- Sparse Matrix (희소행렬 : 행렬의 값이 대부분 0인 행렬.

- 희소행렬을 효율적으로 저장하기 위해 아래와 같은 자료구조를 생각할 수 있다.

B [8] [7]

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	
[0]	0	0	2	0	0	0	12	<0, 2, 2>
[1]	0	0	0	0	7	0	0	<0, 6, 12>
[2]	23	0	0	0	0	0	0	<1, 4, 7>
[3]	0	0	0	31	0	0	0	<2, 0, 23>
[4]	0	14	0	0	0	25	0	<3, 3, 31>
[5]	0	0	0	0	0	0	6	<4, 1, 14>
[6]	52	0	0	0	0	0	0	<4, 5, 25>
[7]	0	0	0	0	11	0	0	<5, 6, 6>
								<6, 0, 52>
								<7, 4, 11>

희소행렬

해당 행렬의 값이 '0'이 아닌 것들만 이 리스트의 원소로 추가함.

- SparseMatrix 자료구조 → 희소행렬을 요약한 정보를 리스트 class type의 객체로 만든다.
  - 행렬의 dimension m, n 을 입력받아 희소행렬 리스트를 [[m,n,0]]로 초기 설정한다. ← 해당 리스트의 첫번째 원소는 원본의 행렬에 대한 대략적인 정보이다.
  - append method: [i, j, value] 형태의 리스트를 희소행렬 리스트에 추가한다.
  - shape method: 희소행렬 dimension을 리턴한다. ← 해당 리스트 첫번째 원소가 담긴 원본
  - getValue method: i, j 행렬값을 리턴한다.
  - print method: 희소행렬을 일반적 행렬 형태로 프린트한다. (0으로 구성된 m,n array를 만들고 i,j 행렬값을 업데이트한다.)

↳ <해당 희소행렬을 numpy array class type의 객체로 나타냄!!!>

X. SparseMatrix의 '0'이 아닌 행렬 값을 원본과 달리는 해당 SparseMatrix를 요약한 list의 '첫번째' 원소부터 걸사해야한다. because, 해당 리스트의 '0'번째는 해당 SparseMatrix를 요약한 정보가 들어가있기 때문이다.

행렬의 값이 '0'이 아닌 것들의 갯수.

```python
class SparseMatrix:

    def __init__(self, m, n):
        self.s = [[m, n, 0]]
        self.m = m
        self.n = n

    def append(self,i,j,val):
        if val != 0:
            self.s.append([i,j,val])
        self.s[0][2] = len(self.s) - 1

    def shape(self):
        return (self.m, self.n)

    def getValue(self, i, j):
        val = 0
        for k in range(1,len(self.s)):
            if self.s[k][0] == i and self.s[k][1] == j:
                return self.s[k][2]
        return 0

    def print(self):
        import numpy as np
        _tmp = np.zeros(self.m, self.n))
        for i in range(1,len(self.s)):
            _tmp[self.s[i][0]-1,self.s[i][1]-1] = self.s[i][2]
        print(_tmp)

a = SparseMatrix(3,3)
b = SparseMatrix(3,3)

a.append(1,1,1)
a.append(2,2,2)
a.append(3,3,3)
b.append(1,1,4)
b.append(1,2,7)
b.append(2,3,2)
b.append(3,3,1)

a.print()
b.print()

print(a.getValue(3,3))
```

```
[[1. 0. 0.]
 [0. 2. 0.]
 [0. 0. 3.]]
[[4. 7. 0.]
 [0. 0. 2.]
 [0. 0. 1.]]
3
```

```python
def transpose(a):
    transposed_a = SparseMatrix(a.n,a.m)

    for i in range(len(a.s)):
        if i == 0:
            transposed_a.s[0][2] = a.s[0][2]
        else:
            transposed_a.append(a.s[i][1],a.s[i][0],a.s[i][2])

    return transposed_a

transposed_b = transpose(b)
transposed_b.print()
```

```
[[4. 0. 0.]
 [7. 0. 0.]
 [0. 2. 1.]]
```

```python
def add(a, b):
    if a.shape() != b.shape():
        print("두 행렬의 차원이 달라 더할 수 없습니다.")
    else:
        c = SparseMatrix(a.shape()[0], a.shape()[1])
        u = set()
        for i in range(1,len(a.s)):
            u.add((a.s[i][0], a.s[i][1]))
        for i in range(1,len(b.s)):
            u.add((b.s[i][0], b.s[i][1]))
        for item in u:
            _tmp = a.getValue(item[0], item[1]) + b.getValue(item[0], item[1])
            if _tmp != 0:
                c.append(item[0], item[1], _tmp)
    return c

a.print()
b.print()
c = add(a, b)
c.print()
```

```
[[1. 0. 0.]
 [0. 2. 0.]
 [0. 0. 3.]]
[[4. 7. 0.]
 [0. 0. 2.]
 [0. 0. 1.]]
[[5. 7. 0.]
 [0. 2. 2.]
 [0. 0. 4.]]
```