算法分析与设计
第六章

为了给大家充足的时间完成project，最后一次实验报告截止时间为6月18日23：59

为了给大家充足的时间备战期末考，
本次实验报告完成两题上限十分，
完成一题上限九分

soj 1051 Biker's Trip Odomete

- 题意：给车前轮直径数，转圈数和时间，求自行车行驶距离和平均速度
- 单位换算：
- Pi = 3.1415927
- 5280 feet = 1 mile
- 12 inches = 1 foot
- 60 minutes = 1 hour
- 60 seconds = 1 minute
- 201.168 meters = 1 furlong
- 解法：直接根据上面的单位换算，计算行驶距离于平均速度即可
- 注意：精确到两位小数 printf(“%.2f”, val);

Soj 1007 To and Fro

- 题意：介绍了一种加密算法：将明文（要加密的信息）从左到右一列一列地写入给定列数的矩形中，在添加字符‘x’将它补满，接着从上到下一行一行地读出来形成密文（加密后的信息）。现在给你密文，要将它翻译成明文。
- 限制：列数2<=n<=20，密文

theresnoplacelikehomeonasnowynightx

toioy
hpknn
eleai
rahsg
econh
semot
nlewx

toioynnkpheleaigshareconhtomesnlewx

Soj 1007 To and Fro

- 直接根据题意模拟就可以了
- 代码：O(m)，m为密文长度

```
 4 void decrypt(char *input, char *output, int column)
 5 {
 6     int len = strlen(input);
 7     int row = len / column;
 8     const int maxn = 100;
 9     static char arr[maxn][maxn];
10     for (int i = 0, k = 0; i < row; i++)
11         for (int j = 0; j < column; j++, k++)
12             arr[i][j] = input[k];
13     for (int j = 0, k = 0; j < column; j++)
14         for (int i = 0; i < row; i++, k++)
15             output[k] = arr[i][j];
16     output[len] = 0;
17 }
```

Soj 1036 Crypto Columns

- 题意：与上一题类似，给定一个加密算法，现在知道密文，求明文（也就是设计解密算法）。
- 加密算法：
- 根据给定的keyword长度，确定矩形的列数，将明文从上到下一行一行地填入矩形。接着根据keyword中的字符字母顺序，依次写出字符代表一列的字符，最终形成密文。
- 约束：keyword长度n<=10，密文长度m<=100

Soj 1036 Crypto Columns

BATBOY
MEETMEBYTHEOLDOAKTREENTH

MEETME
BYTHEO
LDOAKT
REENTH

B A T B O Y
M E E T M E
B Y T H E O
L D O A K T
R E E N T H

EYDEMBLRTHANMEKTETOEEOTH

Soj 1036 Crypto Columns

- 同样是一道模拟题，直接模拟。
- 代码：O(m+nlogn)，m为密文长度

n为keyword长度

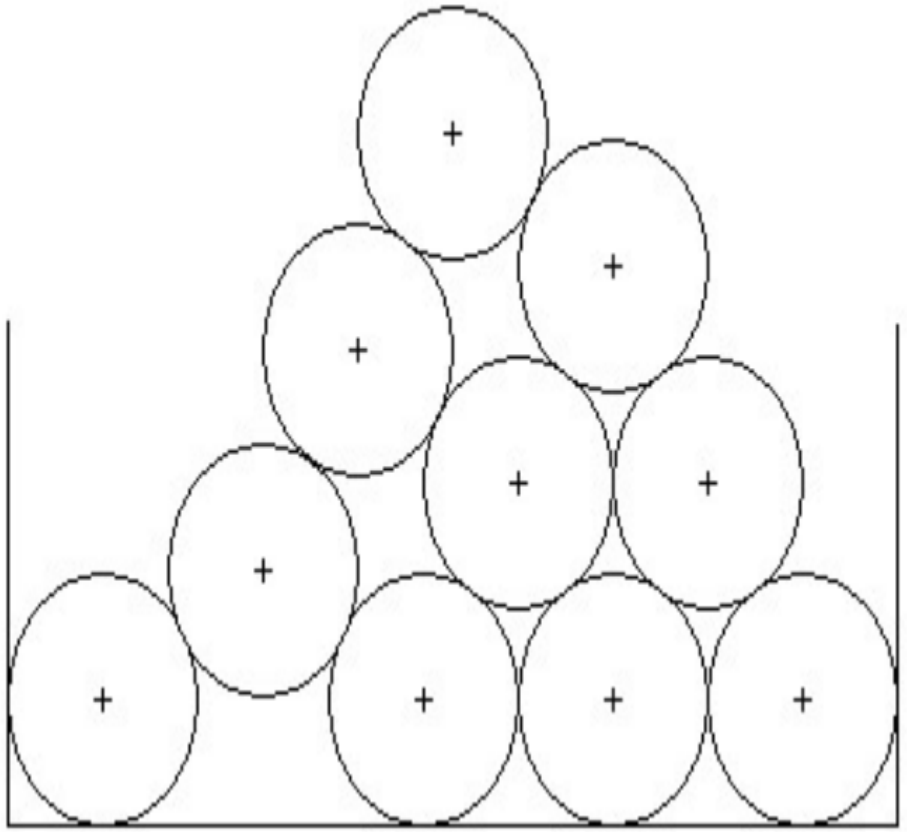
```
19 void decrypt(char *input, char *output, char *keyword)
20 {
21     int len = strlen(input);
22     int column = strlen(keyword);
23     int row = len / column;
24
25     vector< pair<char, int> > vec;
26     for (int i = 0; i < column; i++)
27         vec.push_back(make_pair(keyword[i], i));
28     sort(vec.begin(), vec.end());
29
30     const int maxn = 100;
31     static char arr[maxn][maxn];
32     for (int j = 0, k = 0; j < column; j++)
33     {
34         int pos = vec[j].second;
35         for (int i = 0; i < row; i++, k++)
36             arr[i][pos] = input[k];
37     }
38     for (int i = 0, k = 0; i < row; i++)
39         for (int j = 0; j < column; j++, k++)
40             output[k] = arr[i][j];
41     output[len] = 0;
42 }
```

Soj1206 1012 Stacking Cylinders

- 题意：
- 给出最底层的n个圆柱的位置，求最顶层的圆柱的位置
- 圆柱半径都为1
- 约束：1 <= n <= 10

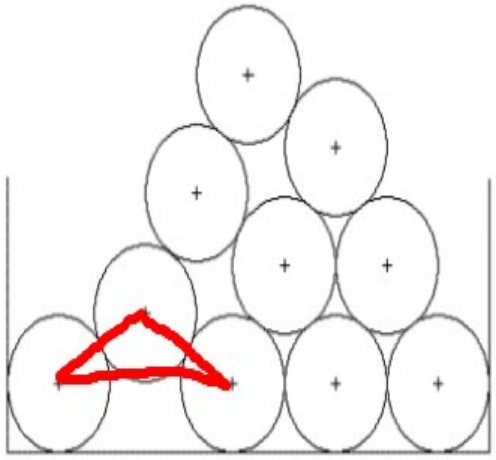
Soj1206 1012 Stacking Cylinders

- 横剖面如图示



Soj1206 1012 Stacking Cylinders

- 可以根据几何知识，得到放在当前层的某两个相邻圆柱体的，上一层圆柱的位置



Soj1206 1012 Stacking Cylinders

- 几何题可以使用stl的complex库来减少代码量

```
typedef complex<double> Point;
#define X real()
#define Y imag()

void test()
{
    Point A = Point(1, 2), B = Point(3, 4);
    double alpha = M_PI / 2.0;
    cout << A << endl;
    A + B;
    A - B; // get vector BA
    abs(A - B); // distance from A to B
    norm(A - B); // the square of distance from A to B
    A * exp(Point(0, alpha)); // rotate alpha
    A * Point(0, 1); // rotate 90 degree
    // ...
}
```

Soj1206 1012 Stacking Cylinders

- 代码：计算下一层的圆心位置

```
Point calcNextCircle(const Point &a, const Point &b)
{
    Point mid = (a + b) / Point(2, 0);
    double len = sqrt(4 - norm(a - mid));
    Point height = (b - a) * Point(0, 1);
    height = height / abs(height) * len;
    return mid + height;
}
```

Soj 1029 Rabbit

- 题意：
- 开始有一对成年兔子
- 每对成年兔子每个月产生一对小兔子
- 每只小兔子经过m个月变成成年兔子
- 问经过d个月后有多少兔子
- 约束：1 <= m <= 10, 1 <= d <= 100

Soj 1029 Rabbit

- 解法：递推
- 这是一道计数问题，对于这类题目，一般是分情况讨论。
- 比如说F[n]表示第n个月时兔子的数量
- 那么F[0] = 1
- 对于F[n]（n>0），可以分成两部分：
- 第一部分上一个月已经有的兔子F[n-1]
- 第二部分新产生的兔子，那么新产生兔子需要经历d个月的兔子，那么就是F[n-d]

Soj 1029 Rabbit

- 解法：递推
- 所以我们可以得到递推式：$F[n] = F[n-1] + F[n-d]$
- 另外注意的是对于$F[n](n<0)$，我们为了切合实际情况，需要另$F[n] = 1$

Soj 1029 Rabbit

- 解法：递推
- 注意到d=1时，F[n] = 2 * F[n – 1]，这个时候F[100] = 2 ^ 100，这时我们需要高精度
- 对于递推式，我们只需要实现高精度加法就可以了

