1. Valid Number

Validate if a given string can be interpreted as a decimal number.

Some examples: "0" => true " 0.1 " => true "abc" => false "1 a" => false "2e10" => true " -90e3 " => true " 1e" => false "e3" => false " 6e-1" => true " 99e2.5 " => false "53.5e93" => true " --6 " => false "-+3" => false "95a54e53" => false

**Note:** It is intended for the problem statement to be ambiguous. You should gather all requirements up front before implementing one. However, here is a list of characters that can be in a valid decimal number:

* Numbers 0-9
* Exponent - “e”
* Positive/negative sign - “+”/“-”
* Decimal point - “.”

Of course, the context of these characters also matters in the input.

**解**

三种情况：

* 整数
* 浮点数
* 指数形式

指数形式中，基数可以为浮点数，指数必须是整数

分类判断，最基本的是判断整数和浮点数

==问题分化==

class Solution {  
public:  
 bool isNumber(string s) {  
 while(s.back() == ' ' && s.size() > 1)s.erase(s.end() - 1);  
 while(s[0] == ' ' && s.size() > 1)s.erase(s.begin());  
 if(isInt(s) || isFloat(s))return true;  
 int posE = s.find('e');  
 if(posE == string::npos)return false;  
 string s1 = s.substr(0, posE);  
 string s2 = s.substr(posE + 1);  
 if((isInt(s1) || isFloat(s1)) && isInt(s2))return true;  
 return false;  
 }  
 bool isInt(const string &s){  
 if(s.size() == 0)return false;  
 int pos = 0;  
 if(s[0] == '-' || s[0] == '+')pos++;  
 if(pos == s.size())return false;  
 while(pos < s.size()){  
 if(s[pos] < '0' || s[pos] > '9')return false;  
 pos++;  
 }  
 return true;  
 }  
 bool isFloat(const string &s){  
 if(s.size() == 0 || s == ".")return false;  
 int pos = 0;  
 if(s[0] == '-'|| s[0] == '+')pos++;  
 if(pos == s.size())return false;  
 if(s.substr(pos) == ".")return false;  
 bool flag = false;  
 while(pos < s.size()){  
 if(s[pos] == '.'){  
 if(flag)return false;  
 flag = true;  
 }else if(s[pos] < '0' || s[pos] > '9'){  
 return false;  
 }  
 pos++;  
 }  
 return true;  
 }  
};