习题课2

苏州大学 计算机科学与技术学院
汪笑宇
Email: xywang21@suda.edu.cn

作业3-4

- 设计一个$O(n^2)$ 时间的算法，求一个n个数的序列的最长单调递增子序列。请按照动态规划算法求解的4个步骤进行解答。

作业3-4 (续)

1. 最优解结构特征：
   （回想最大子数组问题的动态规划解法）
   求以每一个元素结尾的单调递增子序列长度：
   该长度=前缀最长单调递增子序列长度+1
   之后取最大值即可

作业3-4 (续)

2. 递归解：

- $A[1..n]$：待求解序列
- $c[i]$：以下标i元素结尾的最长单调递增子序列长度
- $s[i]$：以下标i元素结尾的最长单调递增子序列倒数第二个元素位置

$$c[i] = \begin{cases} \max\limits_{1 \le j \le i-1 \ \wedge \ A[j] \le A[i]} \{c[j]\} + 1, & \exists 1 \le j < i : A[j] \le A[i], \\ 1, & \forall 1 \le j < i : A[j] > A[i]. \end{cases}$$

$$s[i] = \begin{cases} j, & \exists 1 \le j < i : A[j] \le A[i] \\ & \quad \wedge \ c[j] \text{ is the maximum number in } c[1..i-1], \\ 0, & \forall 1 \le j < i : A[j] > A[i]. \end{cases}$$

作业3-4 (续)

3. 计算最优解的值

```
LIS(A)
n ← A.length
let c[1..n] and s[1..n] be new arrays
for i ← 1 to n do
      c[i] ← 1,  s[i] ← 0
for i ← 1 to n do
      m ← 0  //m记录前缀最长单调递增子序列长度
      for j ← 1 to i – 1 do
            if A[j] ≤ A[i] and c[j] > m
                  m ← c[j],  c[i] ← c[j] + 1,  s[i] ← j
ans ← 0 //ans记录最长单调递增子序列长度
t ← 0  //t记录最长单调递增子序列最后一个元素下标
for i ← 1 to n do
      if c[i] > ans
            ans ← c[i],  t ← i
return ans, t, c, and s
```

$O(n^2)$

作业3-4 (续)

4. 构造最优解

```
LIS_OUTPUT(t, c, s, A)
print "最长单调递增子序列长度：" c[t]
LIS_RECURSIVE(t, c, s, A)
```

```
LIS_RECURSIVE(t, c, s, A)
if  s[t] ≠ 0
        LIS_RECURSIVE(s[t], c, s, A)
print "A[" t "] = " A[t]
```

作业3-5

- 找零问题：设数组$A[1..n]$ 中的元素表示n个零钱面值，设计一个动态规划算法寻找可找开某个金额的最少零钱数量，及相对应的找零方案，若不存在找零方案则返回错误信息。例如：$A = \{1, 2, 5\}$，找零金额为11，则最少零钱数量为3，找零方案为5+5+1。请按照动态规划算法求解的4 个步骤进行解答。

作业3-5 (续)

1. 最优解结构特征：
   找开i $(0 \le i \le N)$元钱的最少零钱数量 =
   找开$(i - A[j])$元钱的最少零钱数量 + 1

2. 递归解：$m[i]$：找开i元钱所需的最少零钱数量

$$m[i] = \begin{cases} 0, & i = 0, \\ \infty, & 0 < i < \min\{A[1..n]\}, \\ \min\limits_{1 \le j \le n, i - A[j] \ge 0} \{m[i - A[j]]\} + 1, & i \ge \min\{A[1..n]\}. \end{cases}$$

作业3-5 (续)

