

7.23 课堂笔记 24

课堂内容

1. P1003

(50 分) 二维数组模拟覆盖操作，受限于数组大小限制 5000*5000

```
#include <bits/stdc++.h>
using namespace std;
int ans[5005][5005];
int main()
{
    memset(ans, -1, sizeof(ans));
    int n;
    cin >> n;
    for(int i=1; i<=n; i++)
    {
        int a, b, x, y;
        cin >> a >> b >> x >> y;
        for(int j=a; j<=a+x; j++)
        {
            for(int k=b; k<=b+y; k++)
            {
                ans[j][k]=i;
            }
        }
    }
    int x, y;
    cin >> x >> y;
    cout << ans[x][y] << endl;

    return 0;
}
```

(100 分) 每次保存数组的边界，反向遍历，每次通过 $x \geq e[i].a$ && $x \leq e[i].a + e[i].g$ && $y \geq e[i].b$ && $y \leq e[i].b + e[i].k$ 判断给出的点是否在当前地毯之上，如果不在，再判断上一张地毯，留意如果遍历结束仍未找到地毯那么应该如题意输出 '-1' 表明没有地毯

```
#include <bits/stdc++.h>
using namespace std;
struct ditan{
    int a;
    int b;
    int g;
    int k;
} e[10010];
int n;
int x, y;

int main() {
    cin >> n;
    for(int i=1; i<=n; i++){
        cin >> e[i].a >> e[i].b >> e[i].g >> e[i].k;
    }
    cin >> x >> y;
    for(int i=n; i>=1; i--){
        if(x >= e[i].a && x <= e[i].a + e[i].g && y >= e[i].b && y <= e[i].b + e[i].k){
            cout << i;
            return 0;
        }
    }
    cout << -1;
    return 0;
}
```

2. P1313

(50 分)

$$(x+y)^1=x+y$$

$$(x+y)^2=x^2+2xy+y^2$$

$$(x+y)^3=x^3+3x^2y+3xy^2+y^3$$

$$(x+y)^4=x^4+4x^3y+6x^2y^2+4xy^3+y^4$$

$$(x+y)^5=x^5+5x^4y+10x^3y^2+10x^2y^3+5xy^4+y^5$$

可以观察出对于 a, b 都为 1 的数据，结果就是杨辉三角的第 $k+1, k-n+1$ 项
养成良好习惯每次都先模数再参与运算

```
#include <bits/stdc++.h>
using namespace std;
int ans[5005][5005];
int main()
{
    int a, b, k, n, m;
    cin >> a >> b >> k >> n >> m;
    ans[1][1] = 1;
    for (int i = 2; i <= k + 1; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            ans[i][j] = (ans[i - 1][j - 1] % 10007 + ans[i - 1][j] % 10007) % 10007;
        }
    }
    cout << ans[k + 1][k - n + 1] << endl;
    return 0;
}
```

(100 分)

对于杨辉三角的结果，再乘 (a^n) 再乘 (b^m)

```

#include<bits/stdc++.h>
using namespace std;
long long h[1010][1010];
long long a, b, k, n, m;
long long ans = 1;

int main() {
    cin >> a >> b >> k >> n >> m;
    h[1][1] = 1;
    k++;
    n++;
    for(int i=2; i<=k; i++) {
        for(int j=1; j<=n; j++) {
            h[i][j] = h[i-1][j-1] + h[i-1][j];
            h[i][j] %= 10007;
        }
    }
    for(int i=1; i<=n-1; i++) {
        a %= 10007;
        ans *= a;
        ans %= 10007;
    }
    for(int i=1; i<=m; i++) {
        b %= 10007;
        ans *= b;
        ans %= 10007;
    }
    ans *= h[k][n];
    ans %= 10007;
    cout << ans;
    return 0;
}

