Design and Analysis of Algorithms

Tutorial 10: Dynamic Programming

童咏昕

北京航空航天大学 计算机学院

yxtong@buaa. edu. cn

问题1

- 现有k种面额的硬币（数量无限，且一定包含面额为1元的硬币），使用硬币组成共计n元的面额，请设计算法计算所需的最少的硬币数。请使用时间复杂度为O(nk)的动态规划算法。

问题1-提示

- 令c[j]表示组成j元的面额所需的最少硬币数，令所给的k种硬币的面额分别为$d_1, d_2, \ldots, d_k$.
- 由于一定有面额为1元的硬币，因此对于任意的j ≥ 0，一定能找到一种组成方式。

问题1-提示

- 如果某一种组成j元的面额的方式中包括了硬币d_i，则有$c[j] = 1 + c[j - d_i]$, 枚举d_i, 选择其中最小的硬币数，即可得到最终的c[j].

$$c[j] = \begin{cases} \infty, & j < 0 \\ 0, & j = 0 \\ 1 + \min\limits_{1 \le i \le k} \{c[j - d_i]\}, & j > 0 \end{cases}$$

问题1-提示

- 下图为算法的伪代码，按照j由小到大的顺序计算c[j]，同时不再定义c[j](j<0)，而是在计算$c[j-d_i]$前先判断j与d_i的关系。算法同时产生了数组denom[1..n]，表示组成j元面额的最优方式中包括面额为denom[j]的硬币。

Compute-Change(n, d, k)

```
Input: k kinds of coins with denominations d make up n dollars.
Output: Minimum number of used coins.
Let c[1..n] and denom[1..n] are two new arrays;
for j ← 1 to n do
    c[j] ← ∞;
    for i ← 1 to k do
        if j ≥ d_i and 1 + c[j − d_i] < c[j] then
            c[j] ← 1 + c[j − d_i]; denom[j] ← d_i;
        end
    end
end
return c,denom;
```

问题1-提示

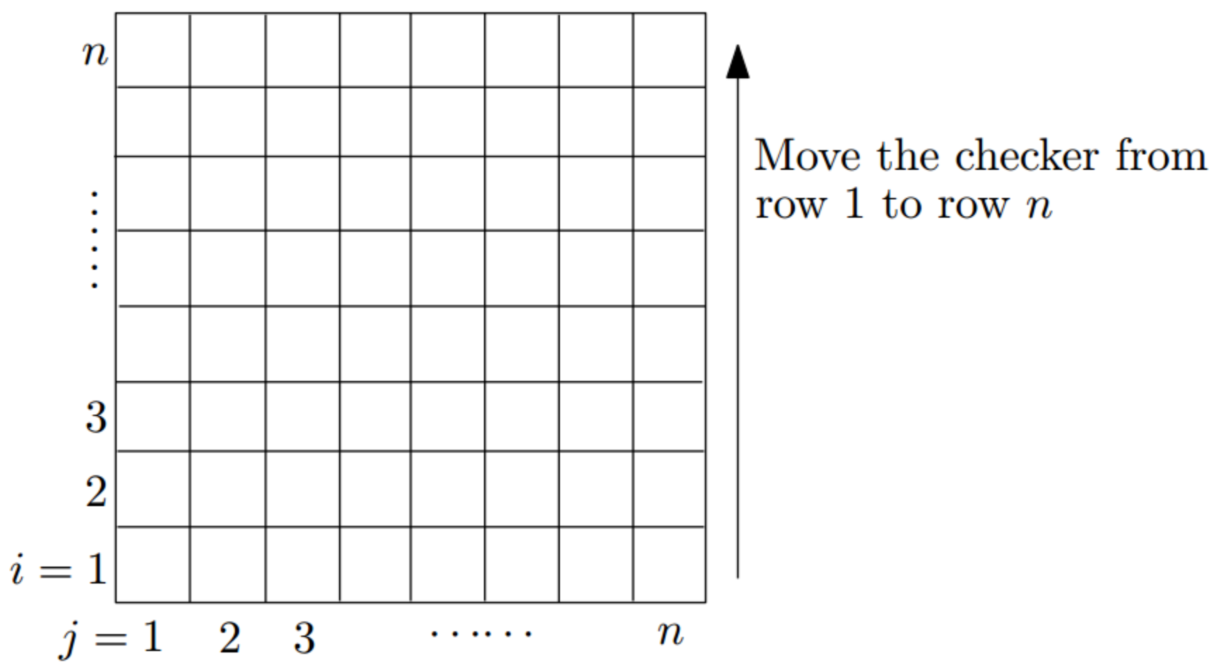
- 下面的算法Give-Change使用数组denom输出组成j元面额的最优方式的各硬币面额。

