第三次作业题解&&总结

第一题 连续线段

- 建议将线段以结构体形式存储，并按照x1的大小进行排序
- 判断连接线段时，注意可能一个短线段可以连接两个大线段成为一个更大的线段

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int x1,x2,y1,y2;//(x1,y1) (x2,y2)
    struct node *next;//指向该连续线段的下一个单线段的指针
    int mark;//表示该单线段在整个连续线段中的位置（起始为1
};
struct node s[1005];
int cmp(const void *a,const void *b)
{
    struct node a1 = *(struct node *)a;
    struct node b1 = *(struct node *)b;
    if(a1.x1 < b1.x1) return -1;
    else return 1;
}
int main()
{
    struct node tmp,*p=NULL;
    int n,cntx,cnty,maxn,maxx,maxy;
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        scanf("%d%d%d%d",&s[i].x1,&s[i].y1,&s[i].x2,&s[i].y2);
        s[i].next=NULL;
        s[i].mark=1;
    }
    //将结构体数组按照x1从小到大排序
    qsort(s,n,sizeof(s[0]),cmp);

    //maxn记录最大连续线段数，至少为1
    maxn=1;
    for(int i=0;i<n-1;i++)
    for(int j=i+1;j<n;j++)
    {
        //如果可以首尾相连
        if(s[i].x2==s[j].x1&&s[i].y2==s[j].y1)
        {
            s[i].next = &s[j];
            s[j].mark = s[i].mark + 1;
            if (s[j].mark > maxn) {
                maxn = s[j].mark;
```

```
            }
        }
    }

    for(int i=0;i<n;i++)
    {
        p=&s[i];
        //如果s[i]是一个连续线段起点
        if(s[i].mark==1)
        {
            while(p!=NULL)
            {
                if(p->mark==maxn)
                {
                    maxx=s[i].x1;
                    maxy=s[i].y1;
                    break;
                }
                p=p->next;
            }
        }
    }

    printf("%d %d %d",maxn,maxx,maxy);
    return 0;
}
```

第二题 空闲空间申请模拟

- 构建循环链表
- 遍历循环链表，寻找最适合结点，若没找到则**位置不变**
- 注意！链表删除时注意仅剩一个结点的情况，需要特判删除
- 在对链表进行操作时注意判断头结点是否为空

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
    int start;
    int size;
    struct node *next;
}*List;
int n;
List find(List L,int size) {
    List q=L,t=L;
    int best=-1,cnt=n;
    if (L==NULL) {
```

```
        return NULL;
    }
    //所求best为差距最小空间
    while(cnt--) {
        if(t->size>=size) {
            if(t->size-size < best ||best == -1) {
                best=t->size - size;
            }
        }
        q=t;
        t=t->next;
    }

    //printf("find best is :%d\n",best);

    if(best == -1) {
        //未找到空间
        return L;
    } else {
        cnt=n;
        while(cnt--) {
            //找到要处理的这块空间
            if (t->size == size+best) {
                if(best==0) {
                    //删除该结点
                    if(n==1) {
                        n--;
                        return NULL;
                    } else {
                        List m=q->next;
                        q->next= m->next;
                        free(m);
                        t=q->next;
                        n--;
                    }
                } else {
                    t->size-=size;
                }
                return t;
            } else {
                q=t;
                t=t->next;
            }
        }
    }
    return L;
}
int main(){
    List p=NULL,q=NULL,head=NULL;
    int request;
    scanf("%d",&n);
    //构建头结点为head的链表
```

```
    for(int i=0;i<n;i++) {
        p=(List)malloc(sizeof(struct node));
        p->next = NULL;
        scanf("%d%d",&p->start,&p->size);
        if(head==NULL) {
            head=p;
            q=p;
        } else {
            q->next = p;
            q=p;
        }
    }
    //头尾相连
    p->next=head;

    p=head;//p为当前节点
    int cnt=n;//当前链表中结点数

    while(~scanf("%d",&request)) {
        if(request==-1) {
            break;
        }
        //从p开始寻找并分配最适合request的空间
        p=find(p,request);
    }

    cnt=n;
    while(cnt--) {
        if(p==NULL) {
            break;
        }
        printf("%d %d\n",p->start,p->size);
        p=p->next;
    }

    return 0;
}
```

