程序设计实践（一）

哈尔滨工业大学 计算机学院
任课教师：孙大烈教授
助教：付万增

第九讲

字符串处理

一、KMP算法
二、字符串哈希
三、高精度运算

字符串

字符串(String)是由数字、字母、下划线组成的一串字符，
是字符的数组，记为 s="a0a1…an-1"，表示文本的数据类型。
举例： char string[] = "Hello, world!";

两个字符串相等的充要条件：
长度相等且各个对应位置上的字符都相等。

一般操作：
1.查找某个子串
2.插入一个子串
3.删除一个子串

问题1

字符串翻转

问题描述：给定字符串，输出翻转以后的字符串。

Input：
!dlrow,olleH

Output：
Hello,world!

Hint: int len = strlen(char* s);

```
int main()
{
    scanf("%s",s);
    int len = strlen(s);
    for(int i=0,j=len-1;i<j;i++,j--)
    {
        swap(s[i],s[j]);
    }
    printf("%s",s);
    return 0;
}
```

问题2

字符串分割

问题描述：给定一句英语（一行），完成单词的分割。
要求输出m行，每行一个单词，m是单词的个数。

Input:
Nice to ment you
Output:
Nice
to
ment
you
Hint: gets(char *s);用来

```
gets(s);
int num = 0,ptr = 0;
for(int i=0;s[i] != '\0';i++)
{
    if(isAlphabet(s[i]) ) tmp[num][ptr++] = s[i];
    else
    {
        if(ptr>0)
        {
            tmp[num][ptr] = '\0';
            num++;
            ptr = 0;
        }
    }
}
```

问题3

单词翻转

问题描述：给定一句英语（一行），完成单词的分割并逆序输出。

Input：

Nice to ment you

Output：

you ment to Nice

```
gets(s);
int len = strlen(s);
reveser(0,len-1);
int last = 0;
for(int i=0;i<=len;i++)
{
    if(!isAlphabet(s[i]))
    {
        reveser(last,i-1);
        last = i+1;
    }
}
puts(s);
```

问题4

字典序

问题描述：比较两个字符串的字典序大小。

Tip：从左到右逐个比较对应的字符的大小，直到不相等。

Input：

Buffer

Buff++

Output：

>

```
char compare(char* s,char* t)
{
    int ls = strlen(s);
    int lt = strlen(t);
    for(int i=0;i<=ls&&i<=lt;i++)
    {
        if(s[i] < t[i]) return '<';
        if(s[i] > t[i]) return '>';
    }
    return '=';
}
```

问题5

字典序

问题描述：排序n个字符串。

Input：	Output：
4	Affff
aAb	Hello
Affff	aAb
Hello	world
world	

```
void sort(int n)
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n-i-1;j++)
        {
            if(compare(s[j],s[j+1]) == 0)
            {
                swap(s[j],s[j+1]);
            }
        }
    }
}
```

问题6

模式匹配

问题描述：给定两个字符串，判断第二个字符串在第一个字符串中出现了多少次？

Input：

abababacab

aba

Output：

3

```
for(int i=0;i<ls;i++)
{
    bool flag = 1;
    for(int j=0;j<lt;j++)
    {
        if(s[i+j] != t[j])
        {
            flag = 0;
            break;
        }
    }
    ans += flag;
}
```

KMP算法

问题描述：给定两个字符串，判断第二个字符串在第一个字符串中出现了多少次？都在哪些位置出现？

举个栗子：

A[] = a b a b a b a a b a b a c b

B[] = a b a b a c b

定义next[j]: 当B[j+1]!=A[i]时A[i]应与B[next[j]+1]比较

我们的学习目标：用程序解决问题。

发现问题->解决问题->更好的解决问题

KMP算法

- KMP算法
- i: 0 1 2 3 4 5 6 7 8 9
- A: a b a b a b a a b a b a c b
- B: a b a b a c b
- j: 0 1 2 3 4 5 6

