# 实习报告

车厢调度问题

班级：电信1808班 姓名：黄嘉翰 学号：U201813468 完成日期：2019/4/10

## 需求分析

1. 假设停在铁路调度站入口处的车厢序列的编号依次为1，2，3…….，n，设计一个程序，求出所有可能的长度为n的车厢序列

2. 本程序要利用栈的顺序储存结构实现栈的基本操作，程序对栈的操作必须借助于基本操作进行

3. 本程序的输入方式为：在计算机终端上显示“提示信息”后，由用户在键盘上输入数字；运算结果会显示在其后

4. 测试数据：

n=3 n=4

1: 3 2 1 1: 4 3 2 1

2: 2 3 1 2: 3 4 2 1

3: 2 1 3 3: 3 2 4 1

4: 1 3 2 4: 3 2 1 4

5: 1 2 3 5: 2 4 3 1

6: 2 3 4 1

7: 2 3 1 4

8: 2 1 4 3

9: 2 1 3 4

10: 1 4 3 2

11: 1 3 4 2

12: 1 3 2 4

13: 1 2 4 3

14: 1 2 3 4

### 2. 概要设计

1.全局变量定义

typedef int SElemType;

typedef int Status;

long total = 0;

int N = 5;

2.栈的定义

typedef struct stacklist

void Init(SqStack \*s)

Status Push(SqStack \*s, SElemType e)

SElemType Pop(SqStack \*s)

Status Empty(SqStack \*s)

void Print(SqStack \*s)

3.本程序包含两个模块

可能性计算模块：

void process(SqStack \*temp, SqStack \*output, SElemType n, SElemType m){

当入口有车时，将车推入栈中，计算可能性后将车推出

当栈不为空时，将栈中车推到出口，计算可能性后将车推回栈中

当出口满时，打印出口车序

}

主程序模块：

int main(){

初始化栈和出口

输入车箱数

计算可能性

}

### 3. 详细设计

1.栈的数据结构

typedef struct{

SElemType \*base;

SElemType \*top;

int stacksize;

}SqStack;

void Init(SqStack \*s){

s->base=(SElemType \*)malloc(N \* sizeof(SElemType));

if(!s->base) exit(0);

s->top=s->base;

s->stacksize=N;

}

Status Push(SqStack \*s, SElemType e){

if(s->top - s->base >= s->stacksize){

s->base = (SElemType \*)realloc(s->base, (s->stacksize + 5) \* sizeof(SElemType));

if(!s->base) exit(0);

s->top = s->base + s->stacksize;

s->stacksize += 5;

}

\*s->top=e;

s->top++;

}

SElemType Pop(SqStack \*s){

if(s->top==s->base) return 0;

return \*--s->top;

}

Status Empty(SqStack \*s){

if(s->top==s->base) return 1;

else return 0;

}

void Print(SqStack \*s){

SElemType \*p;

p=s->base;

printf("%5ld: ", total);

while(p!=s->top){

printf("%3d", \*p++);

}

printf("\n");

}

2.核心算法

void process(SqStack \*temp, SqStack \*output, SElemType n, SElemType m){

if(n <= m){

Push(temp, n);

process(temp, output, n+1, m);

Pop(temp);

}

if(!Empty(temp)){

Push(output, Pop(temp));

process(temp, output, n, m);

Push(temp, Pop(output));

}

else if(n > m){

total++;

Print(output);

}

}

3.主程序

int main()

{ int m;

SqStack temp, output;

Init(&temp);

Init(&output);

printf("please input max number of train:");

scanf("%d",&m);

printf("\nthe result is:\n\n");

process(&temp, &output, 1, m);

}

### 4. 调试分析

1. 忽略了某个变量参数的标识“&”，加上后即解决

2. 程序无法循环执行，每次获得结果后即退出程序，以后改进

3. 对于核心算法的实现，一开始运用深复刻的方法以实现可能性的分支，空间复杂度较高，后采用复位的方法实现可能性的分支，降低了空间复杂度，但实现过程中因思考能力不足而耗费大量时间

4. 算法时空分析

时间复杂度为O(2^n)

空间复杂度O(1)