第三题 多项式

- 注意本题输入时对数据的处理和判断本行条件

```
//在读入第一行时
int ix,epx;
char x;
while(1) {
    scanf("%d%d",&ix,&epx);
    //读入两个连续数值后的字符，如果没有到末尾就是空格
```

```
    scanf("%c",&x);
    if (x == '\n') {
        break;
    }
    ......
}

//读入第二行数据
while(~scanf("%d%d",&p->ix,&p->epx)) {
......
}
```

- 注意不是每个多项式都有常数项
- 注意处理系数为0的项

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
    int epx;
    int ix;
    struct node *next;
}*List;
//删除q的下一个结点
void del(List q)
{
    List t;
    t=q->next;
    q->next=t->next;
    free(t);
}
//将p按序插入以head为头结点的链表中
List insert(List head,List p)
{
    List L=head,q=head;
    if(L==NULL) L=p;
    while(L!=NULL)
    {
        if(L->epx==p->epx)
        {
            L->ix = L->ix + p->ix;
            if(L->ix==0)
            del(q);
            break;
        }
        else if(L->epx>p->epx&&L->next==NULL)
        {
            L->next=p;
            break;
```

```
        }
        else if(L->epx > p->epx&&L->next->epx<p->epx)
        {
            p->next=L->next;
            L->next=p;
            break;
        }
        else
        {
            q=L;
            L=L->next;
        }
    }
    return head;
 }
int main()
{
    char x;
    List h1=NULL,h2=NULL,h3=NULL,p=NULL,q=NULL,t=NULL;
    //读入数据，建立以h1为头结点的链表
    while(1)
    {
        p=(List)malloc(sizeof(struct node));
        p->next=NULL;
        scanf("%d%d",&p->ix,&p->epx);
        if(h1==NULL)
        {
            h1=p;
            q=p;
        }
        else
        {
            q->next=p;
            q=p;
        }
        scanf("%c",&x);
        if(x=='\n')
        break;
    }
    p=(List)malloc(sizeof(struct node));
    p->next=NULL;
    //建立以h2为头结点的链表
    while(~scanf("%d%d",&p->ix,&p->epx))
    {
        if(h2==NULL)
        {
            h2=p;
            q=p;
        }
        else
        {
            q->next=p;
```

```
            q=p;
        }
        p=(List)malloc(sizeof(struct node));
        p->next=NULL;
    }

    //建立以h3为头结点的链表(多项式相乘结果
    //用q遍历h1各结点
    q=h1;
    while(q!=NULL)
    {
        //用p遍历h2各结点
        p=h2;
        while(p!=NULL)
        {
            //t结点结果为q指向结点和p指向结点相乘结果
            t=(List)malloc(sizeof(struct node));
            t->next=NULL;
            t->epx=q->epx + p->epx;
            t->ix=q->ix * p->ix;
            //将t插入以h3为头结点的链表
            if(h3==NULL)
                h3=t;
            else
                h3=insert(h3,t);
            p=p->next;
        }
        q=q->next;
     }

    p=h3;
    while(p!=NULL)
    {
        printf("%d %d ",p->ix,p->epx);
        p=p->next;
    }

