第五讲：数据与数据结构

姓名：丁保荣　　**学号**：171860509

2017 年 10 月 29 日

请独立完成作业，不得抄袭。
若参考了其它资料，请给出引用。
鼓励讨论，但需独立书写解题过程。

第一部分　作业

题目 (DH:2.10)

A permutation (a_1,$\cdots$,a_N) can be represented by a vector P of length N with P[I]=a_1. Design an algorithm which, given an integer N and a vector of integers P of length N, checks whether P represents any permutation of A_N.

解答：

bubblesort():

(1)set i=0;

(2)do the following N-1 times:

(2.1) point to the first element;

(2.2) do the following (N-1-i) times:

(2.2.1) compare the element pointed to with the next element;

(2.2.2) if the compared elements are in the wrong order, exchange them;

(2.2.3) point to the next element;

(2.3) let i++;

main():

B_1=bubblesort(P);

B_2=bubblesort(A_N);

if (B_1==B_2) then print "P represents a permutation of A_N";

else print "P doesn't represent any permutation of A_N";

题目 (DH:2.11)

Design an algorithm which, given a positive integer N, produces all the permutations of

A_N.

解答：

运用递归方法求解，用 for 循环每次确定第一个数为某一值，在每次 for 循环内运用递归求解规模递减 1 的全排列 (用一个状态数组标记前面已确定的数和还未确定的数)

```
1  //
2  // main.cpp
3  // An permutation
4  //
5  // Created by 丁保荣 on 2017/10/27.
6  // Copyright © 2017年 丁保荣. All rights reserved.
7  //
8
9  #include <iostream>
10 using namespace std;
11 int counter=0;
12 void arrange(int A[],int c[], int n,int k);
13 int main(void)
14 {
15     int A[10000],b[10000];
16     int n=5;
17     cin >>n;
18     for(int i=1;i<=n;i++)
19     {
20         cin>>A[i];
21         b[i]=0;
22     }
23     arrange(A,b,n,1);
24     cin.get();
25     return 0;
26 }
27
28 void arrange(int A[],int c[], int n,int k)
29 {
30     if (k==n)
31     {
32         for(int i=1;i<=n;i++)
33         {
34             if (c[i]==0)
35                 cout<<A[n]<<" ";
36             else
37                 cout<<A[c[i]]<<" ";
38         }
39         counter++;
40         cout<<counter<<endl;
41     }
```

```
42      else
43      {
44          for(int i=1;i<=n;i++)
45          {
46              if(c[i]!=0)
47                  continue;
48              else
49              {
50                  c[i]=k;
51                  arrange(A,c,n,k+1);
52                  c[i]=0;
53              }
54          }
55      }
56  }
```

题目 (DH:2.12)

We say that a permutation $\sigma = (a_1, \cdots, a_N)$ can be obtained by a stack, if it is possible to start from the input sequence (1,2,$\cdots$,N) and an empty stack S, and produce the output σ using only the following types of operations:

$\boldsymbol{read(X)}$: Read an integer from the input into variable X.
$\boldsymbol{print(X)}$: Print the integer currently stored in variable X on the output.
$\boldsymbol{push(X,S)}$: Push the integer currently stored in variable X on to the stack S.
$\boldsymbol{pop(X,S)}$: Pop the integer from the top of the stack S into variable X.(This operation is illegal if S is empty).

For example, the permutation(2,1) can be obtained by a stack, since the following series of operations

$$\boldsymbol{read(X), push(X,S), read(X), print(X), pop(X,S), print(X)}$$

applied to the input sequence (1,2) produces the output sequence (2,1).

A permutation can be obtained by a queue, if it can be similarly obtained from the input (1,2,$\cdots$,N), using an initially empty quene Q, and the operations $\boldsymbol{read(X), print(X)}$, and

$\boldsymbol{add(X,Q)}$: Add the integer currently stored in X to the rear of Q.

$\boldsymbol{remove(X,Q)}$: Remove the integer from the front of Q into X.(This opeartion is illegal if Q is empty.)

We can similarly speak of a permutation obtained by two stacks, if we permit the ***push*** and ***pop*** operations on two stacks S and S'.