Soj 1029 Rabbit

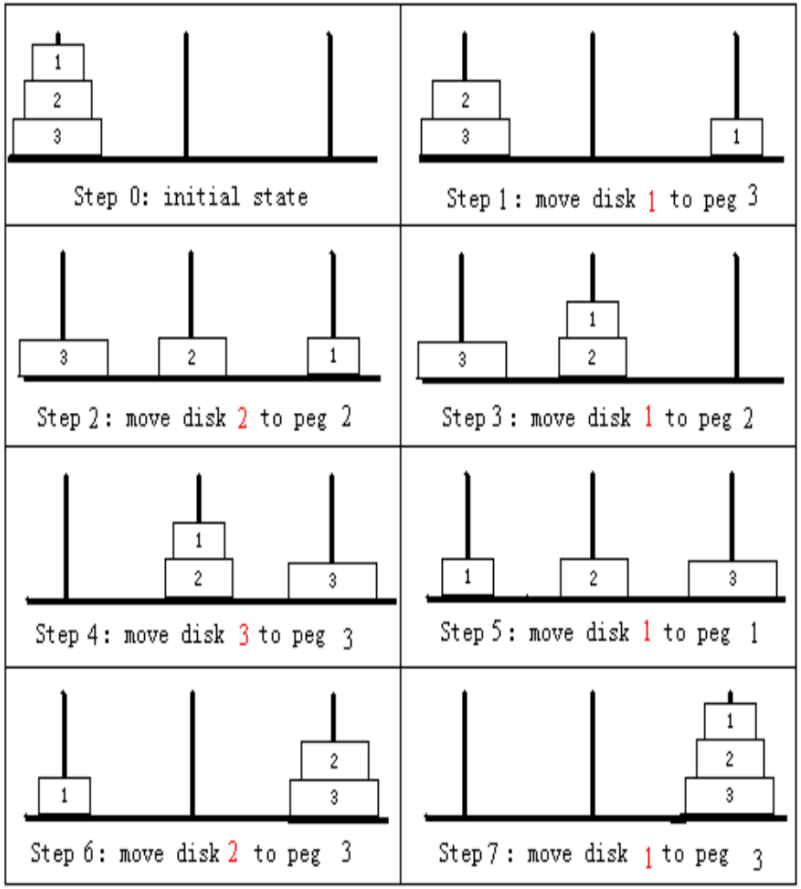
- 模拟竖式加法
- 代码中的高精度是
- 低位到高位的表示

```
struct BigInteger
{
    // from low to high
    int a[maxn], len;
    BigInteger(int val = 0)
    {
        memset(a, 0, sizeof(a));
        len = 1;
        a[0] = val;
    }
};
BigInteger operator + (const BigInteger &lhs, const BigInteger &rhs)
{
    BigInteger ret;
    int i, g;
    for (i = g = 0; i < lhs.len || i < rhs.len || g; i++)
    {
        if (i < lhs.len) g += lhs.a[i];
        if (i < rhs.len) g += rhs.a[i];
        ret.a[i] = g % 10;
        g /= 10;
    }
    ret.len = i;
    return ret;
}
```

Soj 1028 Hanoi Tower Sequence

- 题意：
- 汉诺塔问题，将从上到下的方向从小到大排列的盘子，从第一个柱子中移动到另一个柱子，其中可以借助第三个柱子，并且每次移动后每根柱子从上到下方向柱子大小总是从小到大的。
- 现在给定了一个移动规则，问第p个移动的盘子编号为多少
- 约束：1<=p<=10^100

Soj 1028 Hanoi Tower Sequence



Soj 1028 Hanoi Tower Sequence

- 方法1：
- 另外我们记得汉诺塔问题所需要的步数为
- F[n] = F[n-1] * 2 + 1
- F[1] = 1
- 可以得到通项公式为F[n] = 2 ^ n – 1
- 这启示着我们可以从二进制上考虑

Soj 1028 Hanoi Tower Sequence

- 移动序列：
- 1
- 1 2 1
- 1 2 1 3 1 2 1
- 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
- $F((0001)_2) = 1$ $F((0010)_2) = 2$ $F((0011)_2) = 1$ $F((0100)_2) = 3$…
- 观察发现，二进制中有从低位数起连续的0的数量加1就是当前移动的盘子的编号

Soj 1028 Hanoi Tower Sequence

- 方法2：
- 这种具有自相似的结构，被称为分形，是递归结构
- 许多类似的问题可以用下面的方法解决

Soj 1028 Hanoi Tower Sequence

- 对于这个问题重新定义：
- S[n]表示n个盘子的移动序列
- S[1] = “1”
- S[n] = S[n-1] + “n” + S[n-1]

- 现在要求S[oo]序列中第p个元素，oo表示无穷

Soj 1028 Hanoi Tower Sequence

- 代码：这里不考虑高精度的问题，仅作参考，这种解法更具一般性，可以解决更多的问题

```
int solve(int p)
{
    int n;
    for (n = 1; p > (1 << n) - 1;  n++);
    if (p == (1 << (n - 1)))
        return n;
    else
        return solve(p - (1 << (n - 1)));
}
```