

摘 要

本课程设计初步介绍了火车进站与出站问题的解决方法，并通过编程验证了结果.

关键词：栈,catalan 数列

Abstract

This subject preliminary introduced the problem of train's arrival and departure the station, and verified the results by programming.

Key Words: stack, catalan sequence

目 录

第一章 课题背景	1
1.1 问题背景	1
1.2 模型转化	1
第二章 模型的求解	2
第三章 程序的实现与结果测试	5
3.1 主要功能函数	5
3.1.1 全排列的枚举	5
3.1.2 序列的判断	6
3.2 测试结果及分析	6
第四章 总结和问题	8
参考文献	9
致谢	10
附录：程序代码	11

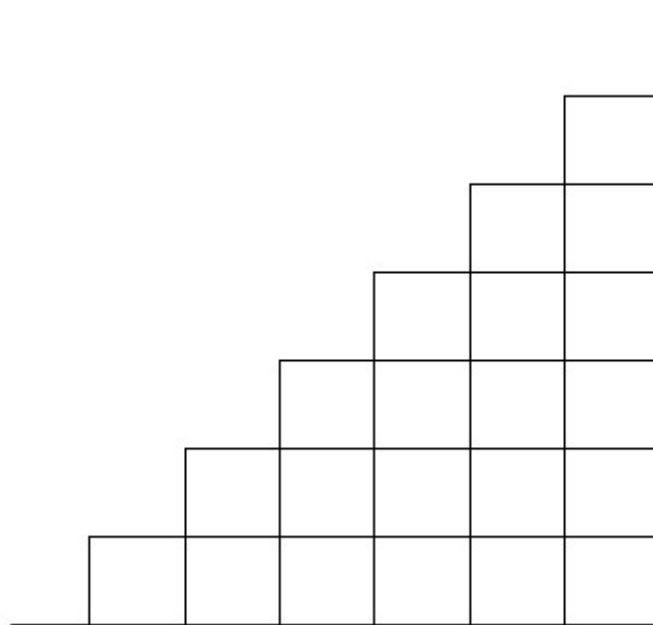
第一章 课题背景

1.1 问题背景

编号为 $1, 2, 3, \dots, n$ 的火车厢被拖入堆栈, 并可以在任何时候将他拖出, 当有 n 节车厢时共有几种排序方式?

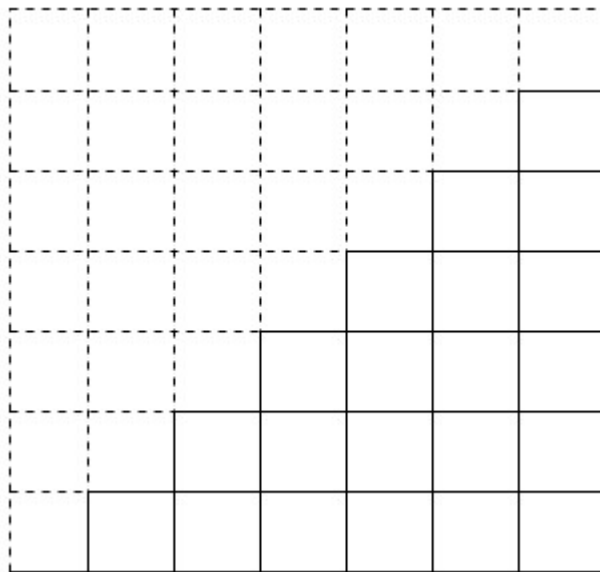
1.2 模型转化

n 个车厢一共进行 n 次进站和 n 次出站, 且任意时刻出站数一定不能大于进站数. 我们把这个问題转化为如下组合问题. 一个点从左下角移动到右上角, 向右走代表进站, 向上走代表出站, 求所有的走法总数.

