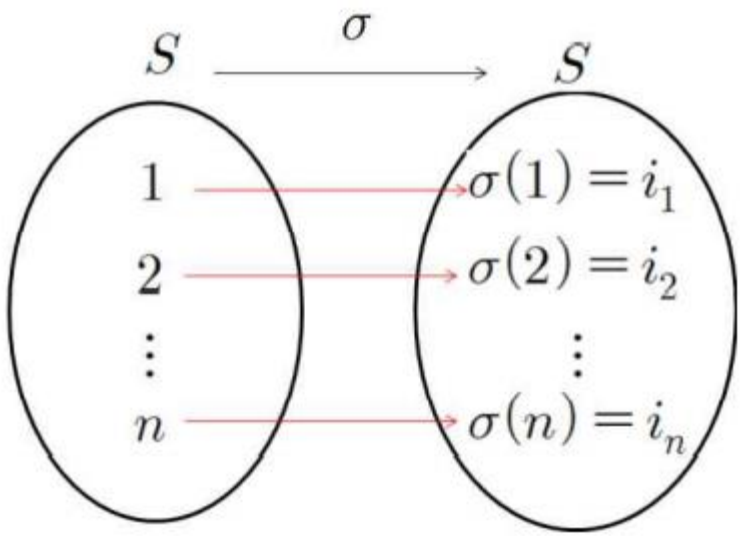
행렬식의 정의와 기본정리

• 자연수의 집합 $S = \{1, 2, \ldots, n\}$에서 S로의 **일대일 대응(one-to-one corresponding)** 혹은 **전단사 함수(bijective function)**를 S의 **치환(permutation, 순열)**이라 한다.



치환을 간단히 $\sigma = (\sigma(1)\ \sigma(2)\ \cdots\ \sigma(n)) = (i_1\ i_2\ \cdots\ i_n)$로 나타낸다. 치환 σ는 일대일 대응이므로 치역 $\{i_1, i_2, \ldots, i_n\}$은 $1, 2, \ldots, n$의 숫자를 일렬로 배열하는 것에 지나지 않는다. 따라서 $S = \{1, 2, \ldots, n\}$의 치환은 모두 $n! = (1)(2)\cdots(n)$개이다.

집합 $S = \{1\}$의 치환은 $1! = (1) = 1$개 존재한다. $\begin{pmatrix} 1 \\ 1 \end{pmatrix} = (1)$

집합 $S = \{1, 2\}$의 치환은 $2! = (1)(2) = 2$개 존재한다. $\begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix} = (1\ 2),\ \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} = (2\ 1)$

집합 $S = \{1, 2, 3\}$의 치환은 $3! = (1)(2)(3) = 6$개 존재한다.

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} = (1\ 2\ 3),\ \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} = (2\ 3\ 1),\ \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix} = (3\ 1\ 2)$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix} = (1\ 3\ 2),\ \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix} = (3\ 2\ 1),\ \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix} = (2\ 1\ 3)$$

집합 $S = \{1, 2, 3, 4\}$의 치환은 $4! = (1)(2)(3)(4) = 24$개 존재한다.

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix} = (1\,2\,3\,4),\ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 4 & 3 \end{pmatrix} = (1\,2\,4\,3),\ \cdots,\ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix} = (4\,3\,2\,1)$$

• 치환 $(i_1\ i_2\ \cdots\ i_n)$에서 **반전(inversion)**이란 큰 자연수가 작은 자연수보다 더 왼쪽에 먼저 나타나는 경우를 말한다. 예를 들어, 치환 $(1\ 4\ 2\ 3)$에서 4는 2보다, 3보다 더 왼쪽에 있으므로 반전이 2번 일어났다.

