Java中float的取值范围 PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/138/2021_2022_Java_E4_B8_ADflo_c97_138650.htm 规格化表示 java中的浮点数采用的事IEEE Standard 754 Floating Point Numbers标准,该标准的规范可以参考http://blog.csdn.net/treeroot/articles/94752.aspx. float占用4个字节,和int是一样,也就是32bit. 第1个bit表示符号,0表示正数,1表示负数,这个很好理解,不用多管. 第2-9个bit表示指数,一共8为(可以表示0-255),这里的底数是2,为了同时表示正数和负数,这里要减去127的偏移量.这样的话范围就是(-127到128),另外全0和全1作为特殊处理,所以直接表示-126到127. 剩下的23位表示小数部分,这里23位表示了24位的数字,因为有一个默认的前导1(只有二进制才有这个特性). 最后结果是:(-1)^(sign) * 1.f * 2^(exponent) 这里:sign是符号位,f是23bit的小数部分,exponent是指数部分,最后表示范围是(因为正负数是对称的,这里只关心正数) 2^(-126) ~~ 2(1-2^(-24)) * 2^127 这个还不是float的取值范围,因为标准中还规定了非规格化表示法,另外还有一些特殊规定. 非规格化表示: 当指数部分全0而且小数部分不全0时表示的是非规格化的浮点数,因为这里默认没有前导1,而是0. 取值位0.f * 2^(-126),表示范围位 2^(-149)~~ (1-2^(-23)) * 2^(-126) 这里没有考虑符号.这里为什么是-126而不是-127? 如果是-127的话,那么最大表示为 2^(-127)-2^(-149),很显然2^(-127) ~~2^(-126) 就没法表示了. 其他特殊表示 1.当指数部分和小数部分全为0时,表示0值,有 0和-0之分(符号位决定),0x00000000表示正0,0x80000000表示负0. 2.指数部分全1,小数部分全0时,表示无穷大,有正无穷和负无穷,0x7f800000表示正

无穷,0xff800000表示负无穷. 3.指数部分全1,小数部分不全0时,表示NaN,分为QNaN和SNaN,Java中都是NaN. 结论: 可以看出浮点数的取值范围是:2^(-149)~~(2-2^(-23))*2^127,也就是Float.MIN_VALUE和Float.MAX_VALUE. 