北京林业大学

计算机算法设计与实践

大作业

学　　院	信息学院
专业名称	计算机科学与技术创新实验班
班　　级	计创 18
学　　号	181002222
姓　　名	连月菡
辅导老师	王春玲

2020 年 06 月 23 日

北 京 林 业 大 学

2019 学年—2020 学年第 2 学期计算机算法设计与实践实验报告书

专业：计算机科学与技术(创新实验班)　班级：计创 18

姓　　名：连月菡　　学　　号：181002222

实验地点：家　　任课教师：王春玲

实验题目：大作业

实验环境：Visual Studio 2019 Community

第一题

一、 问题描述

距离高考还有 7 天时间！晓明想尽可能有效地分配时间进行 4 门课的复习。目前 每门课至少还要复习一天，而且他喜欢每天只复习一门课，所以可能分配给每门课的 复习时间是 1、2、3 或 4 天。他估计不同的复习时间可能提高的分数如下表所示。

复习天数	可能提高的分数			
	语文	数学	英语	综合
1	4	3	5	2
2	4	5	6	4
3	5	6	8	7
4	8	7	8	8

希望你用动态规划算法帮助晓明安排时间，以使他能提高的总分数达到最大。

二、 问题分析

这个问题属于动态规划中的多阶段决策问题,每做一次决策就可以得到解的一部分,当所有决策做完之后,完整的解就浮出水面了。这里要求做出 4 个相应关联的决策,即每门科目应该分配多少天的复习时间,由于这里没有固定的顺序,所以四门课程可以看成动态规划模型中的四个阶段。将语文,数学,英语,综合四门课程分别编号为 1,2,3,4。

阶段 s=1,2,3,4 代表课程安排已经决定了几门; 决策变量 x_s ，即 pattern 是分配给课程 course 的复习天数, 且$x_a + x_b + x_c + x_d = 7$; $P_i(x_s)$代表分配给课程 i 的 x 天所提高的分数, 题目需要求的是选择不同的x_s, 求出

$$maxn = max(P_1(x_s) + P_2(x_s) + P_3(x_s) + P_4(x_s))$$

课程的复习天数和提高分数的关系

```
int score[5][5] = {//课程的复习天数和提高分数的关系
    0,0,0,0,0,
    0,4,3,5,2,//如, 这一行代表复习1天, 课程1~4提高的分数
    0,4,5,6,4,//score[i][j]代表复习i天, 课程j提高的分
数  0,5,6,8,7,
    0,8,7,8,8,
};
```

由于题目要求, 这四门课程的复习天数取值为 1, 2, 3, 4, 因此符合要求的复习天数分配情况如下:[4, 1, 1, 1], [3, 2, 1, 1], [2, 2, 2, 1]。

符合要求的课程安排的走法

```
int pattern[8][5] = {//符合要求的课程安排的走法
    0,0,0,0,0,
    0,4,1,1,1,//如,pattern[i][j]代表此次遇到的打乱顺序的四门课
程  0,3,2,1,1,//按照4,1,1,1天分别分配
    0,1,1,3,2,
    0,1,1,2,3,
    0,1,3,1,2,
    0,1,3,2,1,
    0,1,2,2,2,
};
```

由表格可推得安排的结果：

剩余天数	7	7	7	7
阶段 s=1	1	1	1	1
决策变量 x_1	4	3	2	1
累积分数	8	5	4	4
剩余天数	3	4	5	6
阶段 s=2	2	2	2	2
决策变量 x_2	1	2;1	1;2;3	1;2;3;4
累积分数	11	10;8	7;9;10	7;9;10;11
剩余天数	2	2;3	4;3;2	5;4;3;2
阶段 s=3	3	3	3	3
决策变量 x_3	1	1;1, 2	3, 2;2, 1;1	4, 3, 2, 1;3, 2, 1;2, 1;1
累积分数	16	15;14, 14	15, 13;15, 14;15	15, 15, 13, 12;17, 15, 14;16, 15;16
剩余天数	1	1	1, 2;1, 2;1	1, 2, 3, 4;1, 2, 3;1, 2;1
阶段 s=4	4	4	4	4
决策变量 x_4	1	1;2, 1	1, 2;1, 2;1	1, 2, 3, 4;1, 2, 3;1, 2;1
累积分数	18	17;18;16	17, 17;17, 18;17	17, 19, 20, 20;19, 19, 21;18, 19;18