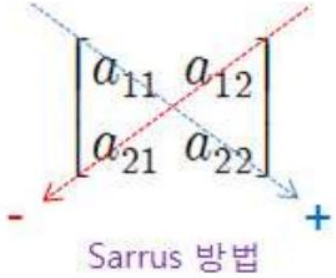
• 치환이 가진 반저의 총 개수가 짝수이면 이 치환은 **짝치환(even permutation)**, 홀수이면 **홀치환(odd permutation)**이라 한다.

```
Permutation([5,1,2,4,3]).inversions() # 반전이 일어난 부분
```

```
Permutation([5,1,2,4,3]).number_of_inversions() # 반전의 개수
```

```
Permutation([5,1,2,4,3]).is_even() # 짝치환인지 확인
```

- $sgn(\sigma) = \begin{cases} 1, & \sigma = \text{짝치환} \\ -1, & \sigma = \text{홀치환} \end{cases}$ **부호화 함수(signature function)**



Sarrus 방법

- 행렬 $A = [a_{ij}]$가 n차 정사각행렬일 때, A의 **행렬식(determinant)**을 $det(A)$ 또는 $|A|$로 나타내고 다음과 같이 정의한다.

$$det(A) = |A| = \sum_{\sigma \in S_n} sgn(\sigma) a_{1\sigma(1)} a_{2\sigma(2)} \cdots a_{n\sigma(n)}$$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

여기서 S_n은 집합 $S = \{1, 2, \ldots, n\}$의 치환들의 집합이다.

$$\sigma = (1\ 2),\ \tau = (2\ 1)$$

$$sgn(\sigma) = 1,\ sgn(\tau) = -1$$

$$det(A) = |A| = sgn(\sigma) a_{1\sigma(1)} a_{2\sigma(2)} + sgn(\tau) a_{1\tau(1)} a_{2\tau(2)}$$

$$= a_{11}a_{22} - a_{12}a_{21}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

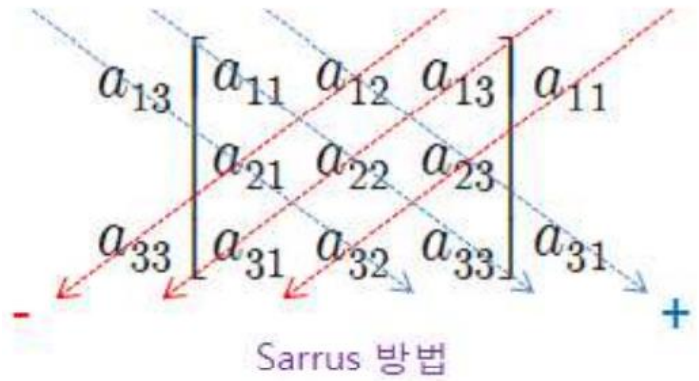
```
B=matrix(QQ, 3, 3, [1, 2, 3, -4, 5, 6, 7, -8, 9])
print B.det()
```

$\sigma_1 = (1\ 2\ 3),\ \sigma_2 = (2\ 3\ 1),\ \sigma_3 = (3\ 1\ 2)$: 짝함수, $sgn(\sigma_i) = 1\ (i = 1, 2, 3)$

$\sigma_4 = (1\ 3\ 2),\ \sigma_5 = (3\ 2\ 1),\ \sigma_6 = (2\ 1\ 3)$: 홀함수, $sgn(\sigma_i) = -1\ (i = 4, 5, 6)$

$$det(A) = |A| = sgn(\sigma_1)a_{1\sigma_1(1)}a_{2\sigma_1(2)}a_{3\sigma_1(3)} + sgn(\sigma_2)a_{1\sigma_2(1)}a_{2\sigma_2(2)}a_{3\sigma_2(3)} + sgn(\sigma_3)a_{1\sigma_3(1)}a_{2\sigma_3(2)}a_{3\sigma_3(3)}$$
$$+ sgn(\sigma_4)a_{1\sigma_4(1)}a_{2\sigma_4(2)}a_{3\sigma_4(3)} + sgn(\sigma_5)a_{1\sigma_5(1)}a_{2\sigma_5(2)}a_{3\sigma_5(3)} + sgn(\sigma_6)a_{1\sigma_6(1)}a_{2\sigma_6(2)}a_{3\sigma_6(3)}$$

$$= a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{11}a_{23}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33}$$



Sarrus 방법

Sarrus 방법은 4차 이상의 행렬식에 대해서는 적용할 수 없다.

