****



**程序设计综合实践课程报告**

**字符串实验**

**学生姓名**

**学院名称 智能与计算学部**

**专 业 工科试验班(智能与计算类)**

**学 号**

# 1. Simpsons’ Hidden Talents

## 1.1题目分析

**要找出第一个字符串的前缀和第二个字符串的后缀的最长相同部分，用kmp预处理的方法取得这个next数组，若最后一位是0则不存在，反之存在。**

## 1.2 题目代码（带注释）

|  |
| --- |
| #include <iostream>  #include <cstring>  #include <cstdio>  #include <cmath>  #include <algorithm>  #include <queue>  #define N 50010  using namespace std;  int nex[N];  int nextt[N];  char s1[N];  char s2[N];  void get\_next(char \*p)  {      /\*int q,k,m;      m=strlen(p);      next[0]=0;      for(q=1,k=0;q<m;++k)      {          while(k&&p[q]!=p[k])          k=next[k-1];          if(p[q]==p[k])          k++;          next[q]=k;      }\*/      int j,k;      j=0;      k=-1;      int len=strlen(p);      nextt[0]=-1;      while(j<len)      {          if(k==-1||p[j]==p[k])          {              j++,k++;              nextt[j]=k;          }          else          k=nextt[k];      }  }  void kmp(char \*p,char \*q)  {      int len2=strlen(q);      get\_next(p);      int i,k;      nex[0]=0;      i=0,k=0;      while(i<len2)      {          if(k==-1||p[k]==q[i])          {              i++,k++;              nex[i]=k;          }          else k=nextt[k];      }    }  int main()  {      while(scanf("%s%s",s1,s2)!=EOF)      {          kmp(s1,s2);          int n=strlen(s2);          if(nex[n]==0)          printf("0\n");          else          {              for(int i=0;i<nex[n];i++)              printf("%c",s1[i]);              printf(" %d\n",nex[n]);          }      }      return 0;  } |
|  |

# 2. Oulipo

## 2.1题目分析

**题目要求求一个字符串在另一个字符串第一次出现的位置  
利用KMP算法（基本与第一题一致，注释省略）**

## 2.2 题目代码（带注释）

|  |
| --- |
| #include <iostream>  #include <cstdio>  #include <cstring>  using namespace std;  char a[10010],b[1000010];  int ne[10010];  void fun(char \*p,int L)//找出子串的相同的前缀与后缀，用ne数组储存起来  {      int i,j;      i=0;j=-1;//大话数据结构的是从ne【1】开始的，但是我不会忽略ne【0】输入一个字符串      ne[0]=-1;      while (i<L)      {          if (j==-1||p[i]==p[j])          {              i++;j++;              ne[i]=j;          }          else          {              j=ne[j];          }      }  }  int main()  {      int n,i,j,k,lena,lenb,count;      scanf("%d",&n);      while (n--)      {          scanf("%s %s",&a,&b);          lena=strlen(a);          fun(a,lena);          lenb=strlen(b);          count=0;          i=-1;j=-1;          while (i<lenb)//这里与寻找ne数组很像          {              if (j==-1||a[j]==b[i])              {                  i++;                  j++;              }              else              {                  j=ne[j];              }              if (j==lena)              {                  count++;                  j=ne[j];              }          }          printf("%d\n",count);      }      return 0;  } |
|  |

