等式(2.13)也让我们认出了哪些情况下会发生溢出：

原理：检测补码加法中的溢出

对满足 $TMin_w \leqslant x$，$y \leqslant TMax_w$ 的 x 和 y，令 $s \doteq x +_w^t y$。当且仅当 $x>0$，$y>0$，但 $s \leqslant 0$ 时，计算 s 发生了正溢出。当且仅当 $x<0$，$y<0$，但 $s \geqslant 0$ 时，计算 s 发生了负溢出。

图 2-25 显示了当 $w=4$ 时，这个原理的例子。第一个条目是负溢出的情况，两个负数相加得到一个正数。最后一个条目是正溢出的情况，两个正数相加得到一个负数。

推导：检测补码加法中的溢出

让我们先来分析正溢出。如果 $x>0$，$y>0$，而 $s \leqslant 0$，那么显然发生了正溢出。反过来，正溢出的条件为：1)$x>0$，$y>0$(或者 $x+y<TMax_w$)，2)$s \leqslant 0$(见公式(2.13))。同样的讨论也适用于负溢出情况。■

练习题 2.29　按照图 2-25 的形式填写下表。分别列出 5 位参数的整数值、整数和与补码和的数值、补码和的位级表示，以及属于等式(2.13)推导中的哪种情况。

x	y	$x+y$	$x +_5^t y$	情况
[10100]	[10001]			
[11000]	[11000]			
[10111]	[01000]			
[00010]	[00101]			
[01100]	[00100]			

练习题 2.30　写出一个具有如下原型的函数：

```
/* Determine whether arguments can be added without overflow */
int tadd_ok(int x, int y);
```

如果参数 x 和 y 相加不会产生溢出，这个函数就返回 1。

练习题 2.31　你的同事对你补码加法溢出条件的分析有些不耐烦了，他给出了一个函数 tadd_ok 的实现，如下所示：

```
/* Determine whether arguments can be added without overflow */
/* WARNING: This code is buggy. */
int tadd_ok(int x, int y) {
    int sum = x+y;
    return (sum-x == y) && (sum-y == x);
}
```

你看了代码以后笑了。解释一下为什么。

练习题 2.32　你现在有个任务，编写函数 tsub_ok 的代码，函数的参数是 x 和 y，如果计算 x-y 不产生溢出，函数就返回 1。假设你写的练习题 2.30 的代码如下所示：

```
/* Determine whether arguments can be subtracted without overflow */
/* WARNING: This code is buggy. */
int tsub_ok(int x, int y) {

    return tadd_ok(x, -y);
}
```

x 和 y 取什么值时，这个函数会产生错误的结果？写一个该函数的正确版本(家庭作业 2.74)。