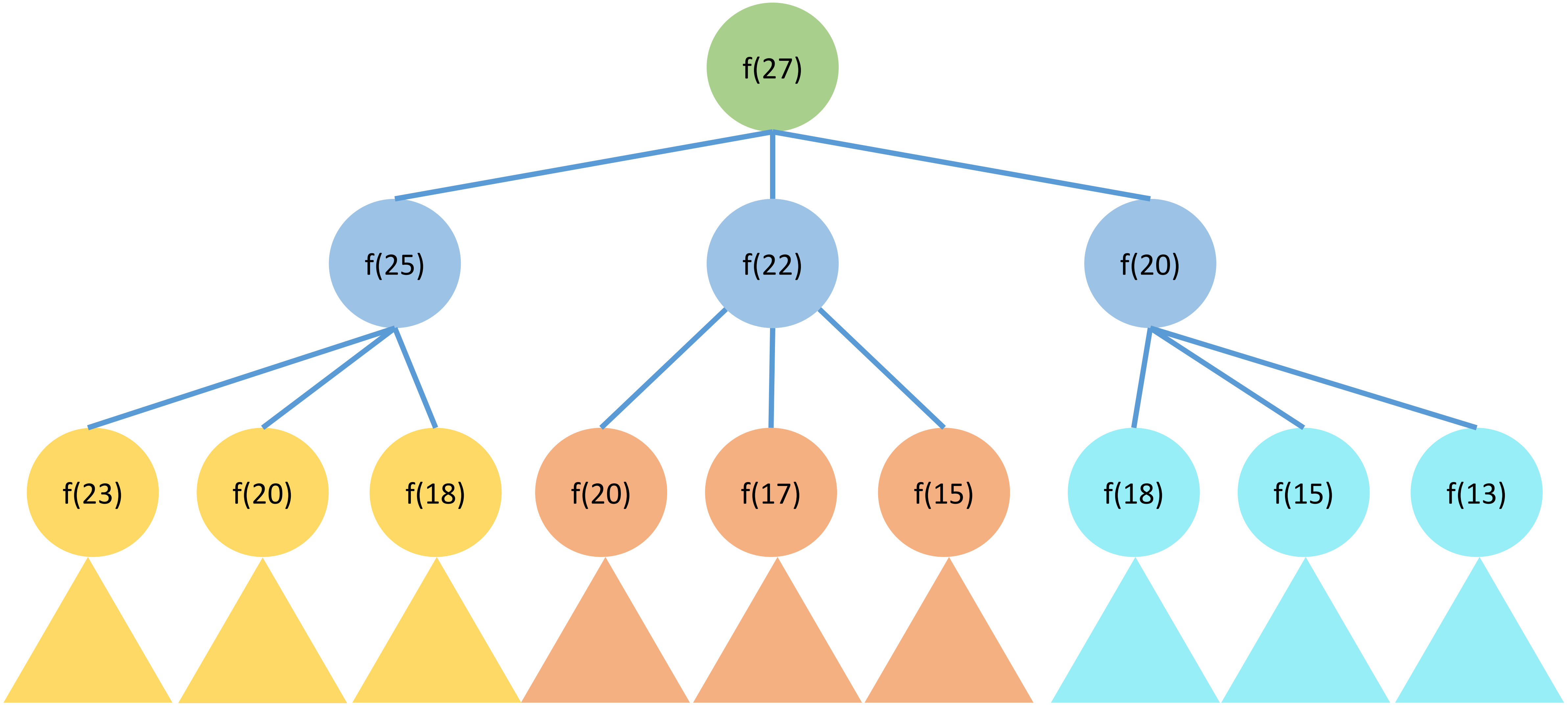
拼出20所需最少的硬币数，加上最后一枚硬币7

递归解法

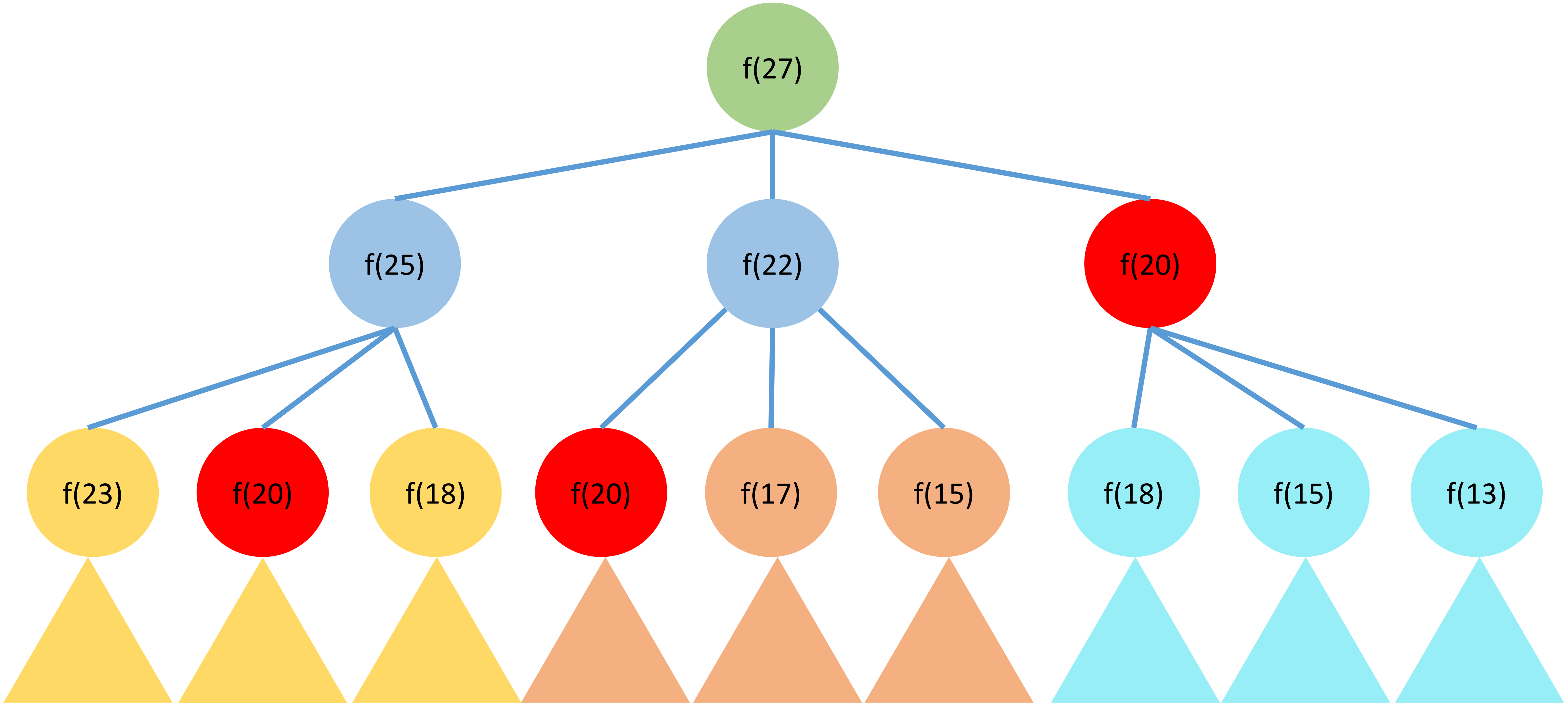
```
int f(int X) {                                   // f(X)=最少用多少枚硬币拼出X
    if (X == 0) return 0;                        // 0元钱只要0枚硬币
    int res = MAX_VALUE;                         // 初始化用无穷大
    if (X >= 2) {                                // 最后一枚硬币是2元
        res = Math.min(f(X – 2) + 1, res);
    }
    if (X >= 5) {                                // 最后一枚硬币是5元
        res =  Math.min(f(X – 5) + 1, res);
    }
    if (X >= 7) {                                // 最后一枚硬币是7元
        res =  Math.min(f(X – 7) + 1, res);
    }
    return res;
}
```

思考：为什么是正无穷？

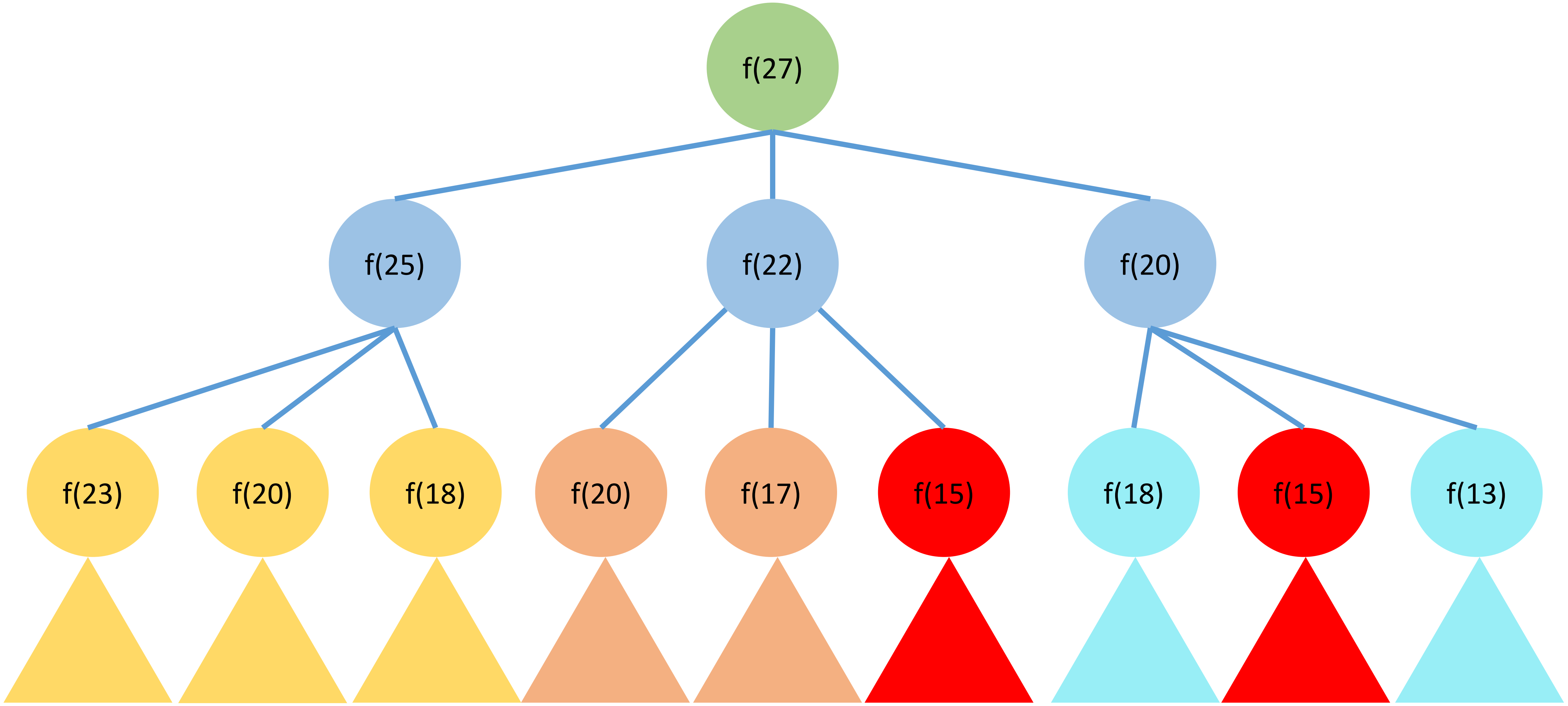
递归解法的问题



递归解法的问题



递归解法的问题



递归解法的问题

- 做了很多重复计算，效率低下
- 如何避免？
- 将计算结果保存下来，并改变计算顺序

动态规划组成部分二：转移方程

- 设状态f[X]=最少用多少枚硬币拼出X
- 对于任意X,

$$f[X] = \min\{f[X-2]+1,\ f[X-5]+1,\ f[X-7]+1\}$$

拼出X所需最少的硬币数

拼出X-2所需最少的硬币数，加上最后一枚硬币2

拼出X-5所需最少的硬币数，加上最后一枚硬币5

拼出X-7所需最少的硬币数，加上最后一枚硬币7

动态规划组成部分三：初始条件和边界情况

- **f[X] = min{f[X-2]+1, f[X-5]+1, f[X-7]+1}**

- 两个问题：X-2, X-5 或者X-7小于0怎么办？什么时候停下来？

- 如果不能拼出Y，就定义f[Y]=正无穷
  – 例如f[-1]=f[-2]=…=正无穷

- 所以f[1] =min{f[-1]+1, f[-4]+1,f[-6]+1}=正无穷，表示拼不出来1

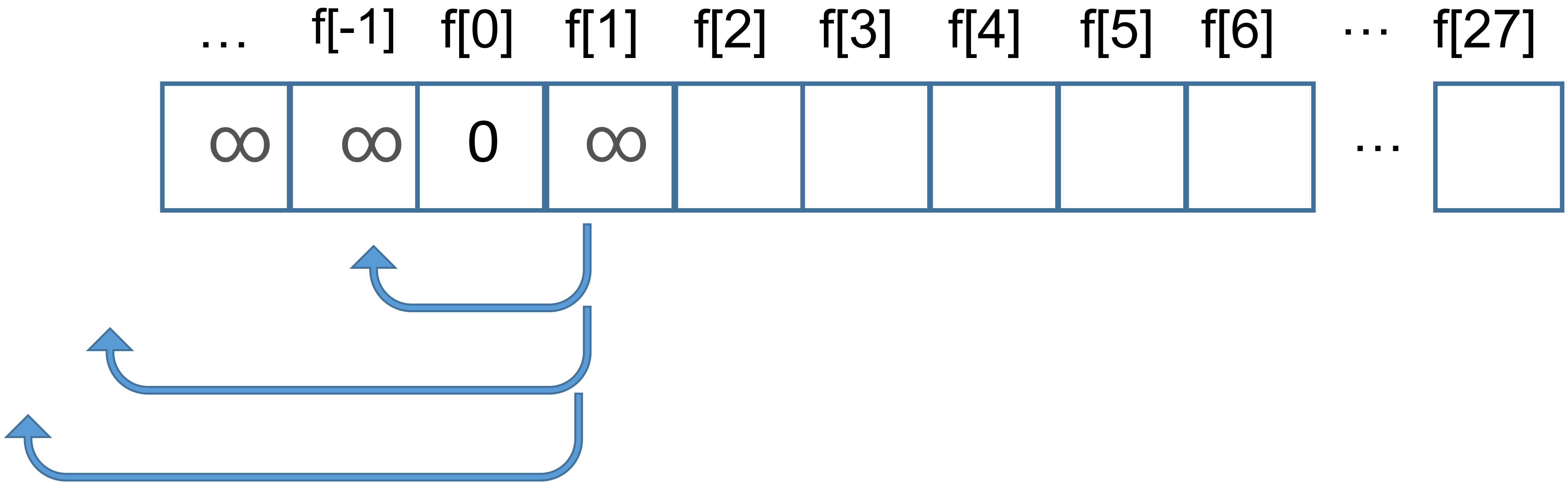
- 初始条件：f[0] = 0

动态规划组成部分四：计算顺序

- **拼出X所需要的最少硬币数：f[X] = min{f[X-2]+1, f[X-5]+1, f[X-7]+1}**
- 初始条件：f[0] = 0
- 然后计算f[1], f[2], …, f[27]
- 当我们计算到f[X]时，f[X-2], f[X-5], f[X-7]都已经得到结果了

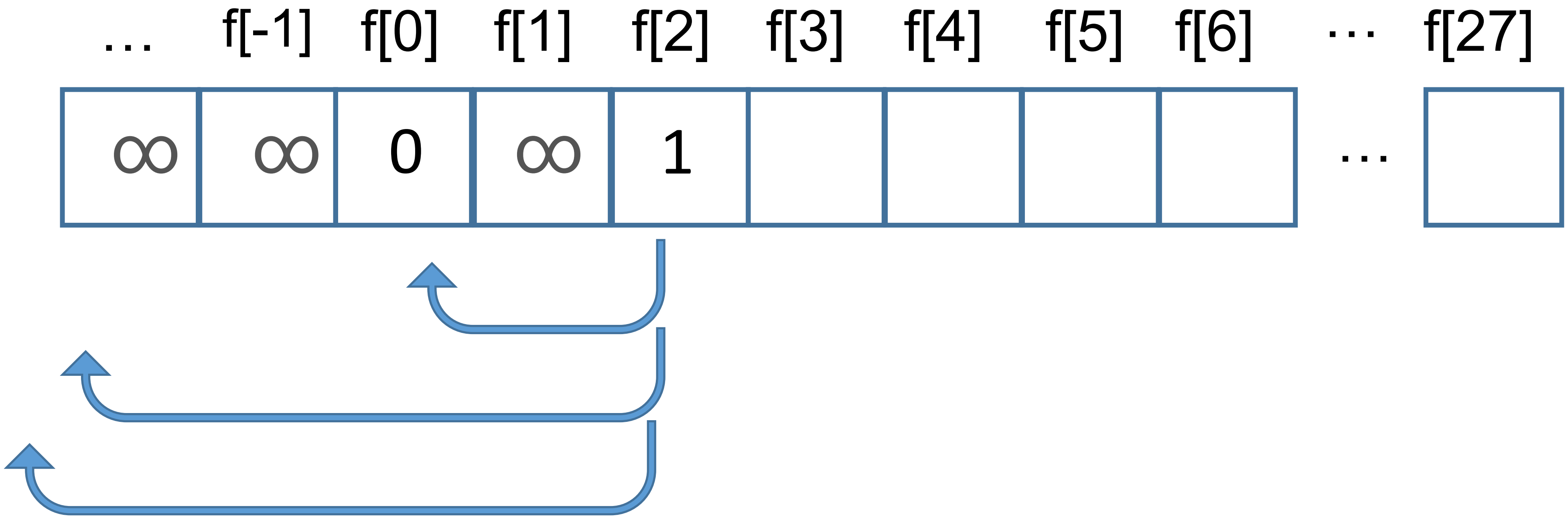
动态规划组成部分四:计算顺序

- f[X] = 最少用多少枚硬币拼出X
- f[X] = ∞ 表示无法用硬币拼出X



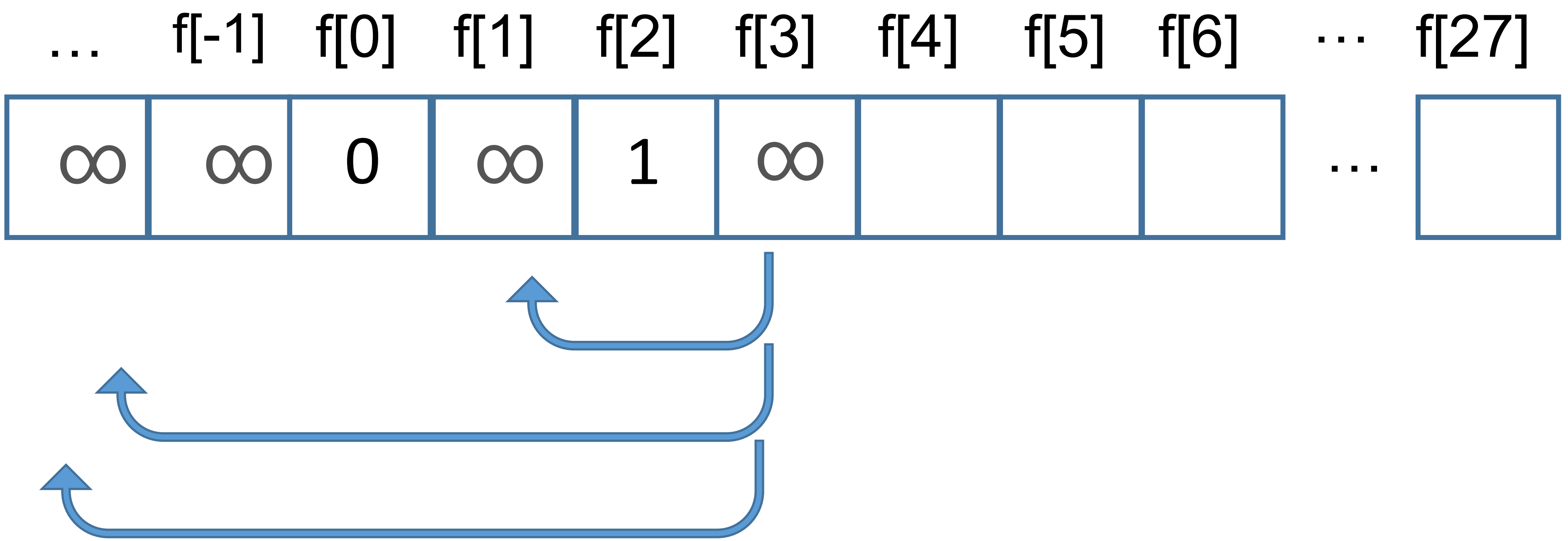
动态规划组成部分四:计算顺序

- f[X] = 最少用多少枚硬币拼出X
- f[X] = ∞ 表示无法用硬币拼出X



动态规划组成部分四：计算顺序

- f[X] = 最少用多少枚硬币拼出X
- f[X] = ∞ 表示无法用硬币拼出X



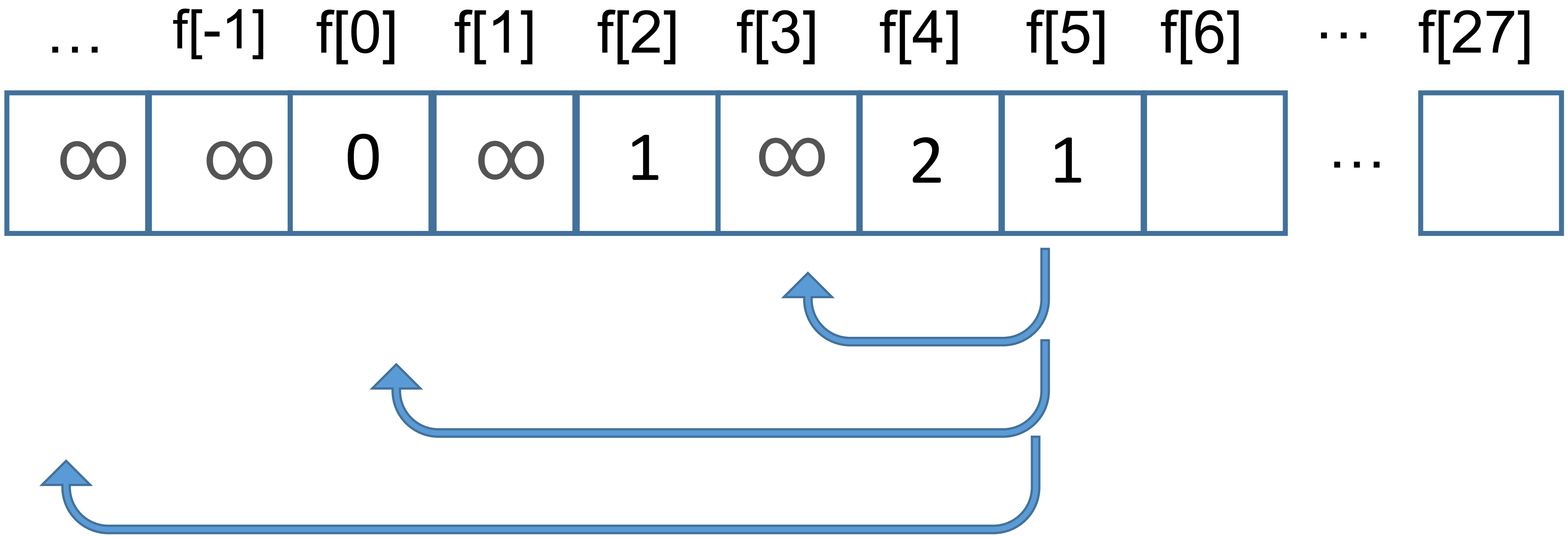
动态规划组成部分四：计算顺序

- f[X] = 最少用多少枚硬币拼出X
- f[X] = ∞ 表示无法用硬币拼出X

…	f[-1]	f[0]	f[1]	f[2]	f[3]	f[4]	f[5]	f[6]	…	f[27]
∞	∞	0	∞	1	∞	2			…	

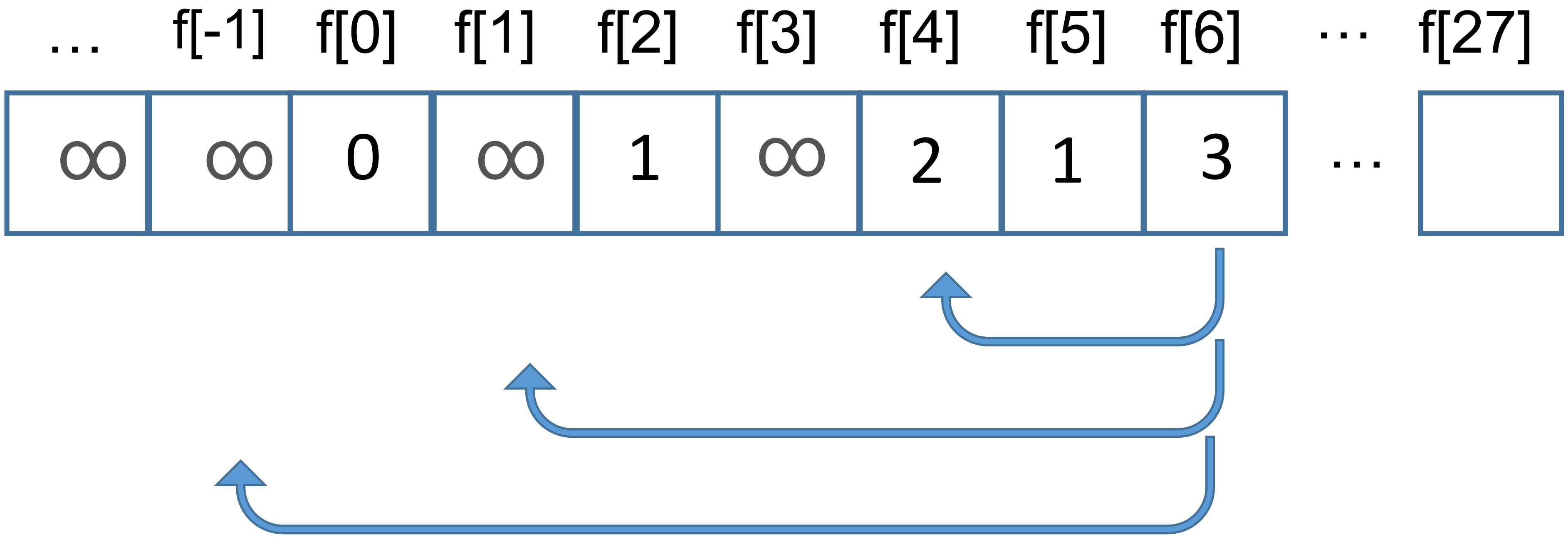
动态规划组成部分四：计算顺序

- f[X] = 最少用多少枚硬币拼出X
- f[X] = ∞ 表示无法用硬币拼出X



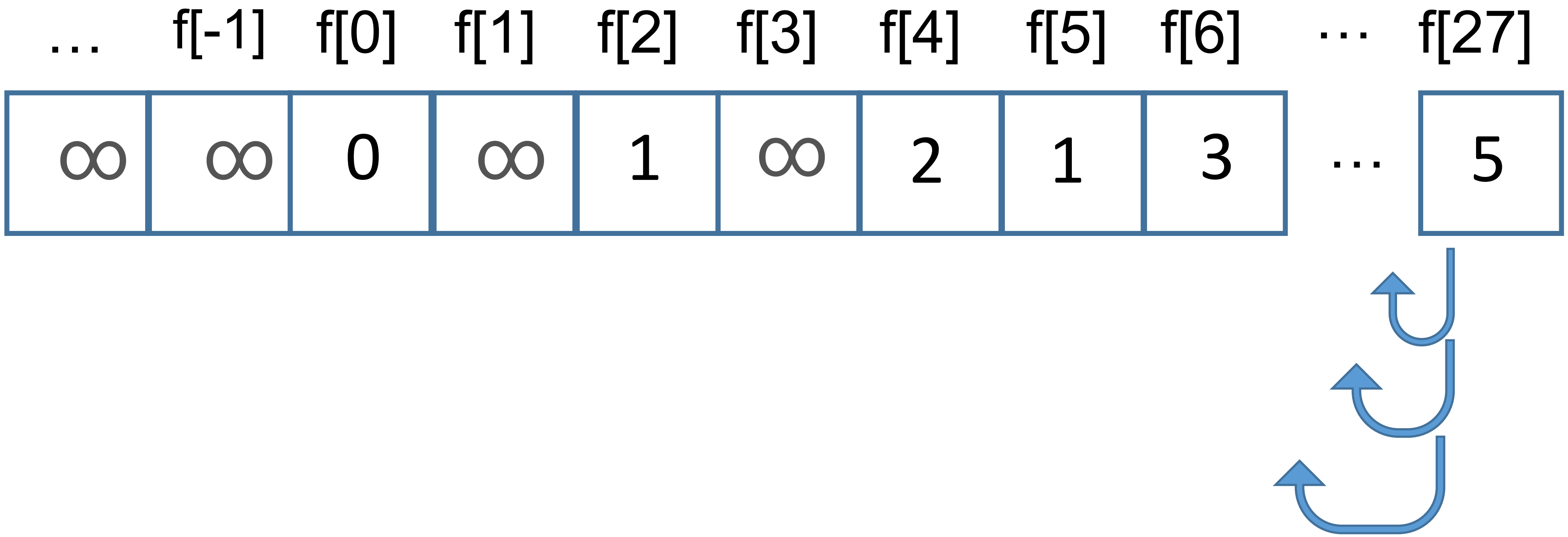
动态规划组成部分四：计算顺序

- f[X] = 最少用多少枚硬币拼出X
- f[X] = ∞ 表示无法用硬币拼出X



动态规划组成部分四：计算顺序

- f[X] = 最少用多少枚硬币拼出X
- f[X] = ∞ 表示无法用硬币拼出X



动态规划组成部分四：计算顺序

- 每一步尝试三种硬币，一共27步
- 与递归算法相比，没有任何重复计算
- 算法时间复杂度(即需要进行的步数)：27 * 3

小结

- 求最值型动态规划

- 动态规划组成部分：
  - 1. 确定状态
    - 最后一步（最优策略中使用的最后一枚硬币a_K）
    - 化成子问题（最少的硬币拼出更小的面值27-a_K）
  - 2. 转移方程
    - f[X] = min{f[X-2]+1, f[X-5]+1, f[X-7]+1}
  - 3. 初始条件和边界情况
    - f[0] = 0, 如果不能拼出Y，f[Y]=正无穷
  - 4. 计算顺序
    - f[0], f[1], f[2], …

- 消除冗余，加速计算

编程

例题

Unique Paths

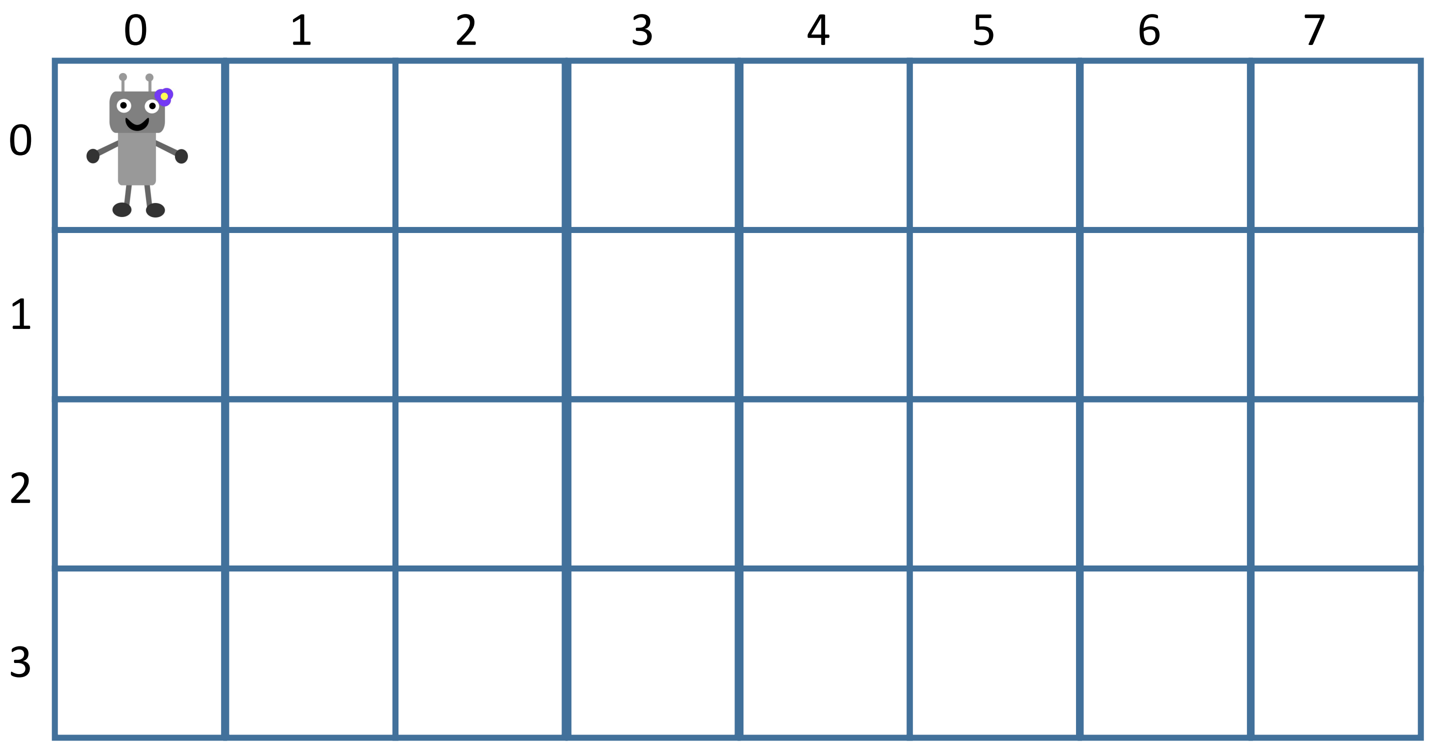
http://www.lintcode.com/en/problem/unique-paths/

http://www.jiuzhang.com/solutions/unique-paths/

LintCode 114: Unique Paths

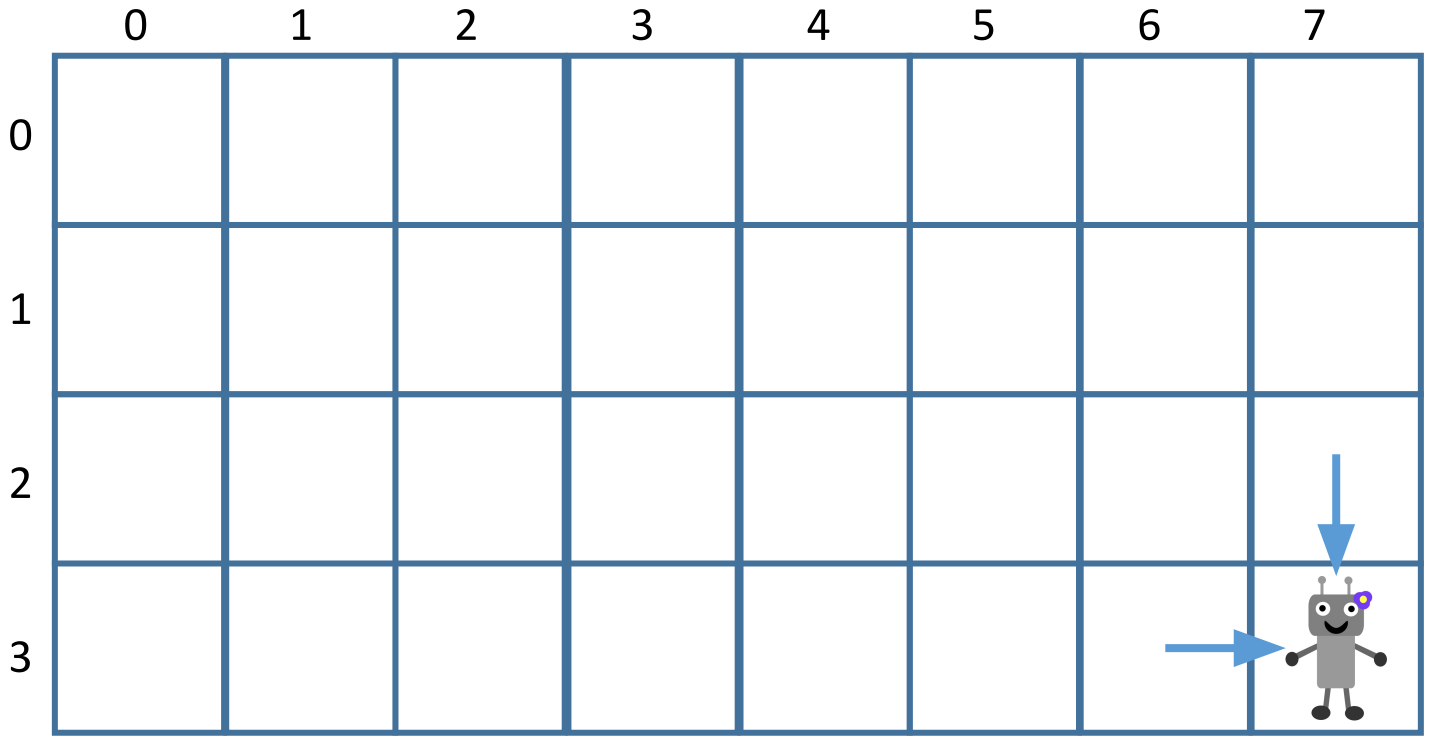
- 题意:
- 给定m行n列的网格，有一个机器人从左上角(0,0)出发，每一步可以向下或者向右走一步
- 问有多少种不同的方式走到右下角

计数型动态规划



动态规划组成部分一:确定状态

- 最后一步:无论机器人用何种方式到达右下角，总有最后挪动的一步:
  - 向右 或者 向下
- 右下角坐标设为(m-1, n-1)
- 那么前一步机器人一定是在(m-2, n-1)或者(m-1, n-2)



子问题

- 那么，如果机器人有**X**种方式从左上角走到(m-2,n-1)，有**Y**种方式从左上角走到(m-1,n-2)，则机器人有**X+Y**种方式走到(m-1, n-1)

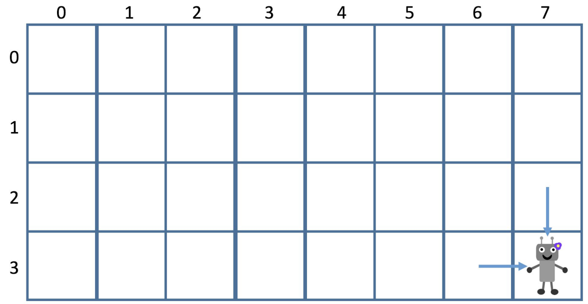
思考：为什么是X+Y

- 问题转化为，机器人有多少种方式从左上角走到(m-2, n-1)和(m-1, n-2)

- 原题要求有多少种方式从左上角走到(m-1, n-1)

- 子问题

- 状态：设f[i][j]为机器人有多少种方式从左上角走到(i, j)

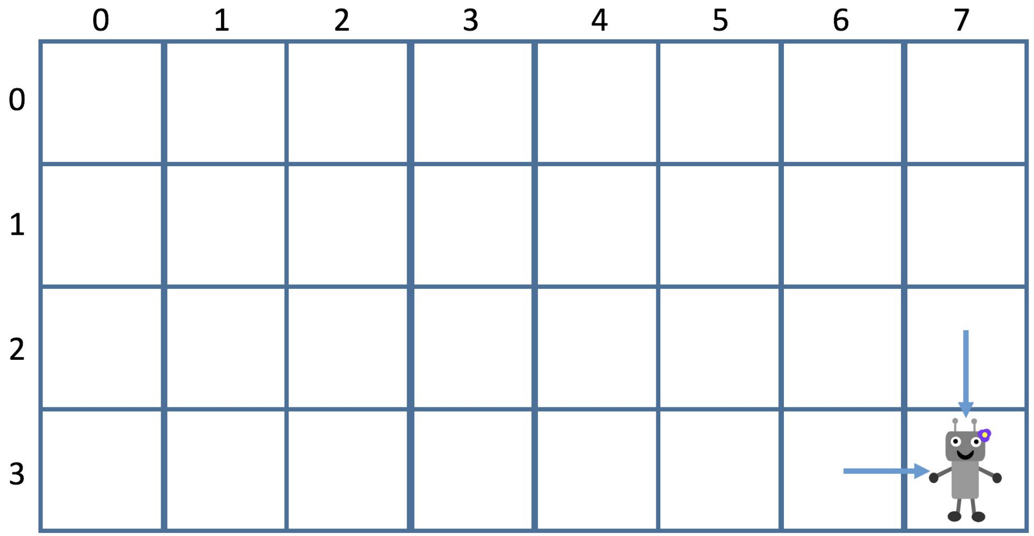


动态规划组成部分二：转移方程

- 对于任意一个格子(i, j)

$$f[i][j] = f[i-1][j] + f[i][j-1]$$

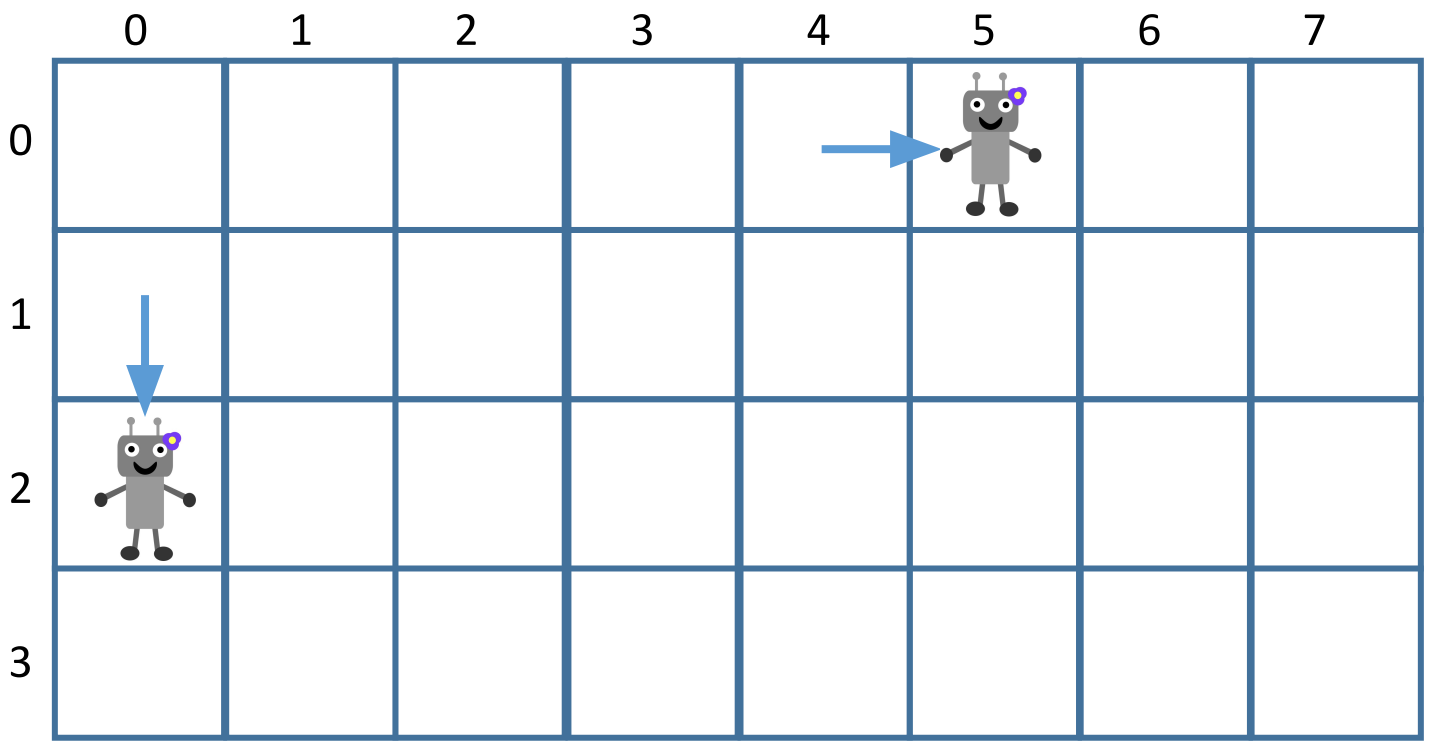
机器人有多少种方式走到(i, j)	机器人有多少种方式走到(i-1, j)	机器人有多少种方式走到(i, j-1)



动态规划组成部分三:初始条件和边界情况

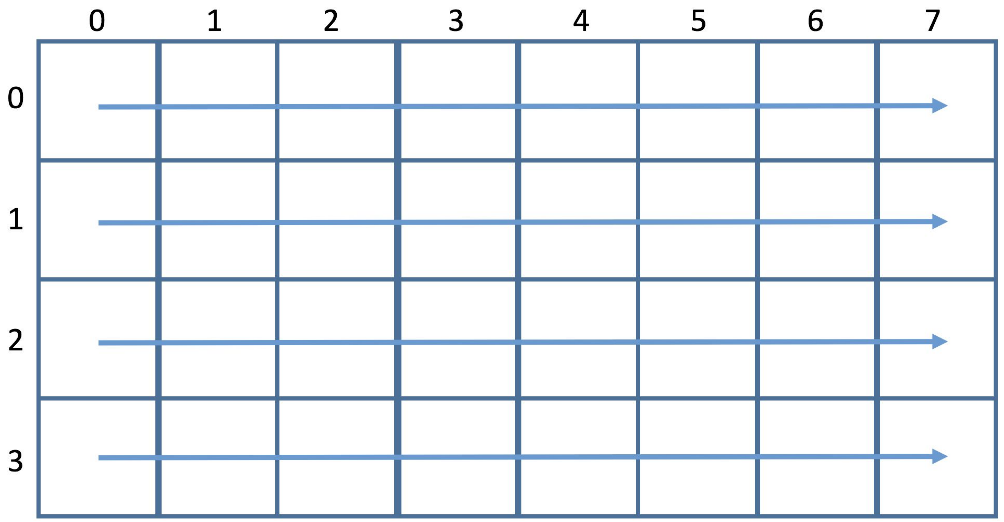
- 初始条件:f[0][0] = 1, 因为机器人只有一种方式到左上角(什么都不做)

- 边界情况:i = 0 或 j = 0, 则前一步只能有一个方向过来



动态规划组成部分四：计算顺序

- f[0][0] = 1
- 计算第0行：f[0][0], f[0][1], …, f[0][n-1]
- 计算第1行：f[1][0], f[1][1], …, f[1][n-1]
- …
- 计算第m-1行：f[m-1][0], f[m-1][1], …, f[m-1][n-1]
- 答案是f[m-1][n-1]
- 时间复杂度（计算步数）：O(MN)，空间复杂度（数组大小）：O(MN)



编程

Jump Game

http://www.lintcode.com/en/problem/jump-game/

http://www.jiuzhang.com/solutions/jump-game/

LintCode 116 Jump Game

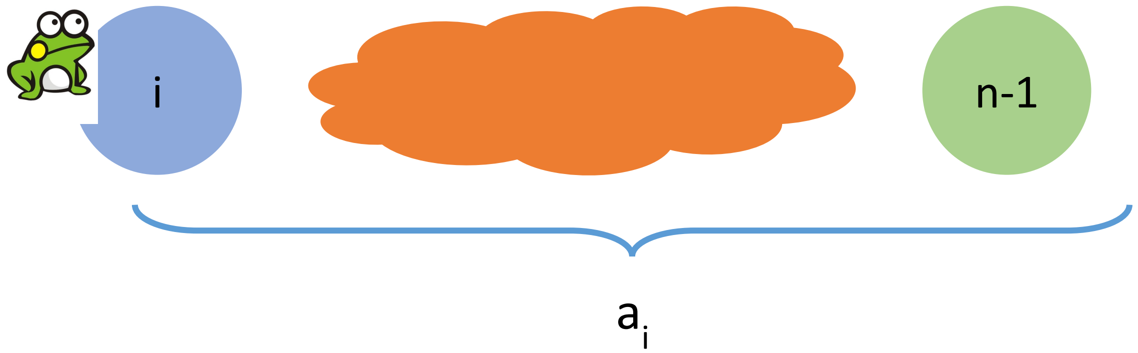
- 有n块石头分别在x轴的0, 1, …, n-1位置
- 一只青蛙在石头0，想跳到石头n-1
- 如果青蛙在第i块石头上，它最多可以向右跳距离a_i
- 问青蛙能否跳到石头n-1

- 例子：
- 输入：a=[2, 3, 1, 1, 4]
- 输出：True
- 输入：a=[3, 2, 1, 0, 4]
- 输出：False

存在型动态规划

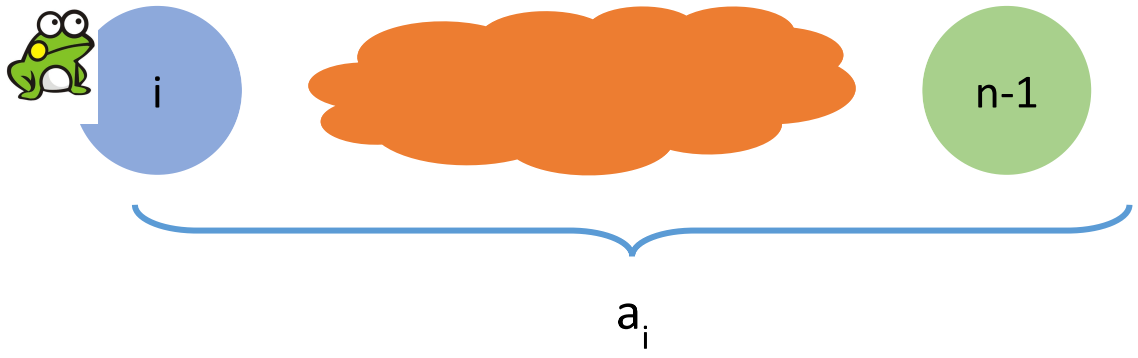
动态规划组成部分一：确定状态

- 最后一步：如果青蛙能跳到最后一块石头n-1，我们考虑它跳的最后一步
- 这一步是从石头i跳过来，i<n-1
- 这需要两个条件同时满足：
  - 青蛙可以跳到石头i
  - 最后一步不超过跳跃的最大距离：$n-1-i<=a_i$