从表中最后一行，可以看出最大提高分数是 21 分。从 21 回溯到前面的表格，可知其具体的复习安排。

三、 算法描述

算法描述

```
对于主函数:
    1.初始化最大值,初始化最大情况下的课程安排
    2.遍历7个课程安排方式(3种,7个),每个课程安排方式遍历4个课程序号
      作为第一阶段所安排的科目。确定好第一阶段的科目以后,初始化本次
      安排的提高分数,初始化课程安排标记vis,初始化各个课程安排天数
      a.记录该课程被安排,记录该课程被安排的天数
      b.给dp()函数传入值
      c.当前值被赋值为dp()的返回值
      d.如果当前值大于最大值,更新最大值,更新最大情况下的复习时间安排

对于dp():
    1.传入日程安排的方式,课程序号,日程安排的阶段数,目前累积提高的分数
    2.如果当前课程已经被安排过了,课程序号加一,直到此课程未被安排为止,
      如果课程1~4已经安排完毕了,则返回目前累积的提高分数
    3.对于这门未被安排的课程
      a.标记这门课被安排复习了
      b.记录这次课程复习的天数
      c.向dp()传入 日程安排的方式,1,日程安排的阶段数+1,
        目前累积提高的分数+安排好此课程所能提高的分数
      d.目前累积的提高分数被赋值为dp()的返回值
    4.返回目前累积的提高分数
}
```

伪代码

```
status dp(日程安排的方式,课程序号,日程安排的阶段数,目前累积提高的分数)
{//由于course每次传值都为1，所以是从课程1开始遍历的
    while 如果当前课程已经被安排过了
        课程序号++//数字加一，遍历到下一门课
        if 课程序号 == 5//说明真实存在的课程1-4已经安排完毕了
            return 目前累积的提高分数
    }
    vis[课程序号]= 1;//标记这门课被安排复习了
    //记录这次课程course安排复习的天数
    sche[课程序号] = pattern[日程安排的方式][日程安排的阶段数];
    目前累积的提高分数 = dp(日程安排的方式,1,日程安排的阶段数+1,
                目前累积提高的分数+安排好此课程所能提高的分数);
    return 目前累积的提高分数
}

int main()
{
    最大值 = 0;//初始化最大值
    最大情况下的课程安排[5] = { 0 };//初始化最大情况下的课程安排
    for p:1->7,p++
        for i:1->4,i++
            ans = 0//初始化本次安排的提高分数
            初始化课程安排标记vis
            初始化各个课程安排天数sche
            vis[i]= 1//记录该课程被安排
            sche[i] = pattern[p][1];//记录该课程被安排的天数
            ans = dp(p,1,2,ans+安排好此课程所能提高的分数);
            if 当前值大于最大值
                最大值 = ans;//更新最大值
                for j:1->4,j++
                    //更新最大情况下的复习时间安排
                    最大情况下的课程安排[j]=sche[j]
                endfor
            endif
        endfor
    endfor
}
```

四、 算法分析

在 main()中，由 3 个层层嵌套的 for 循环,故时间复杂度为 O(n*n*n);
在 dp()中, while 的时间复杂度为 O(1), dp()递归的时间复杂度为 O(n);
因此总的时间复杂度是 O(n^3)+O(n)+O(1)，即 T(n)=O(n^3)。