5. 本次实习作业熟练了栈的数据结构

### 5. 用户使用说明

1. 运行主界面

2. 输入车厢数

3. 显示结果

4. 退出

### 6. 测试结果

n = 2

1: 2 1

2: 1 2

n=5

1: 5 4 3 2 1

2: 4 5 3 2 1

3: 4 3 5 2 1

4: 4 3 2 5 1

5: 4 3 2 1 5

6: 3 5 4 2 1

7: 3 4 5 2 1

8: 3 4 2 5 1

9: 3 4 2 1 5

10: 3 2 5 4 1

11: 3 2 4 5 1

12: 3 2 4 1 5

13: 3 2 1 5 4

14: 3 2 1 4 5

15: 2 5 4 3 1

16: 2 4 5 3 1

17: 2 4 3 5 1

18: 2 4 3 1 5

19: 2 3 5 4 1

20: 2 3 4 5 1

21: 2 3 4 1 5

22: 2 3 1 5 4

23: 2 3 1 4 5

24: 2 1 5 4 3

25: 2 1 4 5 3

26: 2 1 4 3 5

27: 2 1 3 5 4

28: 2 1 3 4 5

29: 1 5 4 3 2

30: 1 4 5 3 2

31: 1 4 3 5 2

32: 1 4 3 2 5

33: 1 3 5 4 2

34: 1 3 4 5 2

35: 1 3 4 2 5

36: 1 3 2 5 4

37: 1 3 2 4 5

38: 1 2 5 4 3

39: 1 2 4 5 3

40: 1 2 4 3 5

41: 1 2 3 5 4

42: 1 2 3 4 5

n=6

1: 6 5 4 3 2 1

2: 5 6 4 3 2 1

3: 5 4 6 3 2 1

4: 5 4 3 6 2 1

5: 5 4 3 2 6 1

6: 5 4 3 2 1 6

7: 4 6 5 3 2 1

8: 4 5 6 3 2 1

9: 4 5 3 6 2 1

10: 4 5 3 2 6 1

11: 4 5 3 2 1 6

12: 4 3 6 5 2 1

13: 4 3 5 6 2 1

14: 4 3 5 2 6 1

15: 4 3 5 2 1 6

16: 4 3 2 6 5 1

17: 4 3 2 5 6 1

18: 4 3 2 5 1 6

19: 4 3 2 1 6 5

20: 4 3 2 1 5 6

21: 3 6 5 4 2 1

22: 3 5 6 4 2 1

23: 3 5 4 6 2 1

24: 3 5 4 2 6 1

25: 3 5 4 2 1 6

26: 3 4 6 5 2 1

27: 3 4 5 6 2 1

28: 3 4 5 2 6 1

29: 3 4 5 2 1 6

30: 3 4 2 6 5 1

31: 3 4 2 5 6 1

32: 3 4 2 5 1 6

33: 3 4 2 1 6 5

34: 3 4 2 1 5 6

35: 3 2 6 5 4 1

36: 3 2 5 6 4 1

37: 3 2 5 4 6 1

38: 3 2 5 4 1 6

39: 3 2 4 6 5 1

40: 3 2 4 5 6 1

41: 3 2 4 5 1 6

42: 3 2 4 1 6 5

43: 3 2 4 1 5 6

44: 3 2 1 6 5 4

45: 3 2 1 5 6 4

46: 3 2 1 5 4 6

47: 3 2 1 4 6 5

48: 3 2 1 4 5 6

49: 2 6 5 4 3 1

50: 2 5 6 4 3 1

51: 2 5 4 6 3 1

52: 2 5 4 3 6 1

53: 2 5 4 3 1 6

54: 2 4 6 5 3 1

55: 2 4 5 6 3 1

56: 2 4 5 3 6 1

57: 2 4 5 3 1 6

58: 2 4 3 6 5 1

59: 2 4 3 5 6 1

60: 2 4 3 5 1 6

61: 2 4 3 1 6 5

62: 2 4 3 1 5 6

63: 2 3 6 5 4 1

64: 2 3 5 6 4 1

65: 2 3 5 4 6 1

66: 2 3 5 4 1 6

67: 2 3 4 6 5 1

68: 2 3 4 5 6 1

69: 2 3 4 5 1 6

70: 2 3 4 1 6 5

71: 2 3 4 1 5 6

72: 2 3 1 6 5 4

73: 2 3 1 5 6 4

74: 2 3 1 5 4 6

75: 2 3 1 4 6 5

76: 2 3 1 4 5 6

77: 2 1 6 5 4 3

78: 2 1 5 6 4 3

79: 2 1 5 4 6 3

80: 2 1 5 4 3 6

81: 2 1 4 6 5 3

82: 2 1 4 5 6 3

83: 2 1 4 5 3 6

84: 2 1 4 3 6 5

85: 2 1 4 3 5 6

86: 2 1 3 6 5 4

87: 2 1 3 5 6 4

88: 2 1 3 5 4 6

89: 2 1 3 4 6 5

90: 2 1 3 4 5 6

91: 1 6 5 4 3 2

92: 1 5 6 4 3 2

93: 1 5 4 6 3 2

94: 1 5 4 3 6 2

95: 1 5 4 3 2 6

96: 1 4 6 5 3 2

97: 1 4 5 6 3 2

98: 1 4 5 3 6 2

99: 1 4 5 3 2 6

100: 1 4 3 6 5 2

101: 1 4 3 5 6 2

102: 1 4 3 5 2 6

103: 1 4 3 2 6 5

104: 1 4 3 2 5 6

105: 1 3 6 5 4 2

106: 1 3 5 6 4 2

107: 1 3 5 4 6 2

108: 1 3 5 4 2 6

109: 1 3 4 6 5 2

110: 1 3 4 5 6 2

111: 1 3 4 5 2 6

112: 1 3 4 2 6 5

113: 1 3 4 2 5 6

114: 1 3 2 6 5 4

115: 1 3 2 5 6 4

116: 1 3 2 5 4 6

117: 1 3 2 4 6 5

118: 1 3 2 4 5 6

119: 1 2 6 5 4 3

120: 1 2 5 6 4 3

121: 1 2 5 4 6 3

122: 1 2 5 4 3 6

123: 1 2 4 6 5 3

124: 1 2 4 5 6 3

125: 1 2 4 5 3 6

126: 1 2 4 3 6 5

127: 1 2 4 3 5 6

128: 1 2 3 6 5 4

129: 1 2 3 5 6 4

130: 1 2 3 5 4 6

131: 1 2 3 4 6 5

132: 1 2 3 4 5 6

### 7. 附录

源程序文件名清单：

Josephproblem.C //主程序