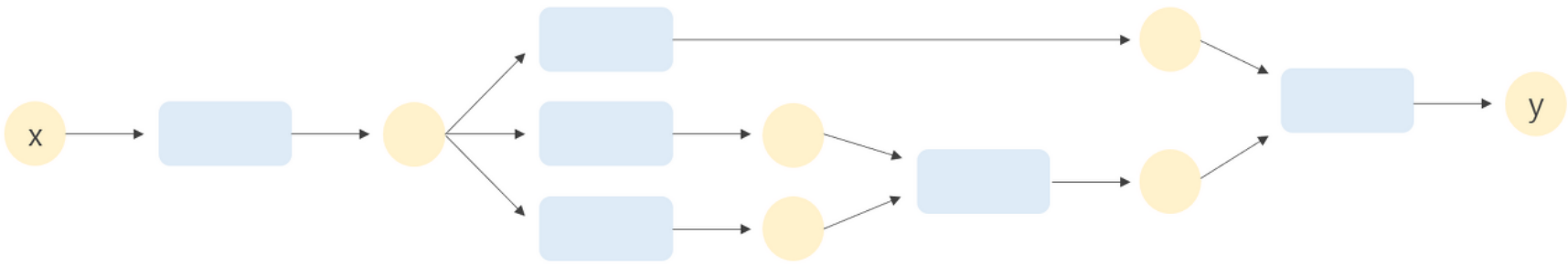
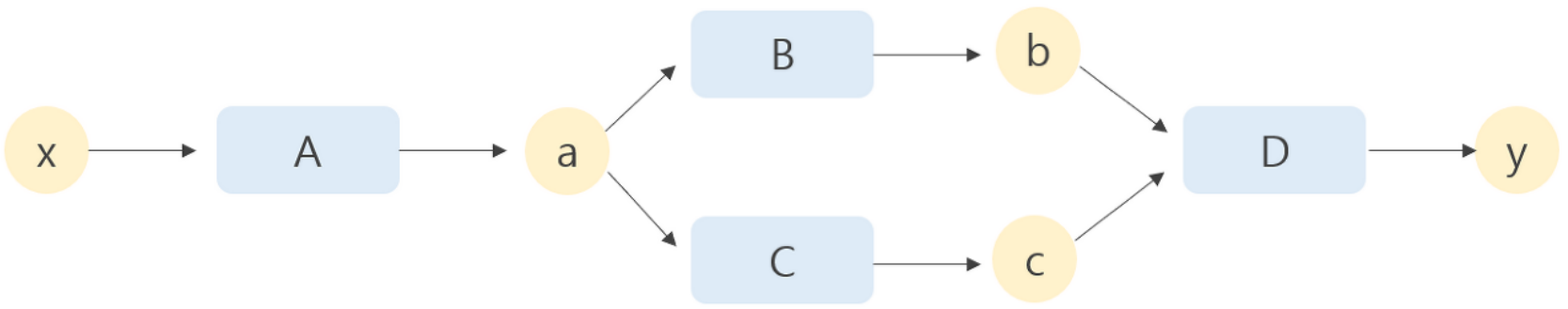
제 2고지 16.복잡한 계산 그래프의 이론 및 구현



- 일차적이지 않은 계산 그래프에 대한 계산 과정



- 아래 코드의 funcs.append(x.creator)로 처리할 함수의 후보를 funcs에 추가하고 pop으로 다음 처리할 함수를 꺼낸다

```
class Variable:

  #생략

    def backward(self):
      if self.grad is None:
          self.grad = np.ones_like(self.data)

        funcs = [self.creator]
        while funcs:
          f = funcs.pop()  ##### 이 부분 #####
            gys = [output.grad for output in f.outputs]
            gxs = f.backward(*gys)
            if not isinstance(gxs, tuple):
              gxs = (gxs,)

            for x, gx in zip(f.inputs, gxs):
              if x.grad is None:
                  x.grad = gx
                else:
                    x.grad += gx

                if x.creator is not None:
                  funcs.append(x.creator)  ##### 이 부분 #####
```