Give-Change(j, $denom$)

Input: $\boldsymbol{denom}$ is the array generated from Compute-Change.
Output: Used coins to make up j dollars.
if $j > 0$ **then**
  Output $denom[j]$;
  Give-Change($j - denom[j]$,$denom$);
end

问题2

- 现有一个nxn的棋盘与一个棋子，你需要将棋子从最下面一行移动到最上面一行，棋子的每一步移动限制在下面三种形式之一。
  - 将棋子从当前格移动到其上面一格。
  - 将棋子从当前格移动到其左上一格（如果存在的话）。
  - 将棋子从当前格移动带其右上一格（如果存在的话）。



问题2

- 每当将棋子从方格（i，j）移动到(i',j')时，你将获得$p((i,j),(i',j'))$元奖励，格间移动对应的奖励p为已知，并假设每一步奖励都为正数。
- 请给出一个算法计算将棋子从最下面一行移动到最上面一行所能获得的最大奖励。棋子的起止位置可任意选择。同时请分析算法的时间复杂度。

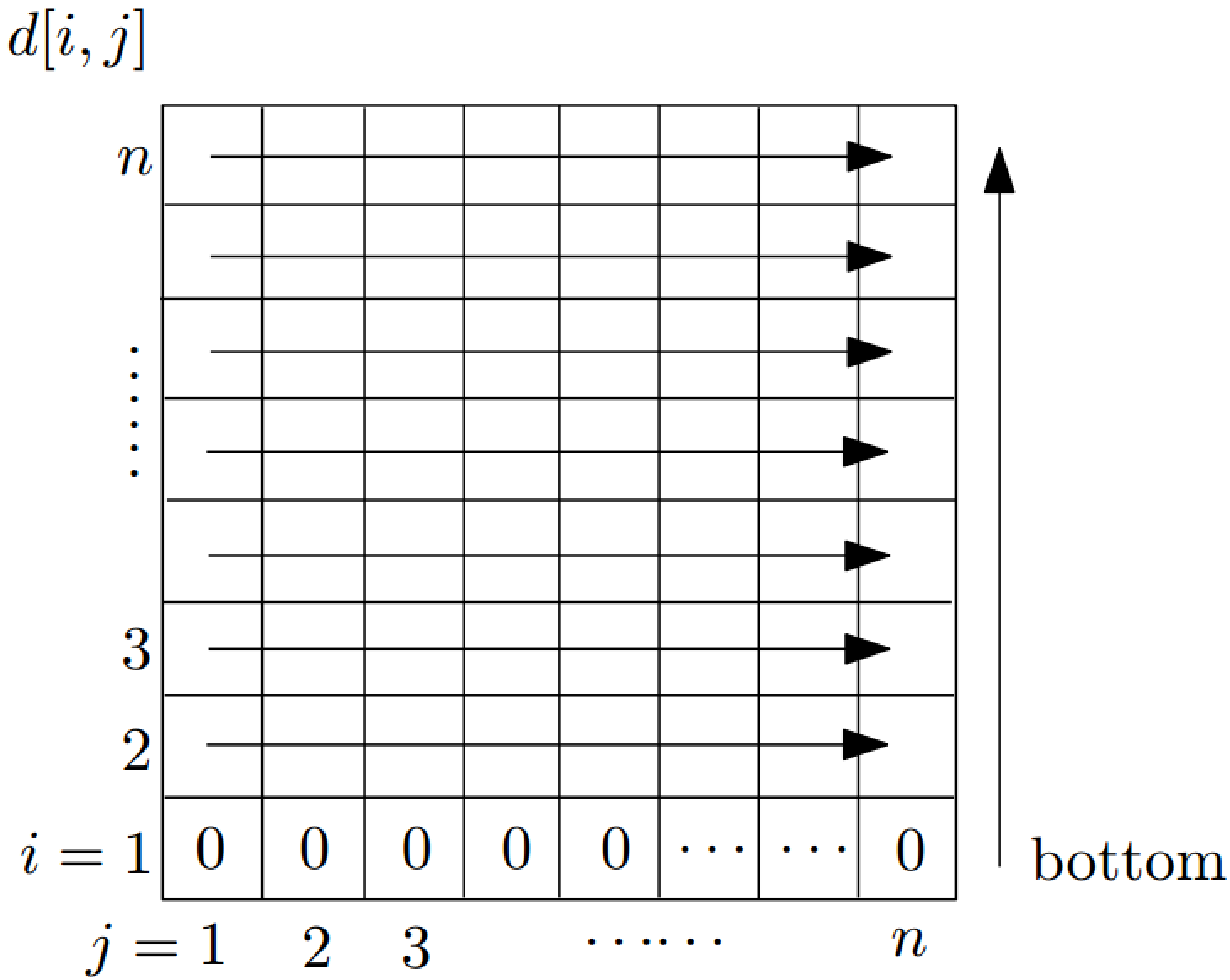
问题2-提示

- 用数对(i, j)表示（从下往上）第i行第j列的方格，$1 \le i,j \le n$. 对于格(i, j), 其一定是由格$(i-1,j')$经过一步移动到达的，其中$j' = j-1, j$或$j+1$. 令d[i, j]为到达格(i, j)所能得到的最大奖励，有如下递推式

$$d[i,j] = max\begin{cases} d[i-1,j-1] + p\big((i-1,j-1),(i,j)\big), j > 1 \\ d[i-1,j] + p((i-1,j),(i,j)), \qquad always \\ d[i-1,j+1] + p\big((i-1,j+1),(i,j)\big), j < n \end{cases}$$

- 按照i由小到达的顺序计算d[i, j], 一旦计算完成，第n行的格中的最大奖励即为所求 $\max_{1\le j\le n}\{d[n,j]\}$。

问题2-提示



$d[i, j]$
n
$\vdots$
3
2
$i = 1$
0 0 0 0 0 $\cdots$ $\cdots$ 0
$j = 1$ 2 3 $\cdots\cdots$ n
bottom

问题2-提示

```
Input: An n × n checkerboard, award function p.
Output: Maximum award.
Let d[1..n,1..n] and w[1..n,1..n] be two 2-dimension arrays;
for j ← 1 to n do
|   d[1, j] ← 0;
end
for i ← 2 to n do
|   for j ← 1 to n do
|   |   up ← d[i − 1, j] + p((i − 1, j), (i, j));
|   |   if j is equal to 1 then
|   |   |   upLeft ← −∞;
|   |   |   upRight ← d[i − 1, j + 1] + p((i − 1, j + 1), (i, j));
|   |   end
|   |   else if j is equal to n then
|   |   |   upLeft ← d[i − 1, j − 1] + p((i − 1, j − 1), (i, j));
|   |   |   upRight ← −∞;
|   |   end
|   |   else
|   |   |   upLeft ← d[i − 1, j − 1] + p((i − 1, j − 1), (i, j));
|   |   |   upRight ← d[i − 1, j + 1] + p((i − 1, j + 1), (i, j));
|   |   end
|   |   d[i, j] ←max(upLeft, up, upRight);
|   |   if d[i, j] is equal to upLeft then
|   |   |   w[i, j] ← j − 1;
|   |   end
|   |   else if d[i, j] is equal to up then
|   |   |   w[i, j] ← j;
|   |   end
|   |   else
|   |   |   w[i, j] ← j + 1;
|   |   end
|   end
end
```

问题2-提示

- 其中数组w[i, j]用来记录格(i, j)的最高奖励是从哪一格得到的。
- 下面的算法将输出得到格(i, j)的最高奖励的步骤。

Print-Moves(w, i, j)

```
if i > 1 then
 | Print-Moves(w, i − 1, w[i, j]);
end
Output (i, j);
```

- 计算二维数组d[1..n, 1..n]和w[1..n, 1..n]的时间复杂度为$O(n^2)$. 执行Print-Moves的复杂度为$O(n)$.