[보충] 파이토치 아키텍처

- **파이토치의 아키텍처**
  - 오프셋과 스트라이드라 이해
  - 전치 행렬 이해
  - A 행렬에서 첫 번째 열을 첫 번째 행으로 위치시키고, 두 번째 열을 두 번째 행으로 위치시키며, A^T로 표현(동일한 원리를 텐서에 적용할 수 있음)

$$A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix} \Rightarrow A^T=\begin{bmatrix}1 & 4 & 7\\ 2 & 5 & 8\\ 3 & 6 & 9\end{bmatrix}$$

파이토치>텐서 튜토리얼
https://pytorch.org/docs/stable/storage.html

$$A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$$ 1차원 텐서로 변환 → | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

$$A^T=\begin{bmatrix}1 & 4\\ 2 & 5\\ 3 & 6\end{bmatrix}$$ 1차원 텐서로 변환 → | 0 | 1 | 4 | 2 | 5 | 3 | 6 |

2.1 파이토치 개요

- **파이토치의 아키텍처**
  - 오프셋, 스트라이드
  - 행과 열의 수가 다른 텐서 A와 A^T를 생성해 보자
  - A와 A^T를 1차원 배열로 바꾸어서 메모리에 저장시키기 위해 텐서의 값들을 연속적으로 배치해 보자

▼ **3차원 텐서를 1차원으로 변환**

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

1차원 텐서로 변환 →

0	1	2	3	4	5	6

$$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

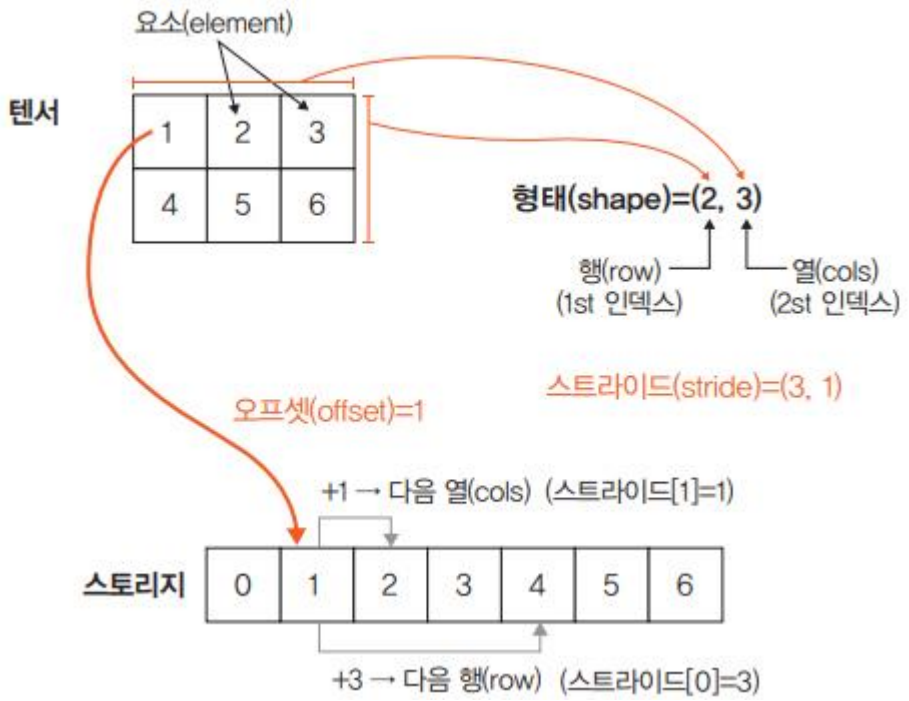
1차원 텐서로 변환 →

0	1	4	2	5	3	6

- A(2×3)와 A^T(3×2)는 다른 형태(shape)를 갖지만 스토리지의 값들은 서로 같음
- A와 A^T를 구분하는 용도로 오프셋과 스트라이드를 사용
- 다음 그림의 스토리지에서 2를 얻기 위해서는 1에서 1칸을 뛰어넘어야 하고, 4를 얻기 위해서는 3을 뛰어넘어야 함
- 텐서에 대한 스토리지의 스트라이드는 (3, 1)

2.1 파이토치 개요

▼ **1차원으로 변환**

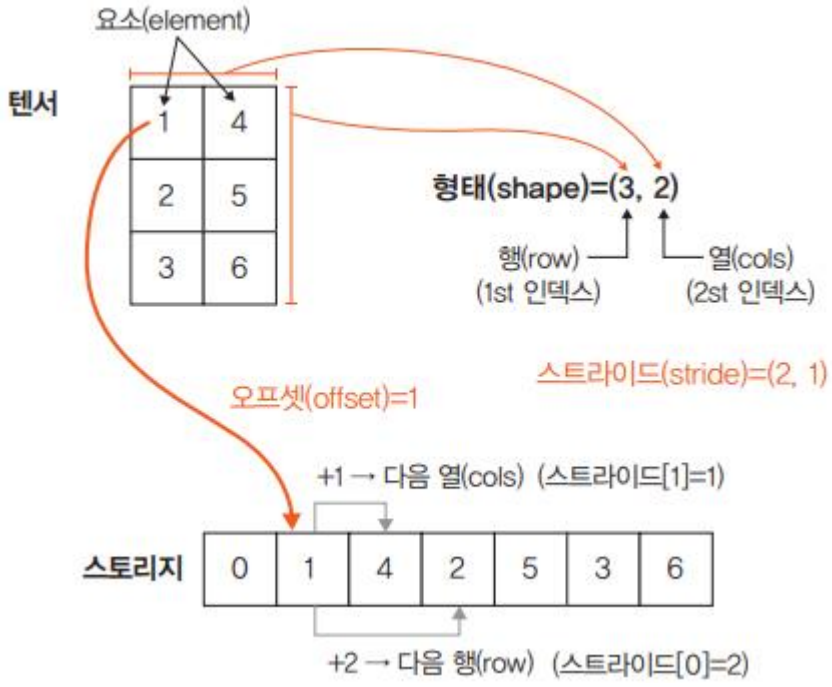


- **파이토치의 아키텍처**
  - 반면에 A의 전치 행렬은 좀 다름
  - 다음 그림의 스토리지에서 4를 얻기 위해서는 1에서 1칸을 뛰어넘어야 하고, 2을 얻기 위해서는 2를 뛰어넘어야 함
  - 텐서에 대한 스토리지의 스트라이드는 (2, 1)

2.1 파이토치 개요

- **파이토치의 아키텍처> 텐서를 메모리에 저장하기**
  - 텐서는 그것이 1차원이든 N차원이든 메모리에 저장할 때는 1차원 배열 형태가 됨
  - 즉, 1차원 배열 형태여야만 메모리에 저장할 수 있음
  - 변환된 1차원 배열을 스토리지(storage)라고 함
  - 스토리지를 이해하기 위해서는 오프셋과 스트라이드 개념을 알아야 함

▼ **전치 행렬을 1차원으로 변환**



  - 오프셋(offset): 텐서에서 첫 번째 요소가 스토리지에 저장된 인덱스
  - 스트라이드(stride): 각 차원에 따라 다음 요소를 얻기 위해 건너뛰기(skip)가 필요한 스토리지의 요소 개수
    즉, 스트라이드는 메모리에서의 텐서 레이아웃을 표현하는 것으로 이해하면 됨
    요소가 연속적으로 저장되기 때문에 행 중심으로 스트라이드는 항상 1

◆ 오프셋과 스트라이드는 데이터 자체가 아닌 행렬/텐서를 구분하기 위해 사용