- funcs 리스트에 저장되는 순서에 맞게 적절한 차례로 함수를 꺼내야 함
  - 적절한 차례 → 함수에 우선순위를 부여
  - 우선 순위를 주기 위해, 순전파 계산의 함수와 변수가 만들어 지는 과정을 이용, creator, 부모-자식 등의 관계를 기준으로 **세대(generation)**을 기록
    - 세대를 기록하여 세대 수가 큰 것부터 처리할 수 있도록
- 세대 설정

```
class Variable:
  def __init__(self, data):
      if data is not None:
          if not isinstance(data, np.array):
              raise TypeError('{}은(는) 지원하지 않습니다.'.format(type(data)))

        self.data = data
        self.grad = None
        self.crator = None
        self.generation = 0 #세대수 변수

    def set_creator(self, func):
      self.creator = func
        self.generation = func.generation + 1 #세대수 기록 (부모 세대 +1)
```

```
    #생략
```

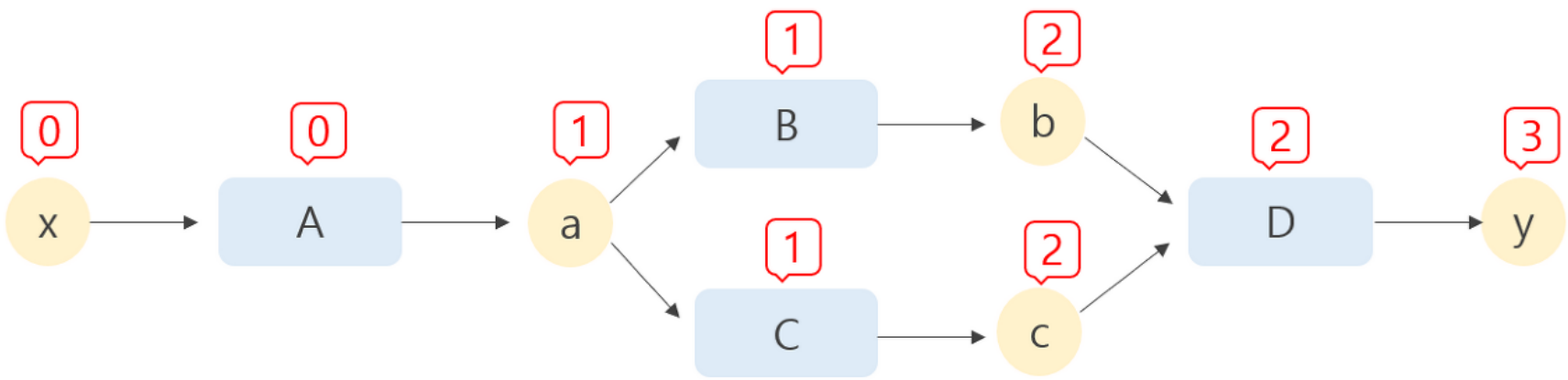
- set_creator메서드가 호출될 때 부모 함수의 세대보다 1 큰 값으로 세대수를 설정

```
class Function(object):
  def __call__(self, *inputs):
      xs = [x.data for x in inputs]
        ys = self.forward(*xs)
        if not isinstance(ys, tuple):
          ys = (ys,)
        outputs = [Variable(as_array(y)) for y in ys]

        self.generation = max([x.generation for x in inputs]) #더 큰 세대수를 받는다
        for output in outputs:
          output.set_creator(self)
        self.inputs = inputs
        self.outputs = outputs
        return outputs if len(outputs) > 1 else outputs[0]
```

- 함수의 generation은 입력 변수의 generation과 같은 값으로 설정
- 입력변수가 여러개라면 더 큰 generation을 가질 수 있도록

세대 순으로 pop




```
generations = [2, 0, 1, 4, 2]
funcs = []

for g in generations:
  f = Function()
    f.generation = g
    funcs.append(f)

[f.generation for f in funcs] #[2, 0, 1, 4, 2]
```

```
funcs.sort(key=lambda x: x.generation)
[f.generation for f in funcs] #[0, 1, 2, 2, 4]

f = funcs.pop()
f.generation #4
```