# 201700949 설재혁

2.

In [26]:

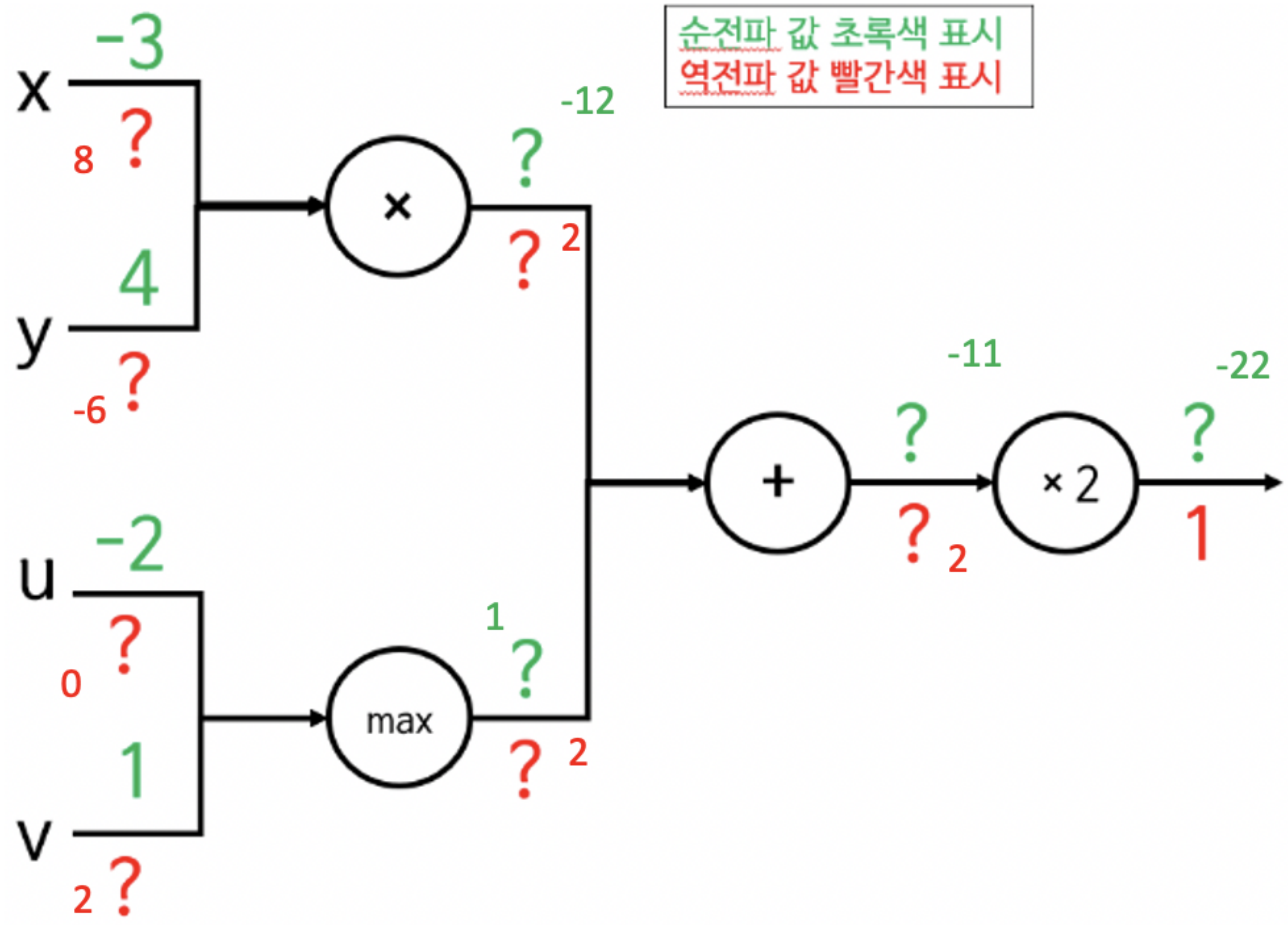
```
x = 2; z = 2
y = x**3 + x

#(1)
dfdx = 6*x*z + 2*(x**3 + x)

#(2) 미분 결과 값 : 44

#(3)

f = 2*x*y + 3*x**2*z + 4*x

dfdx = 6*x*z + 2*(x**3 + x)
print('결과:',dfdx)
```

```
결과: 44
```

3.

f(x,y,u,v) = (x*y) + max(u,v)

(1)



In [27]:

```
x = -3; y = 4; u = -2; v = 1

# forward
a = x * y
b = max(u,v)
c = a + b
f = 2 * c

# Backward
dfdc = 2
dfda = 2 # dfda = 2(a+b)
dfdb = 2 # dfdb = 2(a+b)
dfdx = 2.0 * y
dfdy = 2.0 * x
if(u>v):
    dfdu = dfdb
    dfdv = 0
else:
    dfdv = dfdb
    dfdu = 0

print('Forward pass: ',a,b,c,f)
print('Backward pass: ',dfdc,dfda,dfdb,dfdx,dfdy,dfdu,dfdv)
```

```
Forward pass:  -12 1 -11 -22
Backward pass:  2 2 2 8.0 -6.0 0 2
```

4.

In [28]:

```
import numpy as np

# (1)
def relu(x):
    return 2*(x>0)*x

input_data = np.array([1.0, 0.5])
W_input_output = np.array([[0.9, 0.2], [0.3, 0.8]])


X_output = np.dot(input_data, W_input_output)

actuals = relu(X_output) # 실제 값
print('(1): ',actuals)

# (2)
targets = [1.85, 0.1] # 목표값

E_output = targets - actuals # 오차
W_input_output_sum = W_input_output.sum(axis=1, dtype="float") # 입력 계층과 출력 계층 간
W_input_output_norm = W_input_output.T / W_input_output_sum # 정규화

E_input = np.dot(W_input_output_norm, E_output) # 입력 계층 오차
print('(2):' ,E_input)

# (3)
alpha = 0.1 # 학습률
sum_of_weight = np.dot(W_input_output, X_output)
result = -(E_output * relu(sum_of_weight) * (1 - relu(sum_of_weight)) * X_output) #
variance = alpha * result # 변화량 == 학습률 * 오차 기울기
updatedWeight = W_input_output - variance # 업데이트 된 가중치
print('(3):', updatedWeight)
```

```
(1):  [2.1 1.2]
(2): [-0.50454545 -0.84545455]
(3): [[0.96318113 0.2619146 ]
 [0.36318113 0.8619146 ]]
```

In [ ]: