

华中科技大学

2022

算法设计与分析实践报告

专 业： 计算机科学与技术

班 级： 大数据

学 号： U20211xxxx

姓 名： Xxx

完成期： 2022/12/25



华中科技大学课程设计报告

目 录

1	完成情况	1
2	POJ3233 Matrix Power Series 解题报告	2
2.1	题目分析	2
2.1.1	题目大意	2
2.1.2	算法标签	2
2.1.3	输入输出	2
2.2	算法设计	2
2.2.1	构造矩阵乘法, 矩阵求和函数	2
2.2.2	构造矩阵快速幂函数	2
2.2.3	分治	2
2.3	性能分析	2
2.3.1	时间复杂度:	2
2.3.2	空间复杂度	3
2.4	运行测试	3
3	POJ2228 Naptime 解题报告	4
3.1	题目分析	4
3.1.1	题目大意	4
3.1.2	算法标签	4
3.1.3	输入输出	4
3.2	算法设计	4

华中科技大学课程设计报告

3.2.1	状态转移方程的设计	4
3.2.2	环状处理	5
3.2.3	滚动数组实现空间复杂度优化	5
3.3	性能分析	6
3.3.1	时间复杂度	6
3.3.2	空间复杂度	6
3.4	运行测试	6
4	POJ1062 昂贵的聘礼	7
4.1	题目分析	7
4.1.1	题目大意	7
4.1.2	算法标签	7
4.1.3	输入输出	7
4.2	算法设计	7
4.2.1	问题转换为图问题	7
4.3	性能分析	8
4.3.1	时间复杂度	8
4.3.2	空间复杂度	8
4.4	运行测试	8
5	POJ1184 聪明的打字员解题报告	10
5.1	题目分析	10
5.1.1	题目大意	10
5.1.2	算法标签	10

华中科技大学课程设计报告

5.1.3	输入输出	10
5.2	算法设计	10
5.3	性能分析	11
5.3.1	时间复杂度	11
5.3.2	空间复杂度	11
5.4	运行测试	11
6	总结	12
6.1	实验总结	12
6.2	心得体会与建议	13
6.2.1	心得体会	13
6.2.2	建议	13

1 完成情况

在 6 个实验的算法实践中，按照实验报告要求**已完成 20+4 的任务**，并在剩余题目中选择完成部分题目以及选做题。下图给出 POJ 上显示的完成情况。

Rank:	27680	Solved Problems List
Solved:	34	1000 1005 1042 1050 1062 1077 1084 1088 1184
Submissions:	83	1185 1201 1324 1328 1475 1636 1700 1723 1753
School:	HUST	1860 2228 2366 2387 2449 2503 2506 2586 3040
Email:	PyGoneBe@outlook.com	3169 3233 3269 3295 3579 3660 3714

图表 1-1 题目完成情况

2 P0J3233 Matrix Power Series 解题报告

2.1 题目分析

2.1.1 题目大意

给定 n 、 k 和一个 $n \times n$ 的矩阵, 计算 $S = \sum_{i=1}^k A^i$

2.1.2 算法标签

矩阵快速幂, 二分

2.1.3 输入输出

Input: 第一行输入 n, k, m , 下面 n 行输入一个 $n \times n$ 的矩阵。

Output: $n \times n$ 的结果矩阵模 m 的余数矩阵。

2.2 算法设计

2.2.1 构造矩阵乘法, 矩阵求和函数

2.2.2 构造矩阵快速幂函数

2.2.3 分治

k 为偶数, $S = \sum_{i=1}^k A^i = A + \dots + A^k = (1 + A^{k/2})(A + \dots + A^{k/2})$

k 为奇数, $S = A^k + (1 + A^{k-1/2})(A + \dots + A^{k-1/2})$

2.3 性能分析

2.3.1 时间复杂度:

矩阵快速幂的时间复杂度为 $O(n^3 \times \log k)$

而分治的时间复杂度为 $O(\log k)$

故总的时间复杂度为 $O(n^3 \times (\log k)^2)$

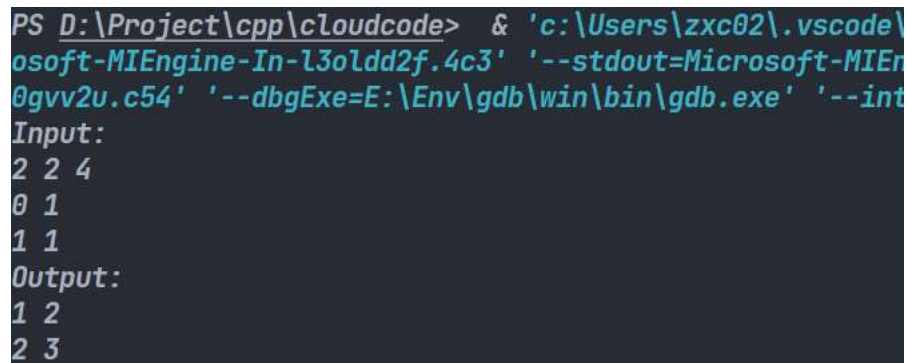
2.3.2 空间复杂度

空间复杂度即为 矩阵的大小 即 $O(n^2)$

2.4 运行测试

Input: 2 2 4 0 1 1 1

Output: 1 2 2 3



```
PS D:\Project\cpp\cloudcode> & 'c:\Users\zxc02\.vscode\Microsoft-MIEngine-In-13o1dd2f.4c3' '--stdout=Microsoft-MIEngine\0gvv2v.c54' '--dbgExe=E:\Env\gdb\win\bin\gdb.exe' '--int
Input:
2 2 4
0 1
1 1
Output:
1 2
2 3
```

图表 2-1 运行测试图

3 P0J2228 Naptime 解题报告

3.1 题目分析

3.1.1 题目大意

一天由 n 个时间段构成, 在第 i 个时间段睡觉能够恢复 U_i 点体力。有一头牛要休息 b 个小时, 可以不连续, 但休息的第 1 个个时间段无法恢复体力。前一天的最后一个小时和第二天的第一个小时是连在一起的, 求这头牛能恢复的体力最大值。

3.1.2 算法标签

动态规划, 环状 DP

3.1.3 输入输出

Input: 第一行: 两个整型数据 N, B

下面 N 行每行一个整数 $U_i (U_i \in [0, 200,000])$

Output: 可以恢复体力的最大值

3.2 算法设计

3.2.1 状态转移方程的设计

DP 数组定义: $dp[i][j][k]$

存储 U_i 的数组: W_i

其中 i, j 代表前 i 个小时中有 j 个小时用于睡觉。

$k=0$ 代表选择第 i 个小时用于休息, $k=1$ 代表不选择第 i 个小时用于休息。

当 $k=0$ 即不选择第 i 个小时用于休息时, 答案应该是不选择第 $i-1$ 个小时用于休息和选择第 $i-1$ 个小时用于休息中的最大值。得:

$$dp[i][j][0] = \max(dp[i-1][j][0], dp[i-1][j][1])$$

当 $k=1$, 即选择第 i 个小时用于休息时, 答案则为选择第 $i-1$ 个小时用于休息和选择第 $i-1$ 个小时用于休息中的最大值。得:

$$dp[i][j][1] = \max(dp[i-1][j-1][0], u[i] + dp[i-1][j-1][1])$$

那么在算法执行过程中, 状态转移方程即为:

$$\begin{cases} dp[i][j][0] = \max(dp[i-1][j][0], dp[i-1][j][1]) \\ dp[i][j][1] = \max(dp[i-1][j-1][0], u[i] + dp[i-1][j-1][1]) \end{cases}$$