• 행렬식의 성질

1. 정사각행렬 A의 행렬식과 A의 전치행렬 A^T의 행렬식의 값은 같다.

```
B=matrix(QQ, 3, 3, [1, 2, 3, -4, 5, 6, 7, -8, 9])
print B.det()
```

```
B=matrix(QQ, 3, 3, [1, 2, 3, -4, 5, 6, 7, -8, 9]) print
B.transpose().det()
```

2. 정사각행렬 A의 한 행(열)의 성분이 모두 0이면 $|A| = 0$이다.

3. 정사각행렬 A의 두 행(열)이 비례하면 $|A| = 0$이다.

```
A=matrix(QQ, 3, 3, [1, 2, 3, -1, 0, 7, 2, 4, 6])
print A.det()
```

4. 행렬 B가 정사각행렬 A의 두 행(열)을 바꾸어서 얻어진 행렬이라면 $|B| = -|A|$ 이다.

5. 정사각행렬 A의 한 행(열)을 k배하여 얻어진 행렬을 B라 하면 $|B| = k|A|$이다.

6. 정사각행렬 A의 한 행(열)의 k배를 다른 행(열)에 더하여 얻어진 행렬을 B라 하면 $|B| = |A|$이다.

7. 정사각행렬 A가 삼각행렬이면 A의 행렬식은 주대각선성분의 곱과 같다.

$$A = \begin{bmatrix} 0 & 1 & 5 \\ 3 & -6 & 9 \\ 2 & 6 & 1 \end{bmatrix}$$

$$= (-3)(1)(1)(-55) = 165$$

$$|A| = \begin{vmatrix} 0 & 1 & 5 \\ 3 & -6 & 9 \\ 2 & 6 & 1 \end{vmatrix} = - \begin{vmatrix} 3 & -6 & 9 \\ 0 & 1 & 5 \\ 2 & 6 & 1 \end{vmatrix} = -3 \begin{vmatrix} 1 & -2 & 3 \\ 0 & 1 & 5 \\ 2 & 6 & 1 \end{vmatrix} = -3 \begin{vmatrix} 1 & -2 & 3 \\ 0 & 1 & 5 \\ 0 & 10 & -5 \end{vmatrix} = -3 \begin{vmatrix} 1 & -2 & 3 \\ 0 & 1 & 5 \\ 0 & 0 & -55 \end{vmatrix}$$

$R_1 \leftrightarrow R_2$ $(-2)R_1 + R_3$ $(-10)R_2 + R_3$

• **기본행렬의 행렬식**

1. E가 I_n의 한 행에 $k(\neq 0)$를 곱한 것이면 $|E| = k$,

2. E가 I_n의 두 행을 서로 바꾼 것이면 $|E| = -1$,

3. E가 I_n의 한 행에 $k(\neq 0)$배하여 다른 행에 더한 것이면 $|E| = 1$,

4. A는 $n \times n$ 행렬이고 E는 기본행렬이다. 그러면, $|EA| = |E||A|$이다.

- A가 가역행렬일 필요충분조건은 $|A| \neq 0$이다.

```
A=matrix(QQ, 2, 2, [1, 2, 3, 4])
B=matrix(QQ, 2, 2, [2, -1, 1, 2])
C=A*B
print "det(AB)=", C.det()
print "det(A)*det(B)=", A.det()*B.det()
```

- 두 행렬 $A,\ B$가 n차 정사각행렬일 때, $|AB| = |A||B|$이다.

- A가 가역행렬이면, $A^{-1}| = \frac{1}{|A|}$이다.

```
A=matrix(QQ, 2, 2, [1, 2, 3, 4])
Ai=A.inverse()
print "det(A)=", A.det()
print "det(A^(-1))=", Ai.det()
```

- 정사각행렬 $A = [a_{ij}]$의 i행과 j열을 제거하여 만든 부분행렬을 A_{ij}라 하고 그의 행렬식 $M_{ij} = |A_{ij}|$를 A의 a_{ij}에 대한 **소행렬식(minor)**이라 한다. 또, $C_{ij} = (-1)^{i+j} M_{ij}$를 A의 a_{ij}에 대한 **여인자(cofactor)**라고 한다. 그리고 $[C_{ij}]^T$를 A의 **수반행렬(adjugate, adjunct, classical adjoint matrix)**이라 하고 $adjA$로 나타낸다.

