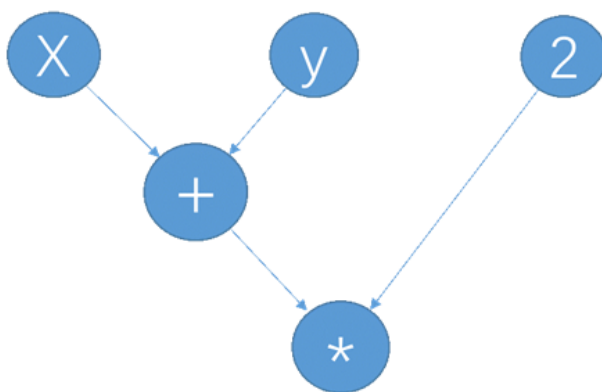


大作业——计算图

一.什么是计算图

计算图是近年来深度学习必需的计算工具，著名的项目有TensorFlow、Theano、PyTorch等。本项目要求大家实现静态计算图，它的本质即是用图来表示数据的处理过程。用计算图的好处在于：它可以保存数据的依赖关系和结构，在重复同样的计算时，有机会去优化和并行；并且也可以实现参数的自动求导。

举一个简单的例子，我们定义 x, y 为placeholder（占位符，即计算图的输入），还有一个常量2。那么通过以下方式我们可以定义一个计算图： $ans = (x + y) \cdot 2$



接下来，我们可以对 x, y 输入不同的值进行反复计算。

$ans.eval(x = 1, y = 2) \rightarrow 6$

$ans.eval(x = 1, y = -1) \rightarrow 0$

因此，即使我们不知道 x, y 的具体数值，我们也可以定义我们所需的计算图。定义完成之后，我们可以通过输入 x, y 得到计算图中其他节点的数值。这就是计算图的使用方式。

二.第一阶段需求

你需要实现一个C++的库，支持计算图的构建与计算。**为了便于测试结果的比较，所有的计算结果保留 4 位小数：**

1. 实现float标量的Placeholder和Constant，并重载加减乘除四则运算（占10%）

o 输入样例：

第一行是变量数 n ，接下来 n 行是对变量的具体描述，每一行由变量名和变量属性组成。属性有两种：P 表示该变量是 Placeholder；C 表示该变量是 Constant，接下来的数字即为该变量的数值。

然后是节点数 m ，接下来 m 行每一行定义一个新节点，每行包含一个等号，等号左边是节点名称，等号右边是得到新节点的计算过程。这些节点共同构成了一张计算图，通过输入不同的 x, y ，可以反复计算图中的其他节点 a, b, c, res 。

然后是计算操作数 q ，接下来 q 行每一行是一个计算操作。每一行第一个元素是待计算结果的节点名称，第二个元素 k 表示接下来有 k 个类型为 Placeholder 的变量需要赋值，然后是 k 个赋值对，每个赋值对包括一个待赋值的 Placeholder 类型的变量和赋值的数值，每一行的计算操作根据给出的赋值输出一个计算结果。

输入输出只是为了检测你的程序是否正确，你并不需要为输入输出语句设计对象。

```

3
x P
y P
z C 3.0
4
a = x + y
b = a - z
c = b * a
res = z / c
5
EVAL x 1 x 1.0
EVAL a 2 x 1.0 y 2.0
EVAL b 2 x 1.0 y 0.0
EVAL c 1 y 1.0
EVAL res 2 x -1.0 y 1.0

```

- 输出样例:

依次输出 q 个操作的计算结果。若待计算的节点依赖的节点未赋值，或除法操作除数为0，则输出相应错误信息。

```

1.0000
3.0000
-2.0000
ERROR: Placeholder missing
ERROR: Division by zero

```

说明:

- 输入 $x = 1.0$, 计算 $x = x = 1.0$
- 输入 $x = 1.0, y = 2.0$, 计算 $a = x + y = 3.0$
- 输入 $x = 1.0, y = 0.0$, 计算 $b = x + y - z = 1.0 + 0.0 - 3.0 = -2.0$
- 输入 $y = 1.0$, 计算 $c = (x + y - z)(x + y)$, c 的计算依赖于 x , 但未输入 x , 则输出错误信息
- 输入 $x = -1.0, y = 1.0$, 计算 $res = \frac{z}{(x+y-z)(x+y)}$, 但 $x + y = 0$, 则输出 ERROR。对于除法, 在定义计算图时由于不知道除数是否为0, 因此不会报错, 但是在计算节点具体数值的过程中如果遇到除数为0, 则需要报错

2. 实现float标量的sin (正弦运算)、log (对数运算)、exp (指数运算)、tanh (双曲正切运算)、sigmoid运算。在log函数的计算过程中, 如果参数不在定义域内则给出错误警告。 (占3%)

- 输入样例:

增加 SIN、LOG、EXP、TANH、SIGMOID五个单目运算符, 其中 $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$, LOG 操作底数为自然指数 e

```

3
x P
y P
z C 3.0
7
a = SIN z
b = LOG y

```

```

c = EXP x
d = SIGMOID c
e = TANH d
t = a + b
res = t * e
4
EVAL res 2 x 1.0 y 2.0
EVAL res 2 x 1.0 y -1.0
EVAL c 2 x 1.0 y 2.0
EVAL d 2 x 1.0 y 1.0

```

o 输出样例:

```

0.6126
Error: LOG operator's input must be positive
2.7183
0.9381

```

说明:

输入 $x = 1, y = 2$

$a = \sin(z) = 0.1411$

$b = \log(y) = 0.6931$

$c = \exp(x) = 2.7183$

$d = \frac{1}{1+e^{-c}} = 0.9381$

$e = \frac{e^d - e^{-d}}{e^d + e^{-d}} = 0.7343$

$t = a + b = 0.8343$

$res = t * e = 0.6126$

3. 实现用于调试的Print运算。当计算图构建完成后，eval的过程对用户是不可见的，这时候调试往往会产生困难。你需要实现Print运算符，这个运算符输入为1个变量，输出仍为这个变量本身，但是能够打印出这个变量的值。 **(占3%)**

o 输入样例

在定义节点时增加单目运算符 Print，其他输入说明不变

```

3
x P
y P
z C 3.0
5
a = x + y
b = PRINT a
c = b * z
d = PRINT c
res = b * d
5
EVAL x 1 x 1.0
EVAL a 2 x 1.0 y 2.0

```

```
EVAL b 2 x 1.0 y 0.0
EVAL c 1 x 2.0 y 1.0
EVAL res 2 x -1.0 y 1.0
```

◦ 输出样例

依次输出 q 个操作的计算结果：

```
1.0000
3.0000
Print Operator: a=1.0
1.0000
Print Operator: a=3.0
9.0000
Print Operator: a=0.0
Print Operator: c=0.0
0.0000
```

4. 为了保持计算图的高效性，即使一个结点被多次引用，也只应该计算一次。这点在使用了Print的情况下会更加明显 **(占3%)**

◦ 输入样例

```
2
x P
z 3.0
4
t = Print x
a = t + z
b = t + a
c = t + b
1
EVAL c 1 x 1.0
```

◦ 输出样例

依次输出 q 个操作的计算结果：

```
Print Operator: c=1.0
6.0000
```

5. 实现变量Variable。当计算图较大，输入过多时，eval的参数会变得很多。但我们经常希望能够保持一些输入变量的值。使用SET指令可以修改Variable的值 **(占5%)**

◦ 输入样例

在输入变量时，增加变量类型 V ，表示该变量是 Variable 类型，接下来的数字表示该变量的初始数值。

在输入计算操作时增加2个操作：

- SETCONSTANT v n ：将变量 v 的值更改为 n ；
- SETANSWER v i ：将变量 v 的值更改为第 i 个操作的输出，保证第 i 个操作在该操作之前，且一定有输出；