3.2.2 环状处理

环状特征: 第一天的第 N 小时和第二天的第 1 小时都在休息。

解决方案: $dp[1][1][1]=u[1]$, 然后继续运行一遍 DP 即可。

3.2.3 滚动数组实现空间复杂度优化

若将数组设置为 $dp[3835][3835][2]$, 会超出题目所给空间限制, 针对状态转移方程, 发现更新过程只需利用 $i-1$ 维得内容

故我们对 DP 数组做一个滚动数组优化, 只保存上一维内容, 即可将数组定义为 $dp[2][3835][2]$, 从而满足题目所给空间限制

与此同时, 结果公式变为: $\max(dp[N+1][B][0], dp[N+1][B][1])$

3.3 性能分析

3.3.1 时间复杂度

动态规划程序会遍历 $(N, B, 2)$ 三维状态空间中的所有格点，故时间复杂度为 $O(NB)$

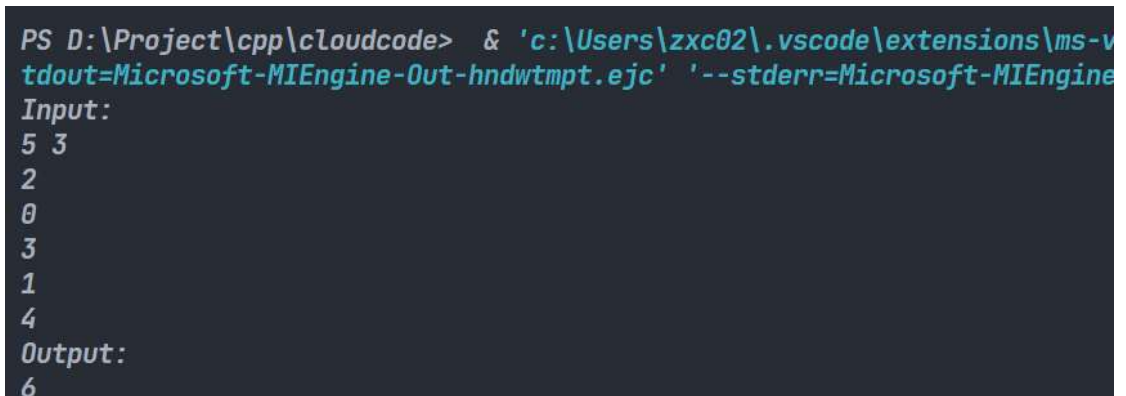
3.3.2 空间复杂度

构建的 DP 数组开销为 $4*N$ ，故空间复杂度为 $O(N)$

3.4 运行测试

Input: 5 3 2 0 3 1 4

Output: 6



```
PS D:\Project\cpp\cloudcode> & 'c:\Users\zxc02\.vscode\extensions\ms-vscode-vscode\bin\code.cmd' --stdout=Microsoft-MIEngine-Out-hndwtmpt.ejc' --stderr=Microsoft-MIEngine-Out-hndwtmpt.ejc'
Input:
5
3
2
0
3
1
4
Output:
6
```

图表 3-1 运行测试图

4 POJ1062 昂贵的聘礼

4.1 题目分析

4.1.1 题目大意

根据题目所述, 有 n 个物品, 对于每个物品, 都有一个等级, 一个直接购买 的价格, 同时, 也有可能存在若干替代品及该替代品所对应的优惠。同时, 给定 一个等级差距 M , 任何合法物品交易链都需保证, 不存在两个物品之间的等级差距超过 M 。

问题为, 求解最少花多少价格才能获得第一个物品

4.1.2 算法标签

单源最短路径

4.1.3 输入输出

Input: 第一行是两个整数 M, N ($1 \leq N \leq 100$), 依次表示地位等级差距限制和物品的总数。接下来按照编号从小到大依次给出了 N 个物品的描述。每个物品的描述开头是三个非负整数 P, L, X ($X \leq N$), 依次表示该物品的价格、主人的地位等级和替代品总数。接下来 X 行每行包括两个整数 T 和 V , 分别表示替代品的编号和"优惠价格"。

Output: 输出最少需要的金币数。

4.2 算法设计

4.2.1 问题转换为图问题

将每个物品视为图上的一个点, 且每个点都有一个等级标签 $level$ 。

华中科技大学课程设计报告

对于替代品与对应优惠做如下处理：若物品 $A = \text{物品 } B + \text{优惠价格 } price$,

则建立一条由结点 B 指向结点 A 的有向边, 权重 为 $price$

然后我们做一个等级约束条件, 做一个长度为 M 包含酋长等级的点的线段, 线段之外的点列为不可访问点, 线段向右移动直至酋长等级点作为左端点.

然后对可访问的点中, 采用 *dijkstra* 算法, 求取源点到各点最短路径(即最佳优惠价格)

在移动过程中, 每次都求取所有点价格与优惠价格之和最小的点, 在移动结束后获取最小的点, 即为题目所求结果

4.3 性能分析

4.3.1 时间复杂度

顺序遍历的 *dijkstra* 算法的时间复杂度为 $O(N^2)$

共需进行 m 次 *dijkstra* 算法, 故总的时间复杂度为:

$$O(mN^2)$$

4.3.2 空间复杂度

使用邻接矩阵存储边, 故空间复杂度为: $O(N^2)$

4.4 运行测试

**Input: 1 4 10000 3 2 2 8000 3 5000 1000 2 1 4
200 3000 2 1 4 200 50 2 0**

Output: 5250

```
PS D:\Project\cpp\cloudcode> & 'c:\Users\zxc02\.vscode\Microsoft-MIEngine-In-ecsksfmx.r34' '--stdout=Microsoft-MI54yiyx.0tq' '--dbgExe=E:\Env\gdb\win\bin\gdb.exe' '--i
Input:
1 4
10000 3 2
2 8000
3 5000
1000 2 1
4 200
3000 2 1
4 200
50 2 0
Output:
5250
```

