행렬 (Matrix)

행렬 (matrix)

- 행(rows)과 열(column)로 이루어진 직사각형의 배열
- 가로줄을 행, 세로줄을 열이라고 부른다.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

행의 개수가 m, 열의 개수가 n이고, ***i***행, ***j***열이 만나는 위치의 성분을 a(***i,j***) 라고 함.

A=(a{i,j})로 표현, 사실 A = m by n matrix로 쓰면 알아들음.

- m=n이면 정사각행렬(square matrix)
- 행이나 열의 길이가 1이면 열 벡터 또는 행 벡터라고 함.

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix}^T$$

행렬 연산

- 공리, 정리는 대학수학 참고

- 행렬은 기본적인 연산 법칙(결합,분배,스칼라배.etc)이 거의 가능하지만 교환법칙은 성립 X

행렬합

행렬곱

기본적인 개념으로 두 행렬의 성분을 곱한다. 이때, 앞 행렬의 열과 뒤 행렬의 행 개수가 같아야 함.

$$AB = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

A = M by N, B = N by P 이면 AB = M by P matrix가 됨.

행렬 예제 코드

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <malloc.h>
#pragma warning(disable: 6031)
#pragma warning(disable: 6011)

int main(void)
{
    srand(time(0));
    int rows, cols;

    printf("행, 열을 입력하세요: ");

    scanf("%d %d", &rows, &cols);
```

```
//////////create A matrix///////////
int **A=(int**)malloc(sizeof(int*)*rows);

for(int i=0;i<rows;i++)
    *(A+i)=(int*)malloc(sizeof(int)*cols);

for(int i=0;i<rows;i++)
{
    for(int j=0;j<cols;j++)
        *(*(A+i)+j)=rand()%10;
}
////////////////////////////////////

//////////create A transpose matrix///
int **A_transpose=(int**)malloc(sizeof(int*)*cols);

for(int i=0;i<cols;i++)
    *(A_transpose+i)=(int*)malloc(sizeof(int)*rows);

for(int i=0;i<rows;i++)
{
    for(int j=0;j<cols;j++)
        *(*(A_transpose+j)+i)=*(*(A+i)+j);
}
////////////////////////////////////

///////////////print matrix///////////
for(int i=0;i<rows;i++)
{
    for(int j=0;j<cols;j++)
        printf("%d ", *(*(A+i)+j));
    printf("\\n");
}

printf("\n");

for(int i=0;i<cols;i++)
```

```
    {
        for(int j=0;j<rows;j++)
            printf("%d ", *(*(A_transpose+i)+j));
        printf("\\n");
    }
    ////////////////////////////////////

    for (int i = 0; i < rows; i++)
        free(*(A + i));
    free(A);
    for (int i = 0; i < cols; i++)
        free(*(A_transpose + i));
    free(A_transpose);
    return 0;
}
```