这些操作没有输出。

```

2
x P
y V 1.0
1
res = x + y
5
EVAL res 1 x 1.0
SETCONSTANT y 2.0
EVAL res 1 x 2.0
SETANSWER y 3
EVAL res 1 x 1.0

```

◦ 输出样例

依次输出 q 个操作的计算结果：

```

2.0000
4.0000
5.0000

```

◦ 样例解释 SETANSWER y 3 将第3个操作的输出4.0000赋给了x

6. 实现float标量的 $>$ 、 $<$ 、 $>=$ 、 $<=$ 、 $==$ 等比较运算，该运算输入两个变量，判断两个变量之间的关系。若满足条件，则输出为浮点数 1.0，否则输出为浮点数 0.0。 **(占3%)**

◦ 输入样例：

```

3
x P
y P
z C 3.0
7
a = x < y
b = a <= y
c = b > a
d = c >= b
e = d == c
t = a + e
res = t * z
1
EVAL res 2 x 1.0 y 2.0

```

◦ 输出样例：

```

6.0000

```

说明：

输入 $x = 1, y = 2$

$a = (x < y) = 1$

$b = (a \leq y) = 1$

$$c = (b > a) = 0$$

$$d = (c \geq b) = 0$$

$$e = (d == c) = 1$$

$$t = a + e = 2$$

$$res = t * z = 6$$

7. 在计算图中增加条件判断命令 COND，接受三个参数，第一个参数是一个比较运算式，第二个参数是如果比较运算式为真时的输出，第三个参数是如果比较运算式为假时的输出。 **(占3%)**

- 输入样例：

由于本次大作业不要求实现布尔类型的变量，因此 COND 的第一个参数可以是任意浮点数，当它大于 0 时返回第二个参数，当它小于等于 0 时假设返回第三个参数：

```
3
x P
y P
z C 3.0
5
a = x + y
b = a - z
c = x > y
t = COND c a b
res = COND b a c
2
EVAL t 2 x 1.0 y -1.0
EVAL res 2 x 1.0 y -1.0
```

- 输出样例：

```
0.0000
1.0000
```

输入 $x = 1.0, y = -1.0$,

$a = x + y = 0.0$

$b = a - z = -3.0$

$c = x > y = 1.0$

则 $t = a = 0.0, res = c = 1.0$

8. 你可以添加其他测试程序，说明你编写的库的使用方式和在极端情况下的运行情况。若涉及到前面要求未覆盖并且合理的部分，可以加分。 **(bonus, 不超过5%)**

三.项目限制

只能使用C++完成，禁止使用第三方库（不包括STL或编译器自带的库）。

四.提交要求

- 提交你编写的计算图库，请遵守OOP的设计规范。
- 给出测试代码，至少包含上面所述的所有示例输入和输出，以及使其能够编译的Makefile。

3. 在readme.md或readme.txt中写明程序的运行环境。（你的库最好能跨平台运行，并且依赖项尽量少，否则别的组将很难运行你的代码。）
4. 给出一个说明文档（markdown或word），写清计算图库的结构、封装、接口。给出开发者指导，便于别的小组能够阅读你的代码。（字数不限，目标是能让别人在最短时间内明白项目结构，太短或太长都不太好。）
5. 善用注释，便于别的小组能够阅读你的代码。
6. 在根目录提交一个developer.txt，写明所有组员的学号姓名。除此以外，项目的任何地方（包括代码、注释、文档）禁止出现任何个人信息。
7. 组与组之间可以讨论，但禁止以任何形式交换代码。在第二阶段互评的过程中，我们会随机发放代码，禁止互相讨论每组手上有哪些代码或者打听分数。最终投票结果我们会在第二阶段结束后给出。这一项涉及到最终得分的公平性，请大家自觉维护。