第 3 章 栈和队列 作业

姓名 王松宸 学号 2024201594 成绩

1. 算法题：实现一个特殊的栈，除了具备栈的基本功能外，该栈还能返回栈中最小元素。要求 pop, push, getMin 操作的时间复杂度都是 O(1)。

① typedef struct {
int *base, *top;
int *min_base, *min_top; //存放最小元素
int stacksize;} SqStack;

② int push (SqStack &S, int e)
{ if (S.top - S.base == stacksize) //栈满
{ int *newbase = (int*) realloc (S.base, newsize * sizeof(int));
if (!newbase) return ERROR;
S.top = newbase + S.stacksize;
S.base = newbase;
S.stacksize = newsize;}
int newsize = S.stacksize + SIZE;
*S.top++ = e; //入栈
if (S.min_base == S.min_top || e < *(S.min_top-1))
{ *S.min_top++ = e; //入最小元素栈} 补充扩充最小栈代码！
return OK;

③ int Pop (SqStack &S)
{ int pop_item;
if (S.base == S.top) //栈空
return ERROR;
pop_item = *(--S.top);
if (pop_item == *(S.min_top-1)) //最小元素栈做出调整
S.min_top--;
return pop_item;}

④ int getMin (SqStack &S)
{ if (S.base == S.top) //栈空
return ERROR;
int min_item = *(S.min_top-1);
return min_item;}

思想：建立一个最小元素栈可以使最小元pop后立刻用第二小的代替

2、使用两个栈 Stack 来实现一个队列，完成队列 Queue 的 Enqueue 和 Dequeue 操作。假设两个栈分别为 s1 和 s2，数据元素类型是 int。假设栈入栈、出栈和判空的操作分别是 Status Push(int), Status Pop(int &)和 bool IsEmpty()。

Stack s1; //第一个栈 →作为输入栈，口为队尾
Stack s2; //第二个栈 →作为输出栈，口为队头
//将 x 入队列，算法思想为：
//直接将x压入S1

```
Status Enqueue(int x){
if(Push(S1, x))
    return OK;
return ERROR;
}
```

//将队头元素出队列并存入 x，算法思想为：
// 若S1为空，直接pop S2
// 若S1不为空，则将S1中的元素pop并压入S2，然后pop S2

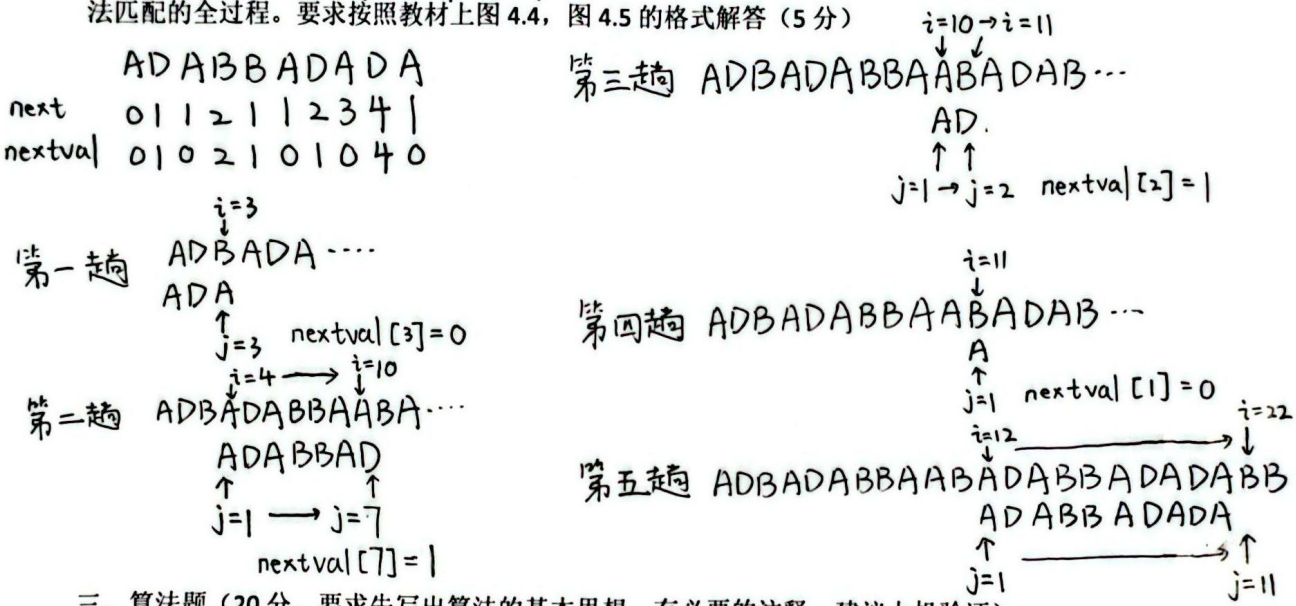
```
Status Dequeue(int &x){
  if(!IsEmpty(S1))
  { if(IsEmpty(S2)) return ERROR;
    x = Pop(S2);
    return OK;}
  else { while(!IsEmpty(S1))
         { int y = Pop(S1);
           if(!Push(S2,y))
             return ERROR;}
         x = Pop(S2);
         return OK;}
}
```

第4章 串 作业 姓名王松宸 学号2024201594 成绩_____

一、已知模式串s = 'aaab'，s = 'abcabca'，u = 'abacabacbbabacabb'，试分别求出它们的 *next* 函数值和 *nextval* 函数值（5分）

	a a a b	a b c a b c a	a b a c a b a c b b a b a c a b b
next	0 1 2 3	0 1 1 1 2 3 4	0 1 1 2 1 2 3 4 5 1 1 2 3 4 5 6 7
nextval	0 0 0 3	0 1 1 0 1 1 0	0 1 0 2 0 1 0 2 5 1 0 1 0 2 0 1 7

二、已知主串 s=ADBADABBAABADABBADADABB 模式串 t= ADABBADADA，画出改进 KMP（基于 *nextval* 函数值）算法匹配的全过程。要求按照教材上图 4.4，图 4.5 的格式解答（5分）



三、算法题（20分。要求先写出算法的基本思想、有必要的注释。建议上机验证）

查阅关于字符串编辑距离的资料。给定一个字符串 s 和字符表 v，设计算法生成所有和 s 编辑距离为 1 的字符串。该算法可以使用 string 和 vector 等 std 库。请注意输出的字符串要去重。同时思考：如何生成所有和 s 编辑距离为 2 的字符串？（不需要解答）

基本思想：三类操作，插入、删除、替换一个字符即生成一个符合要求的字符串

数据结构：用.set库中的set达到去重的目的，用string存字符串，方便操作

```
set<string> Levenshtein_1(char *s, char* v)
{ set<string> result; int n=strlen(s), m=strlen(v);
  for(int i=0; i<n; i++)
  { //删除
    string tmp=s;
    tmp.erase(i,1);
    result.insert(tmp);
   for(int j=0; j<m; j++)
   { //替换
     tmp=s;
     tmp[i]=v[j];
     result.insert(tmp);
    //插入(不包含末尾插入)
     tmp=s;
     tmp.insert(i,1,v[j]);
     result.insert(tmp);}}
  //插入(末尾)
  for(int j=0; j<m; j++)
  { string tmp=s;
    tmp.insert(n,1,v[j]);
    result.insert(tmp);}
  return result;}
//主要利用string带有的erase和insert函数实现
```