```

幂过程可以使用快速幂优化运行速度（本题不优化也能拿满分）

```

int quickPower(int a, int b)//是求a的b次方
{
    int ans = 1, base = a;//ans为答案，base为a^(2^n)
    while(b > 0)//b是一个变化的二进制数，如果还没有用完
    {
        if(b & 1)//&是位运算，b&1表示b在二进制下最后一位是不是1，如果是：
            ans *= base;//把ans乘上对应的a^(2^n)

        base *= base;//base自乘，由a^(2^n)变成a^(2^(n+1))
        b >>= 1;//位运算，b右移一位，如101变成10（把最右边的1移掉了），10010变成1001
    }
    return ans;
}

```

3. P1079

先考虑密钥的处理，把所有的大小写字母处理为在字母表中的顺序便于后续运算
处理密文的时候留意，对于所有处理后超出字母范畴的字母都需要加一个 26 来使其重新回到字母表的范畴

```

#include<iostream>
using namespace std;
string str1, str2;
int len1;
int a[1010];

int main() {
    cin >> str1 >> str2;
    for(int i=0; i<str1.length(); i++){
        if(str1[i] >= 'A' && str1[i] <= 'Z'){
            a[i] = str1[i] - 'A';
        }
        else{
            a[i] = str1[i] - 'a';
        }
    }
    len1 = str1.length();
    for(int i=0; i<str2.length(); i++){
        if(str2[i] >= 'A' && str2[i] <= 'Z' && str2[i] - a[i%len1] < 'A') {
            cout << char(str2[i] - a[i%len1]+26);
        }
        else if(str2[i] >= 'a' && str2[i] <= 'z' && str2[i] - a[i%len1] < 'a'){
            cout << char(str2[i] - a[i%len1]+26);
        }
        else{
            cout << char(str2[i] - a[i%len1]);
        }
    }
    return 0;
}

```

4. P1082

定义 $\gcd(a,b)$ 为 a,b 的最大公约数 (Greatest Common Divisor)。

由欧几里得定理得：

$$\forall a, b \in \mathbb{N}, b \neq 0, \quad \gcd(a, b) = \gcd(b, a \bmod b)$$

特别地： $\gcd(a, 0) = a$

由裴蜀定理得：

$$\forall a, b \in \mathbb{Z}, \exists x, y \in \mathbb{Z}, \quad a * x + b * y = \gcd(a, b)$$

扩展欧几里得算法：假设 $a*x+b*y=\gcd(a,b)$

1. $b=0$ 时， $\gcd(a,b)=a$, 解得 $x=1, y=0$

2. $b \neq 0$ 时，假设 $b * x' + (a \bmod b) * y' = \gcd(b, a \bmod b)$

$$\text{可等价变形为} \quad b * x' + \left(a - \left\lfloor \frac{a}{b} \right\rfloor * b\right) * y' = \gcd(b, a \bmod b) = \gcd(a, b)$$

$$a * y' + b * \left(x' - \left\lfloor \frac{a}{b} \right\rfloor * y'\right) = \gcd(a, b)$$

与方程 $a*x+b*y=\gcd(a,b)$ 比较得： $x=y', y=x'-(a/b)*y'$ ，不断重复操作，递归改变 x, y 即可求出一组特解 x_0, y_0 。

扩展：令 $d=\gcd(a,b)$ ，对于更一般的方程 $a*x+b*y=c$ ，当且仅当 $d|c$ 时有解。假设根据 $a*x+b*y=d$ 求出的一组特解为 x_0, y_0 ，将方程两边同乘 (c/d) ，就得到了原方程的一组特解 $(c/d) * x_0, (c/d) * y_0$ ，原方程的通解为：

$$x = \frac{c}{d}x_0 + k\frac{b}{d}, \quad y = \frac{c}{d}y_0 - k\frac{a}{d} \quad (k \in \mathbb{Z})$$