(a) Show that the following permutations can be obtained by a stack:

i. (3,2,1).

ii. (3,4,2,1).

iii. (3,5,7,6,8,4,9,2,10,1).

(b) Prove that the following permutations cannot be obtained by s stack:

i. (3,1,2).

ii. (4,5,3,7,2,1,6).

(c) How many permutations of A_4 cannot be obtained by a stack?

解答:

(a)

i.read(1), push(1,S), read(2), push(2,S), read(3), print(3),
pop(2,S), print(2), pop(1,S), print(1)

ii.read(1), push(1,S), read(2), push(2,S), read(3), print(3),
read(4), print(4), pop(2,S), print(2), pop(1,S), print(1)

iii.read(1), push(1,S), read(2), push(2,S), read(3), print(3),
read(4), push(4,S), read(5), print(5), read(6), push(6,S),
read(7), print(7), pop(6,S), print(6), read(8), print(8),
pop(4,S), print(4), read(9), print(9), pop(2,S), print(2),
read(10), print(10), pop(1,S), print(1)

(b)

i.the first output is 3, so 1 and 2 should be stored in the stack.
So when output 1 and 2, 1 should be in front of 2,it contradicts with (3,1,2).
So the permutation(3,1,2) cannot be obtained by a stcak.

i.Let's consider the output:7 is before 2 and 6,
with the input we can condlude that 2 and 6 is stored in the stack.
and according to the input, 6 is stored above 2 in the stack.
So 6 should be printed earlier than 2.
So it contradicts with the output.
So the permutation(4,5,3,7,2,1,6) cannot be obtained by a stack.

(c)

There are ten permutations of A_4 cannot be obtained by a stack.
They are (3,1,2,4), (3,1,4,2), (3,4,1,2), (4,3,1,2), (2,4,1,3)
(4,2,1,3), (4,2,3,1), (1,4,2,3), (4,1,2,3), (4,1,3,2)

题目 (DH: 2.13)

Design an algorithm that checks whether a given permutation can be obtained by a stack. In case the answer is yes, the algorithm should also print the appropriate series of operations. In your algorithm, in addition to **read**, **print**, **push**, and **pop**, you may use the test is **empty(S)** for testing the emptiness of the stack **S**.

解答：

完整程序见 DH:2.15

完整程序见 DH:2.15

完整程序见 DH:2.15

题目 (DH: 2.14)

(a) Give series of operations that show that each of the permutations given in Exercise 2.12(b) can be obtained by a queue and also by two stacks.

(b) Prove that every permutation can be obtained by a queue.

(c) Prove that every permutation can be obtained by two stacks.

解答：

(a)

i.(3,1,2)

using a queue:

read(1), add(1,Q), read(2), add(2,Q), read(3), print(3),
remove(1,Q), print(1), remove(2,Q), print(2)

using two stacks:

read(1), push(1,S), read(2), push(2,S'), read(3),
print(3), pop(1,S), print(1), pop(2,S'), print(2)

ii.(4,5,3,7,2,1,6)

using a queue:

read(1), add(1,Q), read(2), add(2,Q), read(3), add(3,Q),
read(4), print(4), read(5), print(5), remove(1,Q), add(1,Q),
remove(2,Q), add(2,Q), remove(3,Q), print(3), read(6), add(6,Q),
read(7), print(7),remove(1,Q), add(1,Q), remove(2,Q), print(2),
remove(6,Q),add(6,Q), remove(1,Q), print(1), remove(6,Q), print(6).

using two stacks:

read(1), push(1,S), read(2), push(2,S)
read(3), push(3,S), read(4), print(4)
read(5), print(5), pop(3,S), print(3)
read(6), push(6,S'), read(7), print(7)
pop(2,S), print(2), pop(1,S), print(1)
pop(6,S'), print(6)

(b)

when the input is fixed,we want certain any permutation.We can do by the following steps.

When output the elements, there is a sequence. And the next output element can only

be in one of the two places:the queue or the input sequence.
(1)if the next output element is in the input sequence,we can add all the elements(if there exists) before the desired element to the queen, and then read the desired element,then output it.
(2)if the next output element is in the queue,we can remove all the elements(if there exists) before the desired element and then add then to the queen respectively.After that we can remove the desired element from the queue, and then output it.

(c)
The condition is similiar to (b)
When the input is fixed, we want certain any permutation. We can do by the following steps.
When output the elements, there is a sequence. And the next output element can only be in one of the threes places:the stack S, the stack S',the input sequence.
(1)if the next output element is in the input sequence, we can push all the elements(if there exists) before the desired element to the stack S (stack S' is also OK).After that, we can read the desired element, and then output it.
(2)if the next output element is in the stack S, we can pop all the elements(if there exists) before the desired element and then push them to the stack S' respectively. After that we can pop the desired element,and then output it.
(3)if the next output element is in the stack S', the operations are similar to (2).

题目 (DH: 2.15)

Extend the algorithm you were asked to design in Exercise 2.13, so that if the given permutation cannot be obtained by a stack, the algorithm will print the series of operations on two stacks that will generate it.

解答:

```
1  //
2  //  main.cpp
3  //  testify  a stack sequence improved
4  //
5  //  Created by 丁保荣 on 2017/10/27.
6  //  Copyright © 2017年 丁保荣. All rights reserved.
7  //
8
9
10 #include <iostream>
11 using namespace std;
12
13 int stack[10000]={0};//S栈
```

```
14 int stack2[10000]={0};//S'栈
15 int a[10000];//输出序列
16 int point=1;//S栈的顶部的指针
17 int point2=1;//S'栈的顶部的指针
18 int j=1;//输出序列的指针
19 int n;//输出长度
20 int place=0;//search函数中所要寻找的元素距离栈顶的距离
21
22
23
24 int  testify (int a [], int n);//测试是否能用一个栈操作
25 void output(int a[],int n);//用一个栈操作
26 void operation(int i);//output中的操作
27 void outputwithtwostacks(void);//用两个栈操作
28 int search(int s);    //查找元素在栈中的位置
29 void operation2(int i);//outputwithstacks中的操作
30 void betweenstacks(void);//在两个栈中间调节输出某一栈中的某一元素
31 int main(void)
32 {
33     cin>>n;  //输出序列的长度
34     for(int i=1;i<=n;i++)
35         cin>>a[i];  //读入输出序列
36     if ( testify (a,n)==0)
37     {
38         output(a,n);
39     } //测试是否能用一个栈操作，如果能，则输出操作步骤
40     else
41     {
42         outputwithtwostacks();
43     } //输出用两个栈操作的操作步骤
44     return 0;
45 }
46
47
48 int  testify (int a [], int n)
49 {
50     int compare=0;
51     for(int i=1;i<=n-1;i++)
52     {
53         compare=a[i];
54         for(int k=i+1;k<=n;k++)
55         {
56             if (a[k]<a[i])
57             {
58                 if(a[k]>compare)
59                 {
60                     cout<<"The permutation cannot be obtained by a stack!"<<endl;
61                     return 1;
62                 }
```

```
63                  else
64                      compare=a[k];
65              }
66          }
67      }
68      cout<<"The permutation can be obtained by a stack!"<<endl;
69      return 0;
70  }
71
72
73  void output(int a[],int n)
74  {
75      for(int i=1;i<=n;i++)
76      {
77          operation(i);
78
79      //所有元素已进栈或输出，下面处理栈内元素
80      }
81      while(point>1)
82      {
83          cout<<"pop("<<stack[point-1]<<",S)"<<endl;
84          cout<<"print("<<stack[point-1]<<")"<<endl;
85          stack[point-1]=0;
86          point--;
87      }
88  }
89
90  void operation(int i)
91  {
92      if(i<a[j]) //所求元素不在栈内，使i元素进栈
93      {
94          cout<<"read("<<i<<")"<<endl;
95          stack[point]=i;
96          point++;
97          cout<<"push("<<i<<",S)"<<endl;
98      }
99      else if(i==a[j]) //所求元素刚刚输入
100     {
101         cout<<"read("<<i<<")"<<endl;
102         cout<<"print("<<i<<")"<<endl;
103         j++;
104     }
105     else if(i>a[j]) //所求元素已进栈
106     {
107         cout<<"pop("<<stack[point-1]<<",S)"<<endl;
108         cout<<"print("<<stack[point-1]<<")"<<endl;
109         stack[point-1]=0;