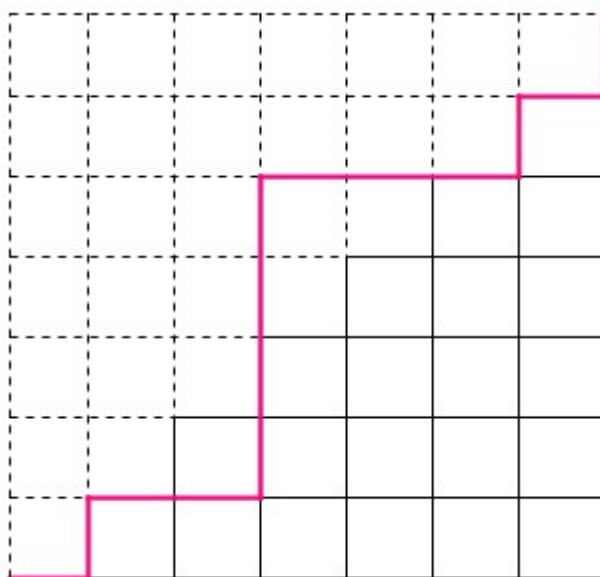


第二章 模型的求解

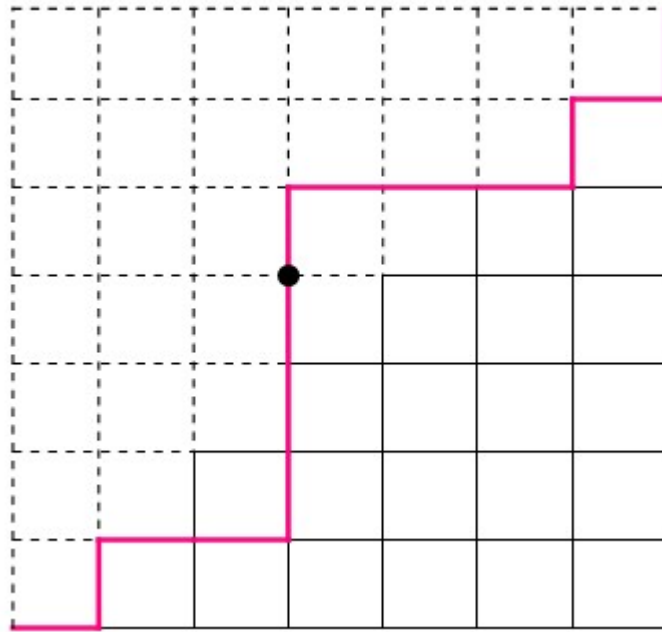
我们将图补充完整



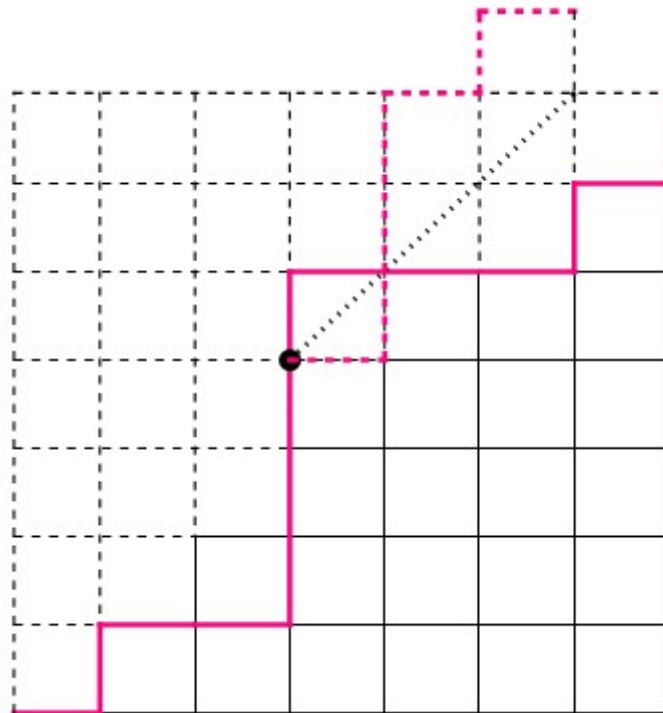
考虑一个错误的走法, 即路线越出了实线走到了虚线部分



我们找到超出实线边界的第一个点



并以该点作出右上角部分的对称图形



这样我们就得到了一个由 $(0,0)$ 到 $(n-1, n+1)$ 的越界的路径, 他由 $n-1$ 次向右和 $n+1$ 次向上运动组成.

