1. 如果算了log，可以+-0.00000000001（按照不等号方向决定加还是减），这样可以补足系统四舍五入带来的问题。
2. ‘**\0’虽然不显示，但是会输出**！！！！多输出了会错！！

**3.（x>y)?x:y;**

4.推递归复杂度的方法：看一共算出了多少个状态。 a[][]

5.quick sort是快于merge sort的。

6.指针相关：int \*p=&a，表示p指向a的地址；

对p赋值相当于对a操作；

对\*p赋值相当于改变p指向的位置；

1. 对大数处理技巧：两个大数算平均，应该先除再加，防止溢出；

如果数很大，设long long int而不是int；

1. 左移<<:相当于乘2

右移>>:相当于除2

异或操作：^

异或的应用：对n位的二进制数，**异或上n个1**，可以得到它的complement。

n个1的计算方法：(1<<n)-1

【注意运算顺序：加法优先于易位】

1. 如果动态规划的函数算了两遍，并且处理的数组改变了。

那一定要注意存储增强的角标！如果还用原来的存储增强，就会出问题。

（例如：在算p的时候，dp[n]代表到第n个时候的值；之后再算p+1的时候，dp[n]因为已经算过了，会直接返回，但实际上它是1到n的值而不是你想要的2到n+1的值）

1. include “xxx.h”的意思是；把这个文件里的内容全部复制过来；

这里是相对路径。

1. Extern:用来声明其他文件里定义的 变量
2. **return** 0\*puts("-1")

可以用来直接讨论。

1. a/b想要**上取整**：（a+b-1/b；）
2. 适用于宽度有限的情况：要求是“步骤最少”

实现宽度优先：队列。每次先讨论最早放入队列的元素

1. (x\*y)mod m =[(x mod m)\*(y mod m)] mod m

大数求余数：可以先求余数，再乘（减少计算量）

1. 大部分cstring里面的函数都只对char[]管用，对string不可以用。
2. 字符‘0’是可以打印输出的，数字0不行；且注意‘’是字符，“”是字符串；当字符串输出到数字0时，自动停止输出；对一个字符串尾巴上自行加上一个数字0是防止诡异输出的好办法。
3. 与之对比，string是没有数字0的；string可以直接相加，就是简单地把string按照顺序接起来；且string只有字符串，只能用””来比较；

譬如 if (out[i].type != "0") //写成‘0’必凉

1. 对字符串需要使用strcpy与strcmp；但是字符本身可以直接赋值与比较；

(a[i + 1] <= 'z'&&a[i+1]>='a')

1. i++：先操作，再加加；++i,反之
2. 递归：一个状态->下一个状态+中间的代价；
3. log默认是ln；想用log10就写成log10；

double x = log10(a);同时注意这里对double的使用

22.快速幂的核心：指数减半，底数乘方。指数为奇数时，result乘上底数；

long long fastPower(long long base, long long power) {

long long result = 1;

while (power > 0) {

if (power & 1) {//此处等价于if(power%2==1)；按位与1相当于奇偶判断

result = result \* base % 1000;

}

power >>= 1;//此处等价于power=power/2

base = (base \* base) % 1000;

}

return result;

}

1. 位操作：
2. a的b次方的头三位：

int qian(long long int a, long long int b) {

{

double x = log10(a);

double y = b \* x;

long long int m = int(y);//这一步很重要，完成整型转换

if (m <= 2)

{

return(pow(10, y) + 0.000001);//给power加上小量是个好习惯

}

double z = y - m;

int arc = pow(10, z + 2) + 0.000001;

return arc;

}

}

24.函数传参数：

A.传入二维数组：int mat[7][7];int det(int jie, int yu[][7]);//注意到后面的二维数组传入的时候必须要第二个空格有数字，且在递归的时候传入的数组也要第二位相同

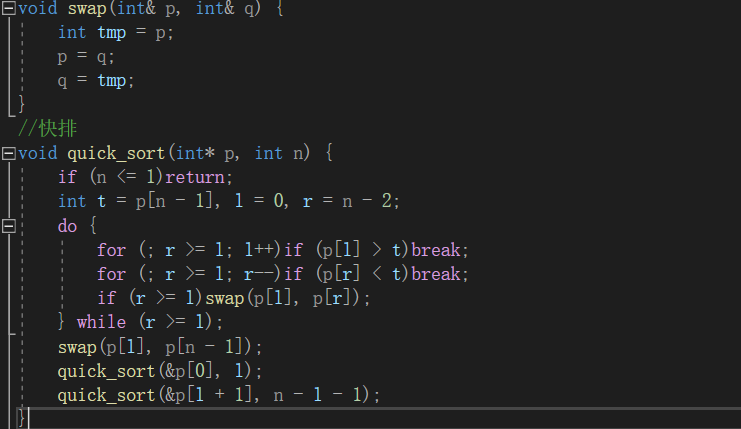
B.取引用：void moneyjudge(int &x, int a[], int y){while (x >= y) a[y]++, x -= y;}//注意到不写&x而仅仅写为x会导致无法改变x的值；需要通过函数改变全局变量，都要取引用。