$$adjA = \begin{bmatrix} C_{11} & C_{21} & \cdots & C_{n1} \\ C_{12} & C_{22} & \cdots & C_{n2} \\ \vdots & \vdots & \cdots & \vdots \\ C_{1n} & C_{2n} & \cdots & C_{nn} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & \cdots & C_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nn} \end{bmatrix}^T = [C_{ij}]^T$$

```
A=matrix(QQ, 3, 3, [3, -2, 1, 5, 6, 2, 1, 0, -3])
print A.adjoint()
```

$$A = [a_{ij}]_{n\times n}$$

$|A| = a_{i1}C_{i1} + a_{i2}C_{i2} + \cdots + a_{in}C_{in}$: 행렬 A의 i행에 관한 **여인자 전개(cofactor expansion)**

$|A| = a_{1j}C_{1j} + a_{2j}C_{2j} + \cdots + a_{nj}C_{nj}$: 행렬 A의 j열에 관한 **여인자 전개(cofactor expansion)**

$$|A| = a_{i1}C_{j1} + a_{i2}C_{j2} + \cdots + a_{in}C_{jn} = 0 \ (i \neq j)$$

$$A(adjA) = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} C_{11} & C_{21} & \cdots & C_{n1} \\ C_{12} & C_{22} & \cdots & C_{n2} \\ \vdots & \vdots & \cdots & \vdots \\ C_{1n} & C_{2n} & \cdots & C_{nn} \end{bmatrix} = \begin{bmatrix} |A| & 0 & \cdots & 0 \\ 0 & |A| & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & |A| \end{bmatrix} = |A|I_n$$

```
A=matrix(QQ, 3, 3, [3, -2, 1, 5, 6, 2, 1, 0, -3])
print "det(A)=", A.det()
print "A*adj(A)=" print A*A.adjoint()
```

$A = [a_{ij}]_{n\times n}$가 가역행렬이면, $A^{-1} = \frac{1}{|A|}adjA$

```
A=matrix(QQ, 3, 3, [3, -2, 1, 5, 6, 2, 1, 0, -3])
dA=A.det()
adjA=A.adjoint()
print "(1/dA)*adjA=" print (1/dA)*adjA
print print "A^(-1)=" print A.inverse()
```

• **행렬식의 응용**

행렬식의 개념을 처음 소개한 것은 1683년 일본의 세키 고와(Takakazu Seki-Kowa)이다. 행렬식(determinant)의 어원은 해의 존재성을 판별한다는 의미에서 유래되었으며, 현재 의미로 행렬식을 사용한 것은 1815년 코시(Cauchy)였다. 여기서는 행렬식의 무수히 많은 응용 중 기하학적 응용과 대수학적 응용의 몇 가지를 소개한다.

행렬식을 이용하면 **넓이**나 **부피**, **직선의 방정식**, **타원의 방정식**, **평면의 방정식**을 쉽게 구할 수 있다. 또 **Vandermonde 행렬**의 행렬식은 통계자료와 실험실에서 나오는 이산적인 데이터에 여러분이 배운 연속함수를 다루는 수학을 연결시켜주는 다리이다.

예제 1. 평면에서 서로 다른 두 점 (x_1, y_1)과 (x_2, y_2)를 지나는 직선의 방정식은 행렬식을 이용하여 구하면 다음과 같다.

$$\begin{vmatrix} 1 & x & y \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{vmatrix} = 0$$

예제 2. 공간에서 서로 다른 세 점 (x_1, y_1, z_1), (x_2, y_2, z_2) 그리고 (x_3, y_3, z_3)를 지나는 평면의 방정식은 행렬식을 이용하여 표현하면 다음과 같다.

(이 슬라이드는 3주 강의노트 참고)

$$\begin{vmatrix} 1 & x & y & z \\ 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \end{vmatrix} = 0$$

예제 3. 평면에서 서로 다른 두 벡터 $\mathbf{a} = (x_1, y_1)$과 $\mathbf{b} = (x_2, y_2)$로 만들어지는 평행사변형의 넓이는 행렬식을 이용하여 구하면 다음과 같다.