KMP算法

- KMP算法
- i: 0 1 2 3 4 5 6 7 8 9
- A: a b a b a b a a b a b a c b
- B: a b a b a c b
- j: 0 1 2 3 4 5 6

KMP算法

- KMP算法
- i: 0 1 2 3 4 5 6 7 8 9
- A: a b a b a b a a b a b a c b
- B: a b a b a c b
- j: 0 1 2 3 4 5 6

KMP算法

- KMP算法
- i: 0 1 2 3 4 5 6 7 8 9
- A: a b a b a b a a b a b a c b
- B: a b a b a c b
- j: 0 1 2 3 4 5 6

KMP算法

- KMP算法
- i: 0 1 2 3 4 5 6 7 8 9
- A: a b a b a b a a b a b a c b
- B: a b a b a c b
- j: 0 1 2 3 4 5 6

KMP算法

- KMP算法
- i: 0 1 2 3 4 5 6 7 8 9
- A: a b a b a b a a b a b a c b
- B: a b a b a c b
- j: 0 1 2 3 4 5 6

KMP算法

- KMP算法
- i: 0 1 2 3 4 5 6 7 8 9
- A: a b a b a b a a b a b a c b
- B: a b a b a c b
- j: 0 1 2 3 4 5 6

KMP算法

- KMP算法
- i: 0 1 2 3 4 5 6 7 8 9
- A: a b a b a b a a b a b a c b
- B: a b a b a c b
- j: 0 1 2 3 4 5 6

KMP算法

- KMP算法
- i: 0 1 2 3 4 5 6 7 8 9
- A: a b a b a b a a b a b a c b
- B: a b a b a c b
- j: 0 1 2 3 4 5 6

KMP算法

•KMP算法

```
j:      0 1 2 3 4 5 6
B:      a b a b a c b
next:-1-1 0 1 2-1-1
```

next:next[i]表示字符串[0..i]的前缀与后缀的一个最大匹配

定义：当B[j+1]!=A[i]时A[i]应与B[next[j]+1]比较

KMP算法


```
int kmp(char *A, char *B, int *next) {
    int j = -1, ret = 0;
    for (int i = 0; A[i]; i++) {
        while (j != -1 && A[i] != B[j + 1]) j = next[j];
        if (A[i] == B[j + 1]) j++;
        if (!B[j + 1]) {
            ret++;
            j = next[j];
        }
    }
    return ret;
}
```

KMP算法

- KMP算法一共两个步骤
- 一步是利用Next数组解决问题。
- 另一步就是求解Next数组。

- 求解next
- j:　　0 1 2 3 4 5 6
- B:　　a b a b a c b
- next:-1-1 0 1 2-1-1

KMP算法

- 求解next

```
void prekmp(char *B, int *next) {
    next[0] = -1;
    int j = -1;
    for (int i = 1; B[i]; i++) {
        while (j != -1 && B[i] != B[j + 1])
            j = next[j];
        if (B[i] == B[j + 1]) j++;
        next[i] = j;
    }
}
```

两类算法

- 确定性算法（现成代码）
  - 多种排序算法，二分（三分）查找，优先队列，最短路，最小生成树，kmp算法
  - （考验的是牢牢掌握和灵活应用）
- 非确定性算法（思想）
  - 贪心，搜索，分治，动态规划
  - （考验的是透彻理解和丰富经验）

我们站在巨人的肩膀上------编程序

哈希算法

问题描述：给定两个字符串，判断第二个字符串在第一个字符串中出现了多少次？都在哪些位置出现？

同样的问题，新算法，新思想。
哈希：建立一种映射关系，更便捷地存储，操作数据。

字符串哈希思路：将字符串映射为一个大数字。
比较字符串是否相等是O(N)的；
比较数字是O(1)的。

字符串哈希

为了完成字符串到数字的一种映射。
首先是观察具体例子，发现内在规律。
数字串：123412 --- 十进制
字母串：absdf --- 二十六进制
01串：01000111 --- 二进制
一般串：aba sdf，--#@@！ --- 128进制（ASCII码128个）