图表 4-1 运行测试图

5 POJ1184 聪明的打字员解题报告

5.1 题目分析

5.1.1 题目大意

给定两个字符串，一个为起始状态，一个为终止状态。同时给出 6 个字符串变换方法。求解最少通过多少次变换可以将字符串由起始状态变化到终止状态。

5.1.2 算法标签

BFS, 剪枝, 哈希

5.1.3 输入输出

Input: 两个字符串，分别是起始字符串和终止字符串

Output: 最少变换次

5.2 算法设计

在状态的最后一位补一个记录光标位置，0 表示第一位，1 表示第二位.....

由 **BFS** 搜索特性，第一个变换后得到结果的即最少次数，故采用 **BFS** 搜索

由哈希来判断队伍有无修改后的元素

BFS 搜索剪枝优化：

- 只有对数字发生了修改，我们去判断队列中有没有该元素
- 只有在当前位和目标位不同时才进行 **Up**, **Down** 操作
- 只有当前位和目标位相同时才进行 **Left**, **Right**, 因为移动不会影响数字的状态

5.3 性能分析

5.3.1 时间复杂度

假设答案在搜索树的第 n 层, 那么 *BFS* 会至多进行 n 次搜索。搜索树的第 i

层有 6^{i-1} 个状态, 所以总的时间复杂度为

$$\sum_{i=1}^n 6^{i-1} = O(6^n)$$

5.3.2 空间复杂度

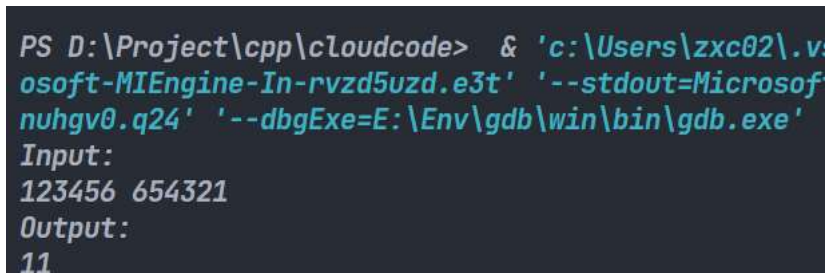
利用哈希表 *map* 存图, 空间复杂度为:

$$O(8 * 6^n) = O(6^n)$$

5.4 运行测试

Input: 123456 654321

Output: 11



```
PS D:\Project\cpp\cloudcode> & 'c:\Users\zxc02\.vscode\bin\code' --stdout=Microsoft.VisualStudio.Code --dbgExe=E:\Env\gdb\win\bin\gdb.exe
Input:
123456 654321
Output:
11
```

图表 5-1 运行测试图

6 总结

6.1 实验总结

经过本次实验，我在算法方面收获很多。收获主要有以下几点。

- 当问题可以分成若干个子问题时，可以使用分治法进行求解。
- 当求解问题中存在最优子结构并且要求出问题最优解时，可以使用动态规划求解，动态规划问题中对于部分问题也可以使用到贪心算法
- 对于部分问题，可以利用图论的知识对问题进行建模，再利用相关算法进行求解，如上文提到的昂贵的聘礼题目
- 最短路算法中，*Bellman-Ford* 和 *SPFA* 算法基于动态规划，可以处理含有负权边的图。*Dijkstra* 算法基于贪心法，只能处理非负权图。
- *BFS*，*DFS* 中，对算法进行合理的剪枝，可以极大提高算法的执行效率吧
- 了解到 *A** 算法的估价函数这一重要概念，可以使用估价函数极大地减少需要搜索的状态数。

6.2 心得体会与建议

6.2.1 心得体会

本次算法实践中，很多题目都是经典的模板题，但由于没有例题指导，或一些启发性的提示，需要比较长的时间搭建一个解题的蓝图，包含需要用到什么算法，需要搭建怎样的数据结构，有没有什么手段减少时间和空间的开销。这是一个成长的过程，在摸索中去了解一些算法的使用场景，又巩固了数据结构的知识，可以说是收获满满。唯一比较失望的就是这个 **OJ** 并不支持 **C++** 新发布的一些语法糖。

6.2.2 建议

在题单中添加多一些和所学算法有关联，但又与所学算法有差异的算法选做题目，来拓宽算法实践的广度，或者需要针对某一种所学算法进行优化类的选做题目来加深对该算法的理解。