# 3. Magical Forest

## 3.1题目分析

**给定N，M和K，表示在一个N\*M的棋盘上有K个棋子，给出K个棋子的位置和值，然后是Q次操作，对应的是：**

**1 a b ：交换a和b两行**

**2 a b : 交换a和b两列**

**3 a b ：查询a b这个位置上棋子的值，没有棋子的话输出0**

**解题思路:一开始X[i]=i，X[j]=j，如果需要交换i和j，那么就令X[i]=j,X[j]=i即可,因为N和M很大，所以用map映射。**

## 3.2 题目代码（带注释）

|  |
| --- |
| #include<algorithm>  #include<cstdio>  #include<cstring>  #include<cstdlib>  #include<iostream>  #include<vector>  #include<queue>  #include<stack>  #include<map>  #include<string>  #include<climits>  #include<cmath>  #define MAX 1100  #define LL long long  using namespace std;  map<int,int>row;  map<int,int>col;  map<int,map<int,int> >val;  int main()  {      #ifndef ONLINE\_JUDGE      freopen("cin.txt","r",stdin);      #endif      int T,kase=1;      cin>>T;      while(T--)      {          row.clear();          col.clear();          val.clear();          int x,y,k,c;          int rrow=1,ccol=1;          scanf("%d %d %d",&x,&y,&k);          while(k--)          {              scanf("%d %d %d",&x,&y,&c);              if(!row[x])                  row[x]=rrow++;              if(!col[y])                  col[y]=ccol++;              x=row[x],y=col[y];              val[x][y]=c;          }          printf("Case #%d:\n",kase++);          int Q,tmp;          cin>>Q;          while(Q--)          {              scanf("%d %d %d",&c,&x,&y);              if(c==1)              {                  tmp=row[x];                  row[x]=row[y];                  row[y]=tmp;              }              else if(c==2)              {                  tmp=col[x];                  col[x]=col[y];                  col[y]=tmp;              }              else                  printf("%d\n",val[row[x]][col[y]]);          }      }      return 0;  } |
|  |

# 4. Crazy Search

## 4.1题目分析

**给定一个字符串，其中含有不同的字母数量为m，现在求这个字符串中有多少个长度为n且长的互不相同的字符子串，用hash判重。**

## 4.2 题目代码（带注释）

|  |
| --- |
| #include<iostream>  #include<string>  #include <string.h>    using namespace std;    const int N = 16e6;  bool hashval[N + 1];  char digit[26];    int main()  {      int t, n, nc;      cin >> t;      while(t--) {          memset(hashval, 0, sizeof(hashval));          memset(digit, 0, sizeof(digit));            string s;          cin >> n >> nc >> s;            int k = 0;          for(int i = 0; s[i]; i++)              if(digit[s[i] - 'a'] == 0)                  digit[s[i] - 'a'] = k++;            int len = s.size() - n + 1;          int ans = len;          for(int i=0; i < len; i++) {              int hv = 0;              for(int j = i; j < i + n; j++)                  hv = hv \* nc + digit[s[j] - 'a'];              hv %= N;                if(hashval[hv])                  ans--;              else                  hashval[hv] = true;          }            cout << ans << endl;            if(t)              cout << endl;      }        return 0;  } |
|  |

# 5. Long Long Message

## 5.1题目分析

**这道题要求的是两个串的最长公共连续子串，在求出第一个数组之后，再把第二个数组区分出来，只要其中一个第二个数组是属于第一串，另外一个属于第二串，那么我们可以求得其最大值，之所以可以这样做，是因为第二个数组已经对字符串按字典序排好序了。**

## 5.2 题目代码（带注释）

|  |
| --- |
| #include <iostream>  #include <cstdio>  #include <cstring>  #include <vector>  #include <algorithm>  using namespace std ;  typedef long long ll ;  typedef unsigned long long ull ;  const int Maxn = 1e5 +10 ;  const int seed = 131 ;    string s1, s2 ;    bool check (int mid){      vector < ull > ve ;      ull p = 1 , \_Hash = 0 ;      for (int i = 0; i < mid; i++){          p \*= seed ;          \_Hash = \_Hash \* seed + s1[i] ;      }      ve.push\_back(\_Hash) ;      for (int i = mid; i < s1.size(); i++){          \_Hash = \_Hash \* seed + s1[i] - p \* s1[i - mid] ;          ve.push\_back(\_Hash) ;      }      sort(ve.begin(), ve.end()) ;      \_Hash = 0 ;      for (int i = 0; i < mid; i++){          \_Hash = \_Hash \* seed + s2[i] ;      }      if (binary\_search(ve.begin(), ve.end(), \_Hash)) return true ;      for (int i = mid; i < s2.size(); i++){          \_Hash = \_Hash \* seed + s2[i] - p \* s2[i - mid] ;          if (binary\_search(ve.begin(), ve.end(), \_Hash)) return true ;      }      return false ;  }    int main (){      cin >> s1 >> s2 ;      int len1 = s1.size() ;      int len2 = s2.size() ;      int right = min(len1, len2) ;      int left = 0 ;      int ans = 0 ;      while (left <= right){          int mid = (left + right) >> 1 ;          if (check(mid)) left = mid + 1 ,ans = mid ;          else right = mid - 1 ;      }      cout << ans << endl ;      return 0 ;  } |
|  |