一个思路：直接利用进制转换，将其他进制形式的字符串转化为十进制的数字。

字符串哈希

一个思路：直接利用进制转换，将其他进制形式的字符串转化为十进制的数字。

可行性？

小数据可行，对于长字符串映射后的数字太大了，存不下来。

放弃？ Or 改进？

字符串哈希

继续观察和比较：
数字存储是连续的，0到max；
字符串没什么规律，afd123b,123bafds,a,b,fwew,离散分布；

建立离散数据到连续数据的直接映射，结果是映射得到的数据也是离散的，有很多数值很小的数字没有利用上。

为了更有效的利用所有数字，利用取模技术。

字符串哈希

继续观察：
k进制转换， H = H*k + s[i];

取模的话， H = (H*k + s[i]) mod q;
q取一个比较大的质数。

完成了k进制的字符串s[i] 到 十进制数B的一种映射，并且限制B的大小不会超过q。

这么容易？是否有问题？

字符串哈希

映射：单射？满射？双射？

双射是最好的，退而求其次，找接近双射的一个单射。

模运算下的进制转换哈希算法。

```
int mapping(char* s)
{
    int ans = 0;
    for(int i=0;s[i];i++)
    {
        ans = (ans*SEED + s[i])%q;
    }
    return ans;
}
```

字符串哈希

K进制转换哈希算法更便捷的应用：快速计算子串的哈希映射值。
子串是字符串的一段连续字母串子集。比如：12,34,23都是数字串12345的子串，但是15不是。

数字串123456 取出子串23 ， 可以利用123456/1000 再 123 % 100 = 23

```
int cal(int l,int r)
{
    int ans = (h[l] - (h[r] * k[r-l]) % q) % q;
    //printf("%d\n",ans);
    return ans < 0 ? ans + q : ans;
}
```

两种哈希算法

对于单射，会有多个值（字符串）映射到同一个数字处，这种情况称为冲突。

有错哈希：不解决冲突。但是错误率很小，几乎可以忽略。
无错哈希：解决冲突。方法，在冲突的地方，拉一条链表，也就是存储所有映射到这个地方的数据（字符串），从而可以等效于双射，甚至可以完成逆映射。

双射的好处：可以快速完成逆映射。
如果F(x) = y，对于任意y，x是确定的。

高精度加法

问题描述：大数加法。

Int的存储范围：-2147483648～2147483647

int64更大：-2^63~2^63+1

但是固定数据类型的存储空间毕竟有限，如果是一百位的数字相加怎么办？

数字即字符串。

想一想小学如何做加法？列竖式。

高精度加法：利用字符串的一系列操作模拟竖式的加法。

```
  212
+  32
= 244
```

高精度减法

问题描述：大数减法。

数字即字符串。

小学减法？列竖式。

```
  212
-  32
= 180
```

高精度减法：利用字符串的一系列操作模拟竖式的减法。

只不过加法考虑进位，减法考虑借位。

默认大数减小数。跟平时的思想是一样的，不过我们平时算太快忽略了。编程注重细节，很多我们平时忽略的东西，编程序时候是需要注意的。

高精度乘法

问题描述：大数乘法。
模拟乘法的竖式计算。
同样考虑进位。

问题描述：大数除法。
模拟除法。

模拟？就是把基本思想一步一步实现成代码，通常思路简单明确，步骤稍显繁琐。

回顾

今天你学会了什么？

一、KMP算法
二、字符串哈希
三、高精度运算

作业

1.熟悉字符串各种操作，学会使用字符串的各种库函数。

2.掌握KMP算法，理解算法含义并可以看懂代码。

3.了解字符串哈希，无错哈希思想。

4.掌握高精度加减法（提高编程能力）。

共同进步

- 一分耕耘，一分收获！
- 希望在今后的学习生活中共同进步！

谢谢！