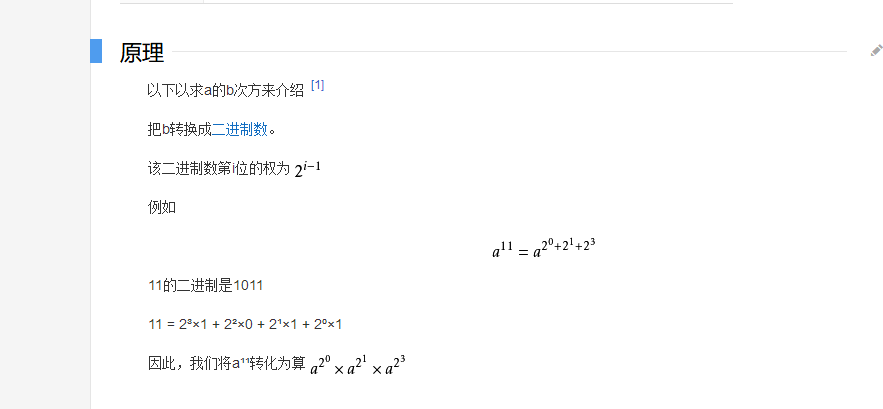
给定一个double类型的浮点数base和int类型的整数exponent。求base的exponent次方。

理解了很久，大概有3种做法，第一种最low的，直接一个for暴力乘，这里不给出代码了。

第二种：快速幂运算。



下一位可以看成前一位的平方。

public class Solution {

public double Power(double base, int exponent) {

int a = exponent;

if (exponent == 0) {

return 1;

}

if (exponent < 0) {

if (base == 0) {

throw new RuntimeException("分母不为0");

}

exponent = -exponent;

}

double result = 1;

double res = base;

while (exponent != 0) {

if ((exponent&1) == 1) {

result = res \* result;

}

res = res\*res;**// 后面的每一项看成前面每一项的平方，先算幂运算，再相乘**

exponent = exponent>>1;

}

return (a > 0 ? result : (1 / result));

}

}

递归的话呢：

public class Solution {

public double Power(double base, int exponent) {

int n = Math.abs(exponent);

if (exponent == 0) {

return 1;

}

if (exponent == 1) {

return base;

}

double result = Power(base, n >> 1);

result = result \* result;//其实这儿和上面的思想差不多，但是其实感觉上面懂了，这儿用递归反而更麻烦，所以位运算其实感觉应该是最优的算法了。

if ((n & 1) == 1) {

result = result \* base;

}

if (exponent < 0) {

result = 1/result;

}

return result;

}

}