110         point--;
111         j++;
```

```
112             operation(i); //将已进栈的所求元素输出，并递归直至i被处理（进栈或输出）
113         }
114
115     }
116     //above is the 2.13 solution
117     //
            ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
118     //不太华丽的分割线
119     //below is the improved function
120     void outputwithtwostacks(void)
121     {
122         for(int i=1;i<=n;i++)
123         {
124             operation2(i);
125         }
126         //所有元素已经栈或输出，下面处理栈内元素
127         while(j<=n)
128         {
129             betweenstacks();
130         }
131     }
132     void operation2(int i)
133     {
134         if(i<a[j]) //所求元素不在栈内，使i进栈
135         {
136             cout<<"read("<<i<<")"<<endl;
137             stack[point]=i;
138             point++;
139             cout<<"push("<<i<<",S)"<<endl;
140         }
141         else if(i==a[j]) //所求元素刚刚进栈
142         {
143             cout<<"read("<<i<<")"<<endl;
144             cout<<"print("<<i<<")"<<endl;
145             j++;
146         }
147         else if(i>a[j]) //所求元素已进栈
148         {
149             betweenstacks();
150             operation2(i); //将所求元素从栈内输出，并递归直至i被处理（进栈或输出）
151         }
152
153     }
154     int search(int s)
155     {
156         for(int i=1;i<=point-1;i++)
157         {
158             if(s==stack[i])
```

```
159         {
160             place=point-i-1;
161             return 1;
162         }
163     }
164     for(int i=1;i<=point2-1;i++)
165     {
166         if(s==stack2[i])
167         {
168             place=point2-i-1;
169             return 2;
170         }
171     }
172     return -1;
173 }
174 void betweenstacks(void)
175 {
176     //首先判断所要输出元素在哪个栈内，并将所求元素调节至栈顶，然后输出。
177     if(search(a[j])==1)
178     {
179         for(int j=1;j<=place;j++)
180         {
181 
182             cout<<"pop("<<stack[point-1]<<",S)"<<endl;
183             cout<<"push("<<stack[point-1]<<",S')"<<endl;
184             stack2[point2]=stack[point-1];
185             stack[point-1]=0;
186             point--;
187             point2++;
188         }
189         cout<<"pop("<<stack[point-1]<<",S)"<<endl;
190         cout<<"print("<<stack[point-1]<<")"<<endl;
191         j++;
192         stack[point-1]=0;
193         point--;
194     }
195     else if(search(a[j]==2))
196     {
197         for(int j=1;j<=place;j++)
198         {
199             cout<<"pop("<<stack[point-1]<<",S')"<<endl;
200             cout<<"push("<<stack[point-1]<<",S)"<<endl;
201             stack[point]=stack2[point2-1];
202             point2--;
203             point++;
204         }
205         cout<<"pop("<<stack2[point2-1]<<",S')"<<endl;
206         cout<<"print("<<stack2[point2-1]<<")"<<endl;
207         j++;
```

```
208         stack2[point2−1]=0;
209         point2−−;
210     }
211 }
```

題目 (DH: 2.16)

Consider the treesort algorithm described in the text.

(a) Construct an algorithm that transforms a given list of integers into a binary search tree.

(b) What would the output of treesort look like if we were to reverse the order in which the subroutine **second-visit-traversal** calls itself recursively? In other words, we consistently visit the right offspring of a node before we visit the left one.

解答:

(a)

```
insert(i,T):
  if(i<T.value)
  {
    if(isempty(Left(T)))
      Left(T).value=i;
    else
      insert(i,Left(T));
  }
  else
  {
    if(isempty(Right(T)))
      Right(T).value=i;
    else
      insert(i,Right(T));
  }



main():
Read(n);
read(a[1]...a[n]);
root=a[1];
for(i=2;i<=n;i++)
{
  insert(a[i],root);
}
```

(b)we will have a permutation of decreasing sequence.

第二部分　订正

题目 (题号)
题目。

错因分析：　简述错误原因（可选）。

订正：
正确解答。

第三部分　反馈

你可以写：

- 对课程及教师的建议与意见
- 教材中不理解的内容
- 希望深入了解的内容
- 等