

SparseMatrix 자료구조

SparseMatrix 클래스를 이용하여 희소행렬 리스트를 `numpy`로 초기화 설정한다.

- `append method` `(i, j, value)` 형태의 리스트를 희소행렬 리스트에 추가한다.
- `shape method`: 희소행렬 dimension을 리턴한다.
- `getValue method` `(i, j)` 행렬값을 리턴한다.
- `print method`: 희소행렬을 일반적 행렬 형태로 프린트한다. (`0`으로 구성된 `numpy array`를 만들고 `(i, j)` 행렬값을 업데이트한다.)

	0	1	2	3	4	5	6
0	0	0	0	3	0	0	0
1	0	14	0	0	0	25	0
2	0	0	0	0	0	0	6
3	52	0	0	0	0	0	0
4	0	0	0	0	11	0	0

희소행렬

행렬의 값이 대부분 0인 행렬.

해당 행렬의 값이 0이 아닌 것들만 이 리스트의 원소로 취함.

→ 희소행렬을 요약한 정보를 리스트 `class type`의 객체로 만든다.

← 해당 리스트의 첫번째 원소는 원본의 행렬에 대한 대략적인 정보이다.

← 해당 리스트 첫번째 원소가 갖고 있는 정보

↳ 해당 희소행렬을 `numpy array class type`의 객체로 나타낼!!!

X. `SparseMatrix`의 '0'이 아닌 행렬 값을 전부 불러와야 할 때는 해당 `SparseMatrix`를 요약한 리스트의 '첫번째' 요소부터 검사해야한다. because, 해당 리스트의 0번째는 해당 `SparseMatrix`를 요약한 정보가 들어가있기 때문이다.

— 행렬의 값이 0이 아닌 것들의 전수.

```

1 public int[] transpose(int[][] matrix) {
2     int row = matrix.length;
3     int col = matrix[0].length;
4     int[][] transposed = new int[col][row];
5     for (int i = 0; i < row; i++) {
6         for (int j = 0; j < col; j++) {
7             transposed[j][i] = matrix[i][j];
8         }
9     }
10    return transposed;
11 }

```

← SparseMatrix 리스트 내의 첫 번째 원소는 해당 희소행렬의 행과 열에 대한 정보가 포함된 '리스트' 객체이다.

← 당연히 해당 리스트 내에는 행렬 값이 0이 아닌 것들만 추가되어야 한다.

← 해당 리스트 내에 '0'이 아닌 행렬값이 들어갔으면, SparseMatrix 리스트의 첫 번째 요소를 업데이트 해주어야 할.

← SparseMatrix 원소들을 차례대로 하나

return을 함수를 종료하고 값을 해당 함수에 돌려주는 역할을 함.

Matrix는 (1,1)부터 시작한다.

이렇게 -1을 빼줘야 우리가 배웠던 행렬형태가 나타난다.

우리가 배웠던 행렬형태가

공통한 차이점!!!

* hump/array로 만든 행렬 (<=) 2차 배열은 (0,0)부터 시작한다.

← 이라는 SparseMatrix를 편지한다.

← transposed 행렬 내의 0이 아닌 행렬값의 개수를 편지하기 위한 행렬과 동일하게 설정해줘야 함.

9 ~~~~~

↳ ex) (2,1)에 존재하던 행렬값을 (1,2)에 넣어야 함.

→ 더 해질 두 객체는 동일한 모양을 띄고 있어야 한다.

← 더 해줘서 나온 결과의 행렬도 당연히 더 해지는 두 행렬의 모양과 일치해야 한다.

행렬값이 0이 아닌 것들만 추가되는 행렬 구조에 넣는다.

이러한 행렬을 생성하는 행렬이다.

↳ 동일한 두 좌표값의 행렬값을 더하여 'hump' 변수에 저장함.

좌표값을 비어있는 집합에 넣는다!!!