考虑所有从 $(0,0)$ 到 (n,n) 的移动, 共有 $\binom{2n}{n}$ 种走法, 而所有越界的走法都可以对应为一个由 $n-1$ 次向右和 $n+1$ 次向上运动组成的走法, 所以错误的走法数

为 $\binom{2n}{n-1}$, 故总的正确走法数为

$$S_n = \binom{2n}{n} - \binom{2n}{n-1} = \frac{1}{n+1} \binom{2n}{n}$$

第三章 程序的实现与结果测试

我们首先通过一个搜索算法来枚举出 $1\ n$ 的全排列, 将结果保存在数组 `target` 中, 然后再调用 `test` 函数判断此序列是否是合法的出站序列.

3.1 主要功能函数

3.1.1 全排列的枚举

```
void dfs(int cur, int depth)//深度优先搜索
{
    if(cur == depth)
    {
#ifdef TEST
        for(int i = 1; i <= depth; i++)
            cout<<target[i];

        if(test(target, depth))
            cout<<" "<<"yes";
        else
            cout<<" "<<"no";
        cout<<endl;
#endif
        if(test(target, depth)) num++;
    }
    else
    {
        for(int i = 1; i <= depth; i++)
        {
            if(vis[i]) continue;
            vis[i] = 1;
```



```

        target[cur+1] = i;
        dfs(cur+1, depth);
        vis[i] = 0;
    }
}
}

```

3.1.2 序列的判断

```

bool test(int *target, int depth)
{
    stack<int> s;

    int a = 1, b = 1;
    while(b <= depth)
    {
        if(a == target[b]) {a++; b++;}

        else if(!s.empty() && s.top() == target[b]) {s.pop(); b++;}

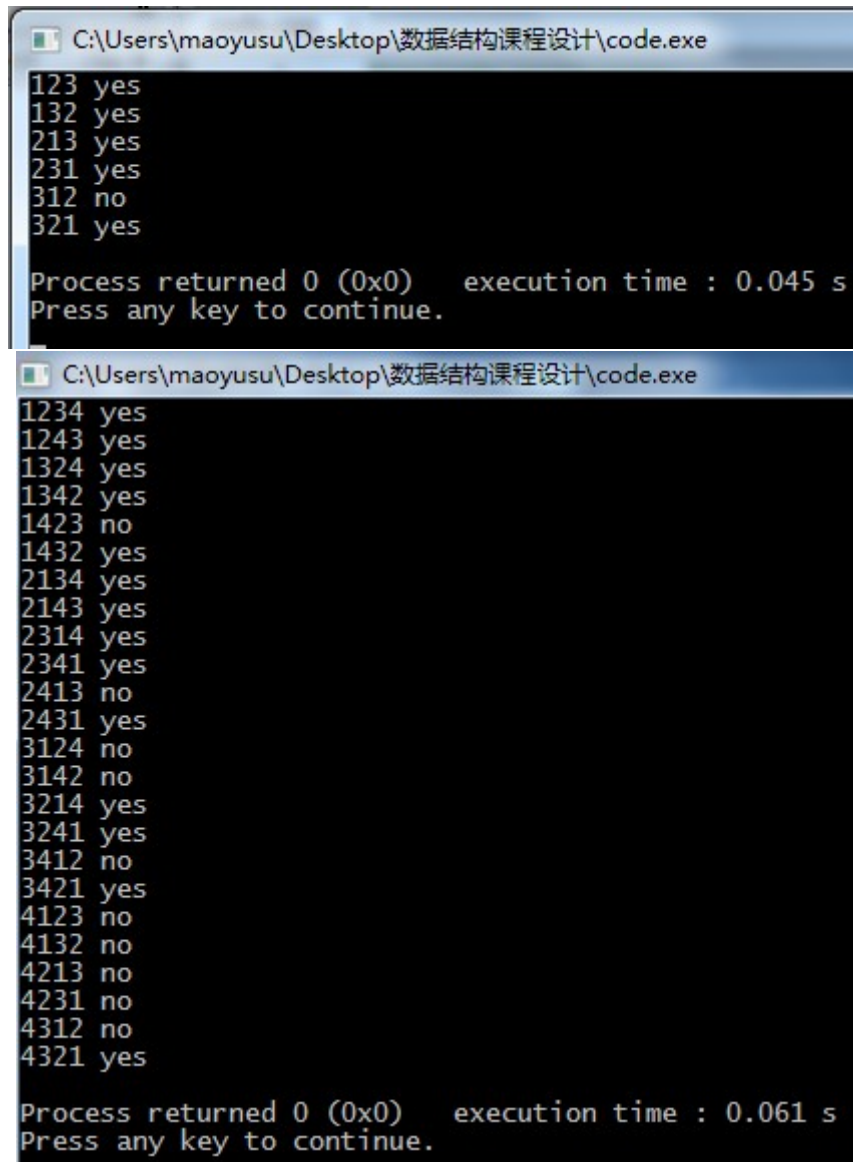
        else if(a <= depth) {s.push(a++);}

        else {return false;}
    }
    return true;
}