3. 计算最优解的值

```
CHANGE(A, N)
let m[0..N] and s[1..N] be new arrays
n ← A.length,  m[0] ← 0,  p ← ∞
for i ← 1 to n do
        if A[i] < p
                p ← A[i]  //p记录最小零钱面值
for i ← 1 to p – 1 do
        m[i] ← ∞
for i ← p to N do
        t ← ∞  //t记录找开(i – A[j])金额的最少零钱数量
        for j ← 1 to n do
                if m[i – A[j]] < t
                        t ← m[i – A[j]],  s[i] ← A[j]
        m[i] ← t + 1
return m and s
```

作业3-5 (续)

4. 构造最优解

```
CHANGE_OUTPUT(N, m, s)
if m[N] = ∞
    print "无法找开"
else i ← N
    while i > 0 do
        print "面值"s[i]
        i ← i – s[i]
    print "最少零钱数："m[N]
```

作业4-3

- 定义装箱问题为：有n个物品，编号为1, 2, ..., n，其中第i ($1 \le i \le n$)号物品的重量为$w_i \in (0, 1]$。需寻找一个使得n个物品全部装箱的装箱方案，且装入的箱子数量最少。注意，这里每个箱子容量都是1。
- 其中FirstFit算法是比较常用的在线装箱算法。在线算法指的是算法执行时不需要知道全局的输入信息。FirstFit算法的基本思想是：对于每个物品，装入第一个可以装进去的箱子。若前面有物品的箱子都无法装入，则新开一个箱子装入。
- (1) 请写出FirstFit算法的伪代码；(2) 请证明该算法得到的解SOL ≤ 2OPT。（提示：最多只有一个箱子是半满的，因此可以找到所有物品重量之和与(SOL-1)/2的关系；且OPT一定不小于所有物品重量之和）

作业4-3 (续)

```
BINPACKING(w)
n ← w.length
k ← 1
for i ← 1 to n do
    将第i号物品放入1..k号箱子中第一个可以放入的箱子
    if 第i号物品无法放入1..k号箱子任一个中
        新开一个箱子放入第i号物品
        k ← k + 1
return k个箱子的装箱情况
```

- 近似比证明：

观察得知：最多只有一个箱子不能达到半满状态

于是有 $\sum_i w_i > (\mathrm{SOL}-1)/2$，并且有 $\mathrm{OPT} \geq \sum_i w_i$

于是 $\mathrm{SOL}-1 < 2\sum_i w_i \leq 2\mathrm{OPT}$，即 $\mathrm{SOL} < 2\mathrm{OPT}+1$

因为SOL和OPT都是正整数，于是 $\mathrm{SOL} \leq 2\mathrm{OPT}$

作业4-3 (续)

- 0-1背包问题变种：每件物品价值相同，仅考虑在有限的背包容量中，装入的总重量最大（子集和问题）
- 近似算法（贪心策略）：将物品按照重量从大到小排序依次装入，直到无法装入为止。该方法近似比为1/2

作业4-3 (续)

```
KNAPSACK1_APPROX(w, W)
将w[1..n]单调递减排序
n ← w.length,  sum ← 0
A ← ∅
for i ← 1 to n do
    if sum + w[i] ≤ W
        A ← A ∪ {w[i]}
        sum ← sum + w[i]
return A and sum
```

■近似比证明：
首先受容量限制一定有
OPT $\leq W$

- 可装入的物品分两种情况：
  (1) 最重物品重量超过$W/2$；
  (2) 最重物品重量不超过$W/2$。
- 第(1)种情况：最重物品重量超过$W/2$时，因为按照重量从重到轻放入背包，则最重物品被装入，此时的解SOL $\geq W/2 \geq$ OPT/2，即SOL/OPT $\geq$ 1/2。
- 第(2)种情况：假设前i个物品都可装入背包，若此时装包重量依旧不超过$W/2$，则i+1号物品依旧可装入，直到装包重量超过$W/2$时，后续物品无法判断是否能装入。此时仍有SOL $\geq W/2 \geq$ OPT/2，即SOL/OPT $\geq$ 1/2。

综上，算法近似比为1/2。