本题解法：题中所给的同余方程等价于 $a*x+b*y=1$ (y 为整数)，此题保证有解，因此 $\gcd(a,b)|1$ ，即 $\gcd(a,b)=1$ 。用扩展欧几里得算法求出特解 x_0, y_0 ， x 的通解为 $x=x_0+k*b$ (k 为整数)。注意题中要求最小正整数解。

```

#include<bits/stdc++.h>
using namespace std;
long long x, y;

void gcd(long long a, long long b)
{
    if(b == 0) {
        x = 1;
        y = 0;
        return;
    }
    gcd(b, a % b);
    long long tx = x;
    x = y;
    y = tx - a / b * y;
}

int main()
{
    long long a, b;
    cin >> a >> b;
    gcd(a, b);
    x = (x % b + b) % b;
    cout << x << endl;;
    return 0;
}

```

5. P1965

显然有答案公式 $(x+m*10^k) \% n$

因为 $\%$ 运算满足分配律所以有 $(x\%n+m\%n*10^k\%n)\%n$

再引入快速幂来计算 10^k 即可

```

//10^k次之后，x走到的位置是： (m * (10^k) + x) % n
#include <bits/stdc++.h>
using namespace std;
long long n, m, k, x;
long long ans;

//快速幂模板
long long quick_pow(long long a, long long p, long long mod)
{
    long long ans = 1;
    while(p)
    {
        if(p & 1) {
            ans = ans * a % mod;
        }
        a = a * a % mod;
        p >>= 1;
    }
    return ans;
}

int main()
{
    cin >> n >> m >> k >> x;
    ans = quick_pow(10, k, n)*m;
    ans = (x + ans) % n;
    cout << ans;
    return 0;
}

```

快速幂在很多题目都属于需要使用的基本知识

6. P1969

模拟操作即可 AC，较为简单

```
#include <bits/stdc++.h>
using namespace std;
int n, h[100050], ans;

int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> h[i];
    }
    for (int i = 1; i <= n; i++) {
        if (h[i] > h[i-1]) {
            ans += h[i] - h[i-1];
        }
    }
    cout << ans;
    return 0;
}
```

初赛讲解

S11

3:

8bit=1Byte

8*1024bit=1024Byte(字节)=1KB=1k

1024KB=1MB

1024MB=1GB

1024GB=1TB

b=bit 表示“二进制位。要么是 0，要么是 1。B=Byte 表示“字节”

5:

按位与 $a \& b$ 按位或 $a|b$

按位异或 a^b 按位取反 $\sim a$ 左移 $a \ll b$ 右移 $a \gg b$

(如果为负数的话，最左边的符号位要保留，正数为 0，负数为 1)

8:

不稳定排序：快速排序、选择排序、堆排序、希尔排序

10:

类别	排序方法	时间复杂度			空间复杂度	稳定性
		平均情况	最好情况	最坏情况	辅助存储	
插入排序	直接插入	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
	Shell排序	$O(n^{1.3})$	$O(n)$	$O(n^2)$	$O(1)$	不稳定
选择排序	直接选择	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
	堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(1)$	不稳定
交换排序	冒泡排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
	快速排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n^2)$	$O(n\log_2 n)$	不稳定
归并排序		$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n)$	稳定
基数排序		$O(d(r+n))$	$O(d(n+rd))$	$O(d(r+n))$	$O(rd+n)$	稳定

注：基数排序的复杂度中， r 代表关键字的基数， d 代表长度， n 代表关键字的个数。

S5:

1:

从左到右遍历表达式的每个数字和符号，遇到是数字就进栈，遇到是符号，就将处于栈顶两个数字出栈，进行运算，运算结果进栈，一直到最终获得结果。

13:

一个有 N 个结点的非空二叉树的高度至少为 $\lg N$

深度为 k 的完全二叉树至少有 $2^{(k-1)}$ 个结点； 最多： $2^k - 1$

具有 n 个结点的完全二叉树的深度为 $\lceil \log_2 n \rceil + 1$