```

3.2 测试结果及分析

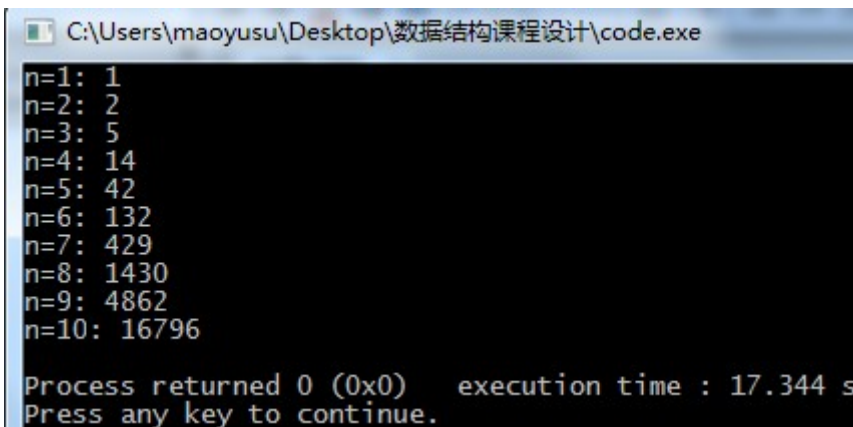
首先给出的是在 $n = 3$ 和 $n = 4$ 时的测试情况, 运行结果给出了每一种序列是否是合法序列.



```
C:\Users\maoyusu\Desktop\数据结构课程设计\code.exe
123 yes
132 yes
213 yes
231 yes
312 no
321 yes
Process returned 0 (0x0)  execution time : 0.045 s
Press any key to continue.

C:\Users\maoyusu\Desktop\数据结构课程设计\code.exe
1234 yes
1243 yes
1324 yes
1342 yes
1423 no
1432 yes
2134 yes
2143 yes
2314 yes
2341 yes
2413 no
2431 yes
3124 no
3142 no
3214 yes
3241 yes
3412 no
3421 yes
4123 no
4132 no
4213 no
4231 no
4312 no
4321 yes
Process returned 0 (0x0)  execution time : 0.061 s
Press any key to continue.
```

