判断唯一可译码

ismdeep

2013-10-17 10:56:58

1 唯一可译码

任意有限长的码元序列，只能被唯一地分割成一个个的码字，便称为唯一可译码。

2 判断唯一可译码

将码C中所有可能的尾随后缀组成一个集合F，当且仅当集合F中没有包含任一码字，则可判断此码C为唯一可译变长码。如何构成集合F，可以如下进行。

首先，观察码C中最短的码字是否是其他码字的前缀。若是，将其所有的可能的尾随后缀排列出。而这些尾随后缀又可能是某些码字的前缀，再将由这些尾随后缀产生的新的尾随后缀列出。

然后再观察这些新的尾随后缀是否是某些码字的前缀，再将产生的后缀列出。

依次下去，直至没有一个尾随后缀是码字的前缀或没有新的尾随后缀产生为止。这样，首先获得由最短的码字能引起的所有尾随后缀。接着，按照上述步骤将次短的码字、…等等，所有码字可能产生的尾随后缀全部列出。

由此，得到由码C的所有可能的尾随后缀组成的集合F。

3 实现代码

```
#include <iostream>
#include <stdlib.h>
#include <string.h>

using namespace std;

struct strings
{
    char *string;
    struct strings *next;
};


struct strings  Fstr, *Fh, *FP;
void outputstr(strings *str)
{
        do
        {
                cout<<str->string<<endl;
                str = str->next;
        }while(str);
        cout<<endl;
}
inline int MIN(int a, int b)
{    return a>b?b:a;     }
inline int MAX(int a, int b)
{    return a>b?a:b;     }
#define length_a (strlen(CP))
```

```
#define length_b (strlen(tempPtr))
//判断一个码是否在一个码集合中，在则返回，不在返回01
int comparing(strings *st_string,char *code)
{
        while(st_string->next)
        {
                st_string=st_string->next;
                if(!strcmp(st_string->string,code))
                        return 0;
        }
        return 1;
}

//判断两个码字是否一个是另一个的前缀，如果是则生成后缀码
void houzhui(char *CP,char *tempPtr)
{
        if (!strcmp(CP,tempPtr))
        {
                cout<<"集合和集合中有相同码字CF:"<<endl
                        <<CP<<endl
                        <<"不是唯一可译码码组!"<<endl;
                exit(1);
        }
        if (!strncmp(CP, tempPtr, MIN(length_a,length_b)))
        {
                struct strings *cp_temp;
                cp_temp=new (struct strings);
                cp_temp->next=NULL;
                cp_temp->string=new char[abs(length_a-length_b)+1];
                char *longstr;
                longstr=(length_a>length_b ? CP : tempPtr);//将
长度长的码赋给longstr

                //取出后缀
                for (int k=MIN(length_a,length_b); k<MAX(length_a,length
                        cp_temp->string[k - MIN(length_a,length_b)]=lon
                cp_temp->string[abs(length_a-length_b)]=NULL;
                //判断新生成的后缀码是否已在集合里，不在则加入集合FF
                if(comparing(Fh,cp_temp->string))
                {
                        FP->next=cp_temp;
                        FP=FP->next;
                }
        }
}


int main()
```

```
{
        cout<<"\t\唯一可译码的判断t!\n"<<endl;
        struct strings  Cstr, *Ch, *CP,*tempPtr;
    Ch=&Cstr;
        CP=Ch;
    Fh=&Fstr;
        FP=Fh;
        char c[]="C :";
    Ch->string=new char[strlen(c)];
    strcpy(Ch->string, c);
    Ch->next=NULL;
    char f[]="F :";
    Fh->string=new char[strlen(f)];
    strcpy(Fh->string, f);
    Fh->next=NULL;
//输入待检测码的个数
        int Cnum;
        cout<<"输入待检测码的个数:";
        cin>>Cnum;
        cout<<"输入待检测码"<<endl;
        for(int i=0; i<Cnum; i++)
    {
                cout<<i+1<<" : ";
                char tempstr[10];
                cin>>tempstr;
                CP->next=new (struct strings);
                CP=CP->next;
                CP->string=new char[strlen(tempstr)] ;
                strcpy(CP->string, tempstr);
                CP->next = NULL;
    }
        outputstr(Ch);
    CP=Ch;
    while(CP->next->next)
    {
                CP=CP->next;
                tempPtr=CP;
                do
                {
                        tempPtr=tempPtr->next;
                        houzhui(CP->string,tempPtr->string);
                }while(tempPtr->next);
    }
    outputstr(Fh);
        struct strings *Fbegin,*Fend;
    Fend=Fh;
    while(1)
    {
```

```
                    if(Fend == FP)
                    {
                cout<<"是唯一可译码码组!"<<endl;
                        exit(1);
                    }
                    Fbegin=Fend;
                    Fend=FP;
                    CP=Ch;
                    while(CP->next)
                    {
            CP=CP->next;
                        tempPtr=Fbegin;
                        for(;;)
                        {
                            tempPtr=tempPtr->next;
                            houzhui(CP->string,tempPtr->string);
                            if(tempPtr == Fend)
                                break;
                        }
                    }
                    outputstr(Fh);//输出集合中全部元素F
    }
    return 0;
}
```