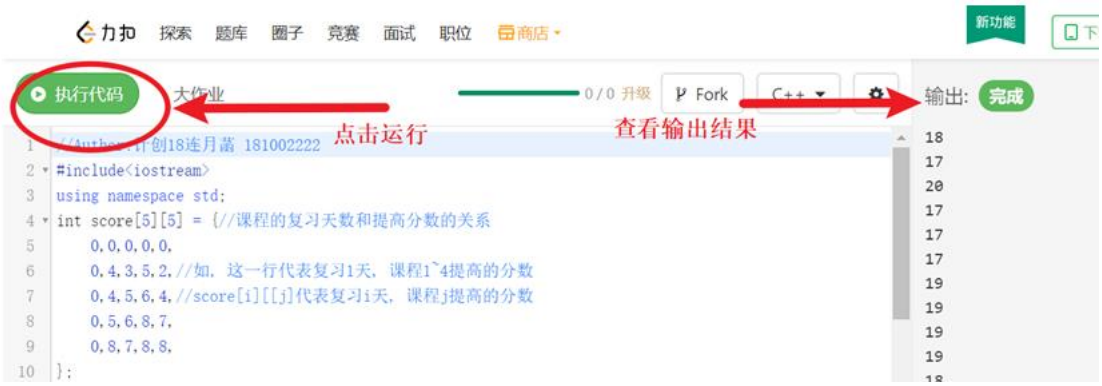
五、 算法实现

算法实现部分

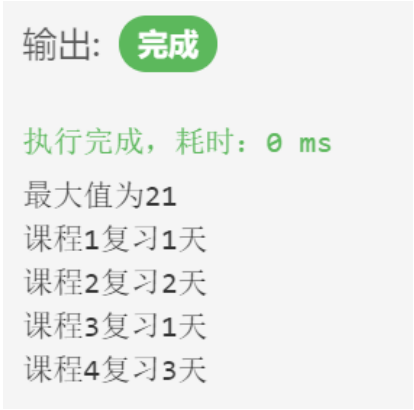
```
int day;//代表日程安排是pattern[day]中的安排方式
bool vis[5];//vis[i]代表课程i是否已经安排完
int sche[5] = { 0 };//sche表示schedule, sche[i]代表课程i复习天数

//day表示课程安排的走法(1-6), course表示课程1,2,3,4
//pos表示课程安排走法到了第几门课了, ans表示目前累积的分数
int dp( int day,int course,int pos, int ans)
{//由于course每次传值都为1, 所以是从课程1开始遍历的
    while (vis[course] == 1) {//如果当前课程已经被安排过了
        course++;//数字加一, 轮到下一门课
        if (course == 5)//如果course==5, 说明真实存在的课程1-4已经安排完毕了
            return ans;//返回目前累积的提高分数
    }
    vis[course]= 1;//标记这门课被安排复习了
    sche[course] = pattern[day][pos];//记录这次课程course安排复习的天数 pattern[day]
[pos]ns = dp(day, 1,pos+1,ans+score[pattern[day][pos]][course]);
    return ans;//返回目前累积的提高分数
}
int main()
{
    int ans,maxn;
    maxn = 0;//初始化最大值
    int maxsche[5] = { 0 };//初始化最大情况下的课程安排
    for (int p = 1; p < 6; p++){
        for (int i = 1; i < 5; ++i){
            ans = 0;//初始化本次安排的提高分数
            memset(vis, 0, sizeof(vis));//初始化课程是否已安排标记
            memset(sche, 0, sizeof(sche));//初始化各个课程安排天数
            vis[i]= 1;//记录该课程被安排
            sche[i] = pattern[p][1];//记录该课程被安排的天数
            ans = dp(p,1,2, ans+score[pattern[p][1]][i]);
            printf("%d\n", ans);
            if (maxn < ans){//如果当前值大于目前的最大值
                maxn = ans;//更新最大值
                for (int j = 0; j < 5; j++)
                    maxsche[j] = sche[j];//更新最大情况下的复习时间安排
            }
        }
    }
    printf("最大值为%d\n", maxn);
    for (int j = 1; j < 5; j++)
        printf("课程%d复习%d天\n", j, maxsche[j]);
    return 0;
}
```

[在线运行此代码](https://leetcode-cn.com/playground/qBPZSGZn/)
(https://leetcode-cn.com/playground/qBPZSGZn/)



运行结果:



因此，晓明要想让复习可能提高的总分数达到最高，应该安排复习语文 1 天,数学 2 天，英语 1 天，综合 3 天。提高分数的最大值为 21 分。

第二题

一、 问题描述

本门课程以计算机算法中的查找、排序、组合、图、几何等问题类型为基础，讨论了蛮力法、分治法、减治法、动态规划法、贪心法、回溯法、分支限界法、近似算法、概率算法等算法思想。结合自己的研究实际，谈谈你对算法设计与分析在项目实践中作用的认识。

二、 个人认识

Ⅰ.概述

本文从”有穷性，正确性，输入，输出，可行性”这五个算法的特征展开，结合自身经历和研究实际，论述算法设计与分析在项目实践中的作用的认识。

Ⅱ.在项目实践中的作用

1. **精确的问题陈述**。

在任何项目启动准备工作中，首先都需要对项目中要解决的问题进行明确。比如要建立起个性化推荐系统，前期收集数据阶段所提出的问题就是”如何收集页面中的用户公开数据?”。对于这个问题，就需要考虑:

a.收集哪些用户数据?

b.使用什么方式解析 xml 内容?

c.遇到什么情况才停止继续收集?

d.读取到用户数据后如何存储?

我们不带偏见地分析完这个收集数据的程序的需求并给出如下的分析结果:

输入: 含有所需数据的页面 xml 文档。这个 xml 文档符合文本的语法规则按照相应的正则表达式，可以提取出需要的数据。

输出: 按照”序号”, ”日期”, ”名称”, ”评分”, ”地址”, ”图片”, ”评价”的增序，每条记录作为一行写在 csv 文件中。

约束: 尽可能快速地收集到页面所需的数据。当记录到 10^7 条时停止。

但是这样其实还不够精确，比如是依次读取同一类型的单个数据，还是一次性读完文件，再分别处理各个类型的单个数据。是一边获取数据一边写入，还是收集到所有数据再写入。我们需要根据数据的规模和类别，以及所使用的语言特性和配套生态进行因地制宜地采取不同的措施。