然后我们测试 $n = 1$ 到 10 时, 对应的出站序列总数, 我们发现与所求通项是吻合的.



```
C:\Users\maoyusu\Desktop\数据结构课程设计\code.exe
n=1: 1
n=2: 2
n=3: 5
n=4: 14
n=5: 42
n=6: 132
n=7: 429
n=8: 1430
n=9: 4862
n=10: 16796
Process returned 0 (0x0)  execution time : 17.344 s
Press any key to continue.
```

第四章 总结和问题

大二一年的学习，我们掌握了更多的编程的知识，也学习了更多的学科，这次的课题设计，不仅是不同学科之间的碰撞和交融，更是所学知识得以综合利用的一种体现，是自身综合能力的一次检验和锻炼，能有效的将所学的知识运用到实际，将不同的学科联系在一起，达到解决相关实际问题的目的。通过相关知识理论的了解和学习，终于设计出一种比较实际而又简洁的程序，很好的解决了二部图最大匹配问题。虽然这个计算程序已开发完成了，它也实现了我们所要解决的问题，但它仍有许多需要改进的地方。

1. 程序运行效率不高, 搜索算法的复杂度太大, 不知可否优化.
2. 我们所求得的数列为 catalon 数列, 还有许多方法 (比如生成函数法) 可以求出它的通项, 有待学习和扩充.

参考文献

- [1] 许胤龙. 组合数学引论 (第 2 版) [M]. 合肥: 中国科学技术大学出版社, 2010.
- [2] knuth. 具体数学: 计算机科学基础 [M]. 北京: 人民邮电出版社, 2013
- [3] 刘汝佳. 算法竞赛入门经典 [M]. 北京: 清华大学出版社, 2009

致 谢

一份课程设计的总结，一份对老师的感谢。首先，必须感谢江老师的悉心教导；过去的几个学期，正是因为有您这样兢兢业业的老师，我们才可以在稳步的迈进程序设计的大门，正是由于您的认真负责，我们才可以在这个贪玩的年龄里，在这个易受外物影响的年龄里能够很好的学到应该掌握的知识，能够很好的将学到的运用于实际；

其次，必须感谢为了这份课程设计奉献过自己努力和给予帮助的同学，正是这份友谊，这份合作，我们才可以将这份课题设计圆满的完成，能够在这份设计中感受到友谊，同时也使自己得到相应的锻炼。谢谢身边的每一个给予帮助的人，感谢生命中每一个孜孜奉献的每一个人，正是因为你们，所以我在这儿，健康而幸福的活着。

附录：程序代码

```
#include <iostream>
#include <stack>
using namespace std;
int vis[11], target[11];
int num;
bool test(int *target, int depth)
{
    stack<int> s;

    int a = 1, b = 1;
    while(b <= depth)
    {
        if(a == target[b]) {a++; b++;}

        else if(!s.empty() && s.top() == target[b]) {s.pop(); b++;}

        else if(a <= depth) {s.push(a++);}

        else {return false;}
    }
    return true;
}
void dfs(int cur, int depth)
{
    if(cur == depth)
    {
#ifdef TEST
        for(int i = 1; i <= depth; i++)
            cout<<target[i];
#endif
    }
}
```

```

        if(test(target, depth))
            cout<<" "<<"yes";
        else
            cout<<" "<<"no";
        cout<<endl;
    #endif
        if(test(target, depth)) num++;

    }
    else
    {
        for(int i = 1; i <= depth; i++)
        {
            if(vis[i]) continue;
            vis[i] = 1;
            target[cur+1] = i;
            dfs(cur+1, depth);
            vis[i] = 0;
        }
    }
}

int main()
{
    for(int i = 1; i <= 10; i++)
    {
        num = 0;
        dfs(0,i);
        cout<<num<<endl;
    }
    return 0;
}

```