    return 0;
}
```

第四题 文件加密

- 用flag数组标志该字符是否已在密文中
- 注意在加密文件时ascii码在32-126中的字符才需要进行转换
- 在对已连接成循环链表的密文进行删除、计算对应密文时，记得先删除当前结点；当前位置移到后一个结点时，移动次数+1

- 同样注意仅剩一个结点的情况
- 在存储对应密钥时，同样可以用数组存，如字母h的对应密钥为a，那么key['h'-32]='a'

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
char key1[1005],key2[1005],file[2005];
int mark[1005],ss[150],ans[150];
int cnt=126-32+1;
typedef struct node
{
    char x;
    struct node *next;
}*List;
List work(List L,int k)
{
    List q=L,t=NULL;
    //delete L->next
    t=q->next;
    q->next=t->next;
    free(t);
    cnt--;
    for(int i=1;i<=k;i++)
    {
        q=L;
        L=L->next;
    }
    ans[k]=(int)L->x;
    return q;
}
int main()
{
    List p=NULL,head=NULL,q=NULL;
    FILE *in,*out;
    in=fopen("in.txt","r");
    out=fopen("in_crpyt.txt","w");
    int len,n=0;
    gets(key1);
    len=strlen(key1);
    //去重
    for(int i=0;i<len;i++)
    {
        if(mark[key1[i]] == 0) {
            key2[n++] = key1[i];
            mark[key1[i]]=1;
        }
    }
    //在去重后key后加上未出现字符
    for(char i=32;i<=126;i++)
    {
        if(mark[i]==0)
```

```
        {
            key2[n++] = i;
        }
    }
    key2[n] = '\0';

    //构建链表
    for(int i=0;i<n;i++)
    {
        p=(List)malloc(sizeof(struct node));
        p->x=key2[i];
        p->next=NULL;
        if(head==NULL)
        {
            head=p;
            q=p;
        }
        else
        {
            q->next=p;
            q=p;
        }
    }
    //头尾相连，形成环
    p->next=head;

    //p为第一个字符的前一个结点
    //cnt为当前链表中结点数
    while(cnt>0)
    {
        //特判
        if(cnt==1)
        {
            ans[p->x]=(int)key2[0];
            break;
        }
        //p为将要删除结点的前一个结点
        //p->next->x为循环次数
        p=work(p,p->next->x);
    }

    //fgets可以读入末尾'\n'
    while(fgets(file,2005,in)!=NULL)
    {
        int len2;
        len2=strlen(file);
        for(int i=0;i<len2;i++)
        {
            if(file[i]>=32&&file[i]<=126)
            fprintf(out,"%c",ans[file[i]]);
            else
            fprintf(out,"%c",file[i]);
```

```
        }
    }
    fclose(in);
    fclose(out);
    return 0;
}
```

第五题 词频统计

- 注意统计时应将大写字母转换为小写字母记录，但是输出文本不是只有小写字母！！！
- 注意一段包含其他字符的字符串应循环处理来提取其中单词，如ans.wer.c，该字符串包含ans、wer、c三个单词
- 一个合法单词全由字母组成，且单词两边或是其他字符(不是字母的字符)，或是开头或结尾
- 本题不判断单词正确性，只需要单词由连续字母组成即可

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>
char a[1055],b[105];
typedef struct node
{
    char s[105];
    int number;
    struct node *next;
}*List;
List insert(List L,char s1[])
{
    List p=(List)malloc(sizeof(struct node));
    p->next=NULL;
    p->number=1;
    strcpy(p->s,s1);
    if(L==NULL)
    L=p;
    else
    {
        List q=L;
        if(strcmp(s1,L->s)<0)
        {
            p->next=L;
            return p;
        }
        while(q)
        {
            if(strcmp(s1,q->s)==0)
            {
```

```
                q->number++;
                break;
            }
            else if(strcmp(s1,q->s)>0&&q->next==NULL)
            {
                q->next=p;
                break;
            }
            else if(strcmp(s1,q->s)>0&&q->next!=NULL)
            {
                if(strcmp(s1,q->next->s)<0)
                {
                    p->next=q->next;
                    q->next=p;
                    break;
                }
                else
                {
                    q=q->next;
                }
            }
        }
    }
    return L;
}
int main()
{
    FILE *in;
    List head=NULL,p=NULL;

    in=fopen("article.txt","r");
    while(fgets(a,1055,in)!=NULL)
    {
        int len=strlen(a),k=0,cnt=0;
        for(int i=0;i<len;i++)
        {
            if(isalpha(a[i]))
            {
                if(a[i]>='A'&&a[i]<='Z')
                a[i]+=32;
                b[cnt++]=a[i];
            }
            else
            {
                if(b[0]!='\0')
                {
                    b[cnt]='\0';
                    head=insert(head,b);
                }
                memset(b,'\0',sizeof(b));
                cnt=0;
            }
```

```
        }
        if(b[0]!='\0')
        {
            b[cnt]='\0';
            head=insert(head,b);
        }
    }
    p=head;
    while(p!=NULL)
    {
        printf("%s %d\n",p->s,p->number);
        p=p->next;
    }
    fclose(in);
    return 0;
}
```