有了对问题的精确陈述，才有下一步对算法的构思。同时，最先明确好问题，对于一个大中型项目而言更加重要，因为有了前期提出需求者和我们程序员的沟通，可以避免其中可能存在的理解偏差导致的过失，也让项目进度的推进更加有条不紊，防止因为没有解决到问题，而不得不使项目推翻重来。

这一点也是保证了算法的正确性:

（1）对于合法的输入数据能够产生满足要求的输出结果。

（2）对于非法的输入数据能够得出满足规格说明的结果。

（3）对于精心选择的，甚至刁难的测试数据都有满足要求的输出结果。

2. **算法和数据结构。**

马丁·加德纳在《啊哈!灵机一动》中提到:”看起来很困难的问题解决起来可能很简单，并且还可能出人意料之外。” 和一些高级方法不同，如果我们缘

以在编程前，编程中，编程后认真思考，那么就有可能锻炼出这种对于应用算法出神入化的能力。

同样，还是个性化推荐系统，如果要搜寻一个特定的物品，那么这就是一个查找问题。如果是顺序查找，那么平均比较次数就是 n/2 次，如果是二分查找则不会超过 $\log_2 n$ 次，但是二分查找的前提是一组排序好的数据。因此这里还需要用到排序算法。

而如何设计或应用查找和排序算法，与所要查询的数据所存储的方式有关。譬如在页式虚拟存储管理中，会使用到先进先出调度算法，最近最少调度算法，最近最不常用调度算法。如果数据在库中存储方式也是按照最近最不常用调度算法的思想，那么在个性化推荐系统中查询数据时，就可以按照查找频次为关键字进行排序，那么顺序查找算法也不会比二分查找落后了。

有时候我们会直接凭着第一感觉编写出一个洋洋洒洒几百行的程序，但实际上，可能利用一半长度的代码就有可能实现相同的结果，这样在维护程序的时候，会大大简化新旧程序员更替带来的不便，利用更短的时间来研读曾经写下的代码。比如每种数据使用的是不同的数据类型或结构，但是可以通过编写模板类，使得各个类型的数据都能通过一个函数反复处理。

这一点使得算法更具有可读性。

3. **算法优化。**

   有些相邻的数据两两相关，若原本采用线性结构存储，则需要用其他算法进行两两计算，一旦采用树形结构，则允许由一个函数进行识别和处理，这样就减少了程序的运行时间。

4. **数据结构重组。**

   每个时间步都重新组织数据结构，通过花费少量的时间，来节省更多的总的运行时间。

5. **代码优化。**

   有些情况下，并不需要数据并不需要达到 64 位双精度浮点数的标准，可以修改为 32 位单精度，这样替换也能将运行时间减半。对自己的程序的每个函数进行运行时间统计，重新修改耗时最长的那个函数，或者使用汇编语言实现同样的函数。使用 inline 关键字，减少函数调用参数，从而减少函数的调用开销。或根据发生频率，安排 switch 语句的 case 顺序。将递归代码非递归化，这样避免了大量的过程调用和堆栈保存。

6. **性能估计。**

   在限制内存大小的基础上，需要简单估计自己编写代码所占的内存大小是否小于最高限制，如果使用 malloc 函数动态分配空间，还会多占用一些空间。在完成代码的编写以后，还可以使用大 O 分析法了解整个程序的运行时间成本，但实际上只能了解数量级，而不能知道具体的执行次数，但通常来说，我们使用渐进运行时间作为程序效率的重要的衡量标准就够了。这样也便于直观地对算法进行优化，比如把 O(n^3)的复杂度优化成 O(logn)的复杂度。

7. **压缩空间。**

   比如在一张固定大小的数据表中，数据零散分布在各个地方，那么这个数据表就十分稀疏，剩余的空间也没有得到利用，如果存储空间十分紧张的情况下，我们会尽可能地使用哈希链表进行存储。起初数据量较小的时候，可以使用固定大小的数据表，但是随着数据规模的增长，空间资源越来越紧张，就不得不考虑压缩空间，既可以使用键值对数组进行索引，释放未使用的空

间，也可以压缩数据表的行列号位数。同样借用到操作系统中最近最不常用调度算法的概念，可以在资源不足的情况下，优先释放出最不常用的数据，把空间留给新放入的数据。

Ⅲ.总结

在编写项目代码的过程中，主要考虑的通常是运行时间和空间的优化，而算法设计与分析所扮演的角色，就是设计中考虑如何提高程序的运行效率与节省空间，分析中判断设计时疏忽的算法优化空间，进而让一个项目在有限的时间和空间中实现最大的效益。