子问题

- 那么，我们需要知道青蛙能不能跳到石头i (i<n-1)
- 而我们原来要求青蛙能不能跳到石头n-1
- 子问题
- 状态：设f[j]表示青蛙能不能跳到石头j

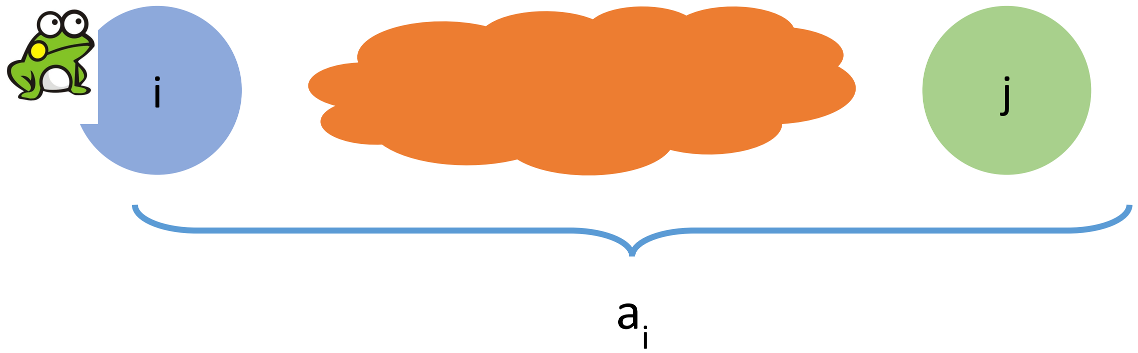


动态规划组成部分二:转移方程

- 设f[j]表示青蛙能不能跳到石头j

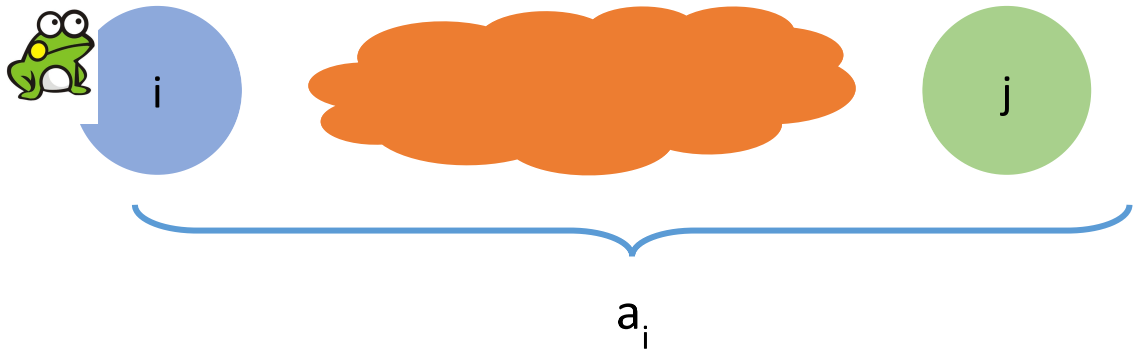
$$f[j] = OR_{0<=i<j}(f[i] \text{ AND } i + a[i] >= j)$$

f[j]	$OR_{0<=i<j}$	f[i]	i + a[i] >= j
青蛙能不能跳到石头j	枚举上一个跳到的石头i	青蛙能不能跳到石头i	最后一步的距离不能超过a_i



动态规划组成部分三：初始条件和边界情况

- 设f[j]表示青蛙能不能跳到石头j
- 初始条件：f[0] = True，因为青蛙一开始就在石头0



动态规划组成部分四：计算顺序

- 设f[j]表示青蛙能不能跳到石头j
- $f[j] = OR_{0<=i<j}(f[i] \text{ AND } i + a[i] >= j)$
- 初始化f[0]=True
- 计算f[1], f[2], …, f[n-1]
- 答案是f[n-1]
- 时间复杂度：$O(N^2)$，空间复杂度（数组大小）：$O(N)$

编程

动态规划入门总结

九章算法

四个组成部分

- 确定状态
  - 研究最优策略的最后一步
  - 化为子问题
- 转移方程
  - 根据子问题定义直接得到
- 初始条件和边界情况
  - 细心，考虑周全
- 计算顺序
  - 利用之前的计算结果

课程FAQ

- 是否涵盖所有的动态规划考题类型
  - 是

- 常见动态规划类型
  - 坐标型动态规划 (20%) 重点
  - 序列型动态规划 (20%) 重点
  - 划分型动态规划 (20%) 重点
  - 区间型动态规划 (15%) 重点
  - 背包型动态规划 (10%)
  - 拓扑型动态规划 (5%)
  - 博弈型动态规划 (5%)
  - 综合性动态规划 (5%)

课程FAQ

- 我需要什么基础才可以上这个班
  – 写一门基础语言，写过二三十道题，想对动态规划有透彻了解
- 上完这门课我能学到什么
  – 对于面试中常见动态规划题目能迅速判断并找到解题要领
  – 对于动态规划变种题能找到解题的突破口并轻松解决
  – 可以对动态规划算法进行时间和空间上的优化
  – 面试中将不再存在你不会做的动态规划题

课程安排

第一讲：动态规划入门

第二讲：坐标型动态规划

第三讲：序列型动态规划

第四讲：划分及拓扑型动态规划

第五讲：区间和背包型动态规划

第六讲：难题专场一

第七讲：难题专场二和总结

课表

- Link

时间

- 美东时间
- 美西时间
- 北京时间

为什么要报名上直播课

- 内容总是最新
  - 结合实时面试趋势
  - 讲解实时热门真题
- 每周定时定量，起到督促作用
  - 克服懒惰心理
- 学习积极性更高
- 讲师助教实时答疑
  - 及时清扫障碍

你可以获得哪些学员权限

- LintCode专属阶梯训练题
- 九章QA发问权限
  - 助教老师100%回答
- 九章QA课程与内推板块浏览权限
  - 最新最热面试题面经实时分享
  - 让九章老学员帮你内推各大公司
- 九章课程QQ群
  - 与同学们实时交流学习问题
  - 随时@老师@助教答疑解惑
  - 认识更多志同道合的朋友，一起打鸡血
  - 学员线下活动（自行组织）

付款方式？

九章官网登录→我的课程
付费之后即可开启LintCode阶梯训练权限，有效期一年
使用支付宝的同学请至少提前1小时付款，否则可能耽误上课

优惠码的获得？

关注微信“九章算法”
点击右下角“课程优惠”按照提示操作

版权声明

九章的所有课程均受法律保护，不允许录像与传播录像
一经发现，将被追究法律责任和赔偿经济损失

谢谢！

请提问