C.传数组：写定义时，在参数表里 long long a[] 等价于 long long\* a，表示要传入一个long long 型指针，它在函数体里被当作数组的首指针；调用时，仅仅需要写a，也就是a数组的首指针；

D.传引用和传地址的关系：

void swap(int& p, int& q) {int tmp = p; p = q; q = tmp;}与void swap(int\* p, int\* q) { int tmp = \*p; \*p = \*q; \*q = tmp;}效果一样，但是原理不同：第一个是传引用，第二个传的是指针（可以理解成只有一个元素的数组）；第二个写法在函数里面都要写\*p，第一种写法直接写p；

传数组时只能传指针进去，且可以用p[n-1]而前面不写\*；



bool judge(long long x,long long &b)

{

for (int i = 0; i < n; i++)

{

if (cqq + a[i] <= x)cqq += a[i];

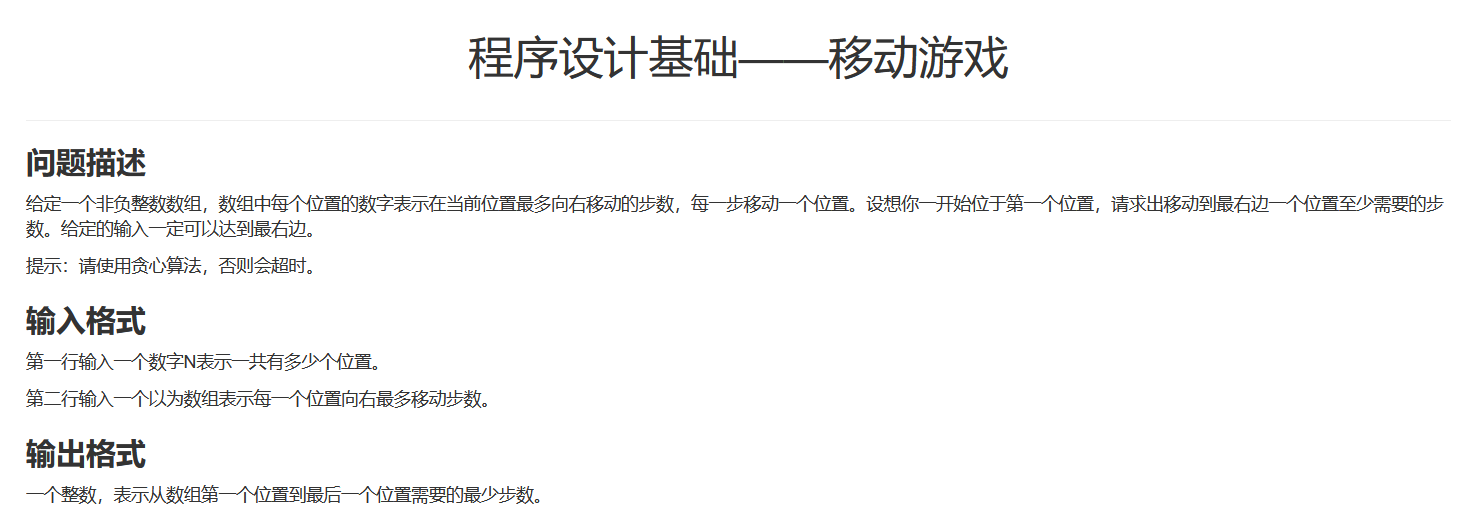
else cqq = a[i], zcy++;

}

return (zcy > m);

调用：if (judge(mid, a))lef = mid + 1;

25.边界情况与隐藏的边界



如果n为1的话，直接cout<<0;

26.动态分配：

int N=0;cin>>N;student \*stu=new student [N];类型，取地址，名字=类型，[数目]。然后正常使用stu[0]到stu[N-1]; 结尾：delete []stu;

1. %是取余数，/是按照类型做除法；
2. 全局变量字符串不初始化是默认是'\0'（空字符），不是'0'（字符0）
3. 在某种类型的dp中，如果只要确定了最终的选择里有某个数，就可以得到结果，那么可以考虑把所有的可能性都算出来，然后排一下序！
4. 先排序再计算往往可以简化计算过程（如，求交集并集，可以先排序在做。但是要注意可能需要保存原序号）
5. 可以按照某个单位把所有cin的数据压缩成一个：（例如：每个人、每个商品）；且这个压缩后的数据可以反应所有原有信息。