A가 2차 정사각행렬이면 $|detA|$는 행렬 A의 두 개의 열벡터에 의해서 결정되는 평행사변형의 넓이와 같다.

$$\left| \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} \right|$$

예제 4. 공간에서 서로 다른 세 벡터 $\mathbf{a} = (x_1, y_1, z_1)$, $\mathbf{b} = (x_2, y_2, z_2)$ 그리고 $\mathbf{c} = (x_3, y_3, z_3)$로 만들어지는 평행육면체의 부피는 행렬식을 이용하여 구하면 다음과 같다.

A가 3차 정사각행렬이면 $|detA|$는 행렬 A의 세 개의 열벡터에 의해서 결정되는 평행육면제의 부피와 같다.

$$\left| \begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{vmatrix} \right|$$

• 두 벡터 $\mathbf{u}$, $\mathbf{v} \in \mathbb{R}^n$가 만드는 평행사변형의 넓이는 $A = [\mathbf{u}|\mathbf{v}]$라 할 때, $\sqrt{detA^T A}$ 이다.

• **Vandermonde 행렬과 행렬식**

$(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$을 xy-평면상의 x좌표가 서로 다른 n개의 점이 있으면 이들 n개의 점을 모두 지나는 $n-1$차 다항식

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

을 결정해 보자. 이들 n개의 점이 주어진 식을 만족하므로

$$a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_{n-1} x_1^{n-1} = y_1$$

$$a_0 + a_1 x_2 + a_2 x_2^2 + \cdots + a_{n-1} x_2^{n-1} = y_2$$

$$\vdots$$

$$a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_{n-1} x_n^{n-1} = y_n$$

을 만족한다. 여기에서 $x_1, x_2, \ldots, x_n$이 서로 다르므로 계수행렬 V_n의 행렬식

$$detV_n = \begin{vmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{vmatrix} \neq 0$$

이고 V_n을 n차 **Vandermonde 행렬**이라 한다.

$n = 3$인 경우

$$detV_3 = \begin{vmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{vmatrix} = det(V_3^T) = \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 \\ 0 & x_2 - x_1 & x_3 - x_1 \\ 0 & x_2(x_2 - x_1) & x_3(x_3 - x_1) \end{vmatrix}$$

$$= \begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ x_2(x_2 - x_1) & x_3(x_3 - x_1) \end{vmatrix} = (x_2 - x_1)(x_3 - x_1) \begin{vmatrix} 1 & 1 \\ x_2 & x_3 \end{vmatrix} = (x_2 - x_1)(x_3 - x_1)(x_3 - x_2)$$

$$\therefore \ |V_3| = \prod_{1 \le i < j \le 3} (x_j - x_i)$$

$$\therefore \ detV_n = \begin{vmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{vmatrix} = \prod_{1 \le i < j \le n} (x_j - x_i)$$

예제 5. 3개의 점 $(39, 34)$, $(99, 47)$, $(38, 58)$을 지나는 그래프를 Vandermonde 행렬을 이용하여 구하여라.

```
def Vandermonde_matrix(x): # Vandermonde 행렬 생성
 n=len(x)
 A=matrix(RDF, n, n, [[z^i for i in range(n)] for z in x])
 return A
x=[39, 99, 38] # x 좌표
V=Vandermonde_matrix(x)
y=vector([34, 47, 58]) # y 좌표
print "V="
print V
print
print "a=", V.solve_right(y)
```

과제 5. 다음을 Sage 명령어를 이용하여 구하여라.

(1) 행렬 $A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 1 & 3 \end{bmatrix}$ 에 대하여 $det\left(\frac{1}{2}adj(A)\right)$을 구하여라.

(2) 공간에서 서로 다른 세 점 $(4, 5, 6),\ (-3, 7, 8)$ 그리고 $(6, 02)$를 지나는 평면의 방정식을 행렬식을 이용하여 구하여라.

(3) 4개의 점 $(-2, 11),\ (-1, 2),\ (1, 2),\ (2, -1)$을 지나는 그래프를 Vandermonde 행렬을 이용하여 구하여라.