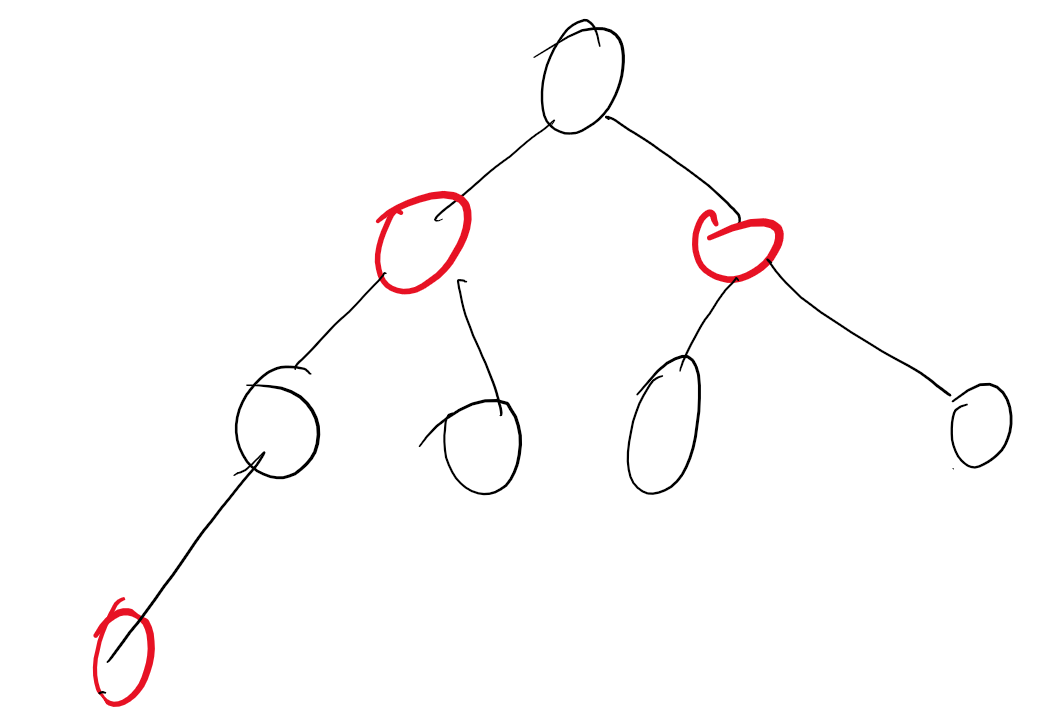
2018 年秋季学期《算法基础》 期中试题

1. 简答题
2. 动态规划的时间性能优势：

保存已解决的子问题的答案，在需要时再查找，可避免重复计算、节省时间。

1. 红黑树：



1. 计算题
2. ？？？
3. ？？？
4. 动态规划矩阵

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | G | T | T | A | C | G |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0↑ | 0↑ | 0↑ | 0↑ | 1↖ | 1← |
| T | 0 | 0↑ | 1↖ | 1↖ | 1← | 1↑ | 1↑ |
| G | 0 | 1↖ | 1↑ | 1↑ | 1↑ | 1↑ | 2↖ |
| A | 0 | 1↑ | 1↑ | 1↑ | 2↖ | 2← | 2↑ |
| C | 0 | 1↑ | 1↑ | 1↑ | 2↑ | 3↖ | 3← |
| T | 0 | 1↑ | 2↖ | 2↖ | 2↑ | 3↑ | 3↑ |
| G | 0 | 1↖ | 2↑ | 2↑ | 2↑ | 3↑ | 4↖ |
| T | 0 | 1↑ | 2↖ | 3↖ | 3← | 3↑ | 4↑ |

LCS：TACG

1. 算法设计与分析
2. （1）？？？

（2）？？？

1. （1）

（2）

1. （1）count(i,j)表示从j个物品中选出i个的方式数

递归式：

伪代码：

count[][] //用于存储组合数

for i = 1 to m do

count[i][i] = 1

for i = 1 to n do

count[1][i] = i

for i = 1 to n - m do

for j = 2 to m do

count[j][i+j] = count[j][i+j-1]+count[j-1][i+j-1]

return count[m][n];

（2）改进思想：由于每轮以对角线的方向填count表，所以用一个长度为m的数组存储每轮更新的count，数组存储的是上次更新的count，下次更新时读取数组就是读取上次更新的count。

伪代码：

count[] //用于存储组合数

for i = 1 to m do

count[i] = 1

for i = 1 to n – m do

count[1] = i

for j = 2 to m do

count[j] += count[j – 1]

return count[m]

1. （1）算法思想：构建一个m\*n的表table，行表示字符串strm的字符，列表示字符串strn的字符。填表时

table中最大值就是最长公共子串的长度。假设table(i,j)为最大值，则从strn[table(i,j)]开始的向前table(i,j)个字符组成的子串就是最长公共子串。

伪代码：

str1 //长度为n的字符串

str2 //长度为m的字符串

maxlen = 0//最大长度

index //最长公共子串的尾序号

table[][]

for i = 0 to m do

table[i][0] = 0

for j = 0 to n do

table[0][j] = 0

for i = 1 to m do

for j = 1 to n do

if str1[j] == str2[i] then

table[i][j] = table[i – 1][j – 1] + 1

if table[i][j] > maxlen then

maxlen = table[i][j]

index = j

else

table[i][j] = 0

return str1.subString(j – maxlen, j)

时间复杂度：O(nm)

（2）改进思路：由于最长公共子串是连续的，所以可以按对角线的方向填表，这样只需维护一个值来存储子串的长度，空间复杂度由（1）中的O(nm)降为O(1)。