기하학

201807042 이무현

목차

목차

기하학 (Geometry)

기하학 : 점, 직선, 곡선, 면, 부피 등 공간의 성질을 연구하는 수학 분야

계산 기하학 : 기하학에 관한 알고리즘을 다루는 컴퓨터과학의 한 분야

최소 볼록 집합, Line segment intersection, 보로노이 다이어그램 등

고전 기하학

- 피타고라스의 정리
- 유클리드 기하학

근대 기하학

- 해석 기하학
- 비유클리드 기하학
- 위상수학

현대 기하학

등

선형 대수학

대수학 : 일련의 공리들을 만족하는 수학적 구조들의 일반적인 성질을 연구하는 분야

집합과 그 위에 정의된 연산에 대한 규칙을 연구하는 학문

$f(x) = y$

선형 대수학 : 벡터공간과 선형사상에 관한 대수학

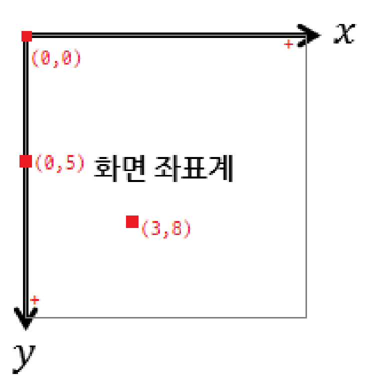
일차함수, 벡터, 행렬 등을 연구하는 학문이다.

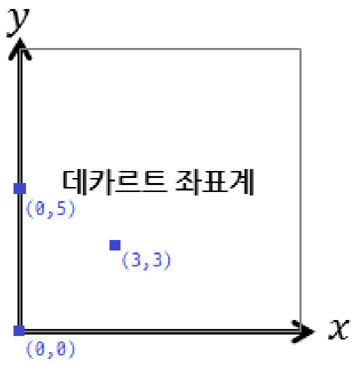
주의점

1. 좌표계

수학에서 사용하는 좌표계는 데카르트 좌표계로 좌측 하단을 원점으로 하는 좌표계를 갖는다.

컴퓨터에서 사용하는 좌표계는 좌측 상단을 원점으로 하는 좌표계를 사용한다.





2. 좌표의 표현 방식

수학에서 좌표는 (x, y) 방식으로 표현한다.

컴퓨터에서는 2차원 배열의 생성 구조때문에 (y, x)의 방식을 사용한다.

두 방식을 혼용하게 되면 기준이 모호해지는 등 문제가 발생할 가능성이 커진다.

(3차원은 (z, y, x) 와 같이 나타낸다.)

주의점

3. 퇴화 도형

일반 위치 : 도형들의 상대적 위치가 일반적인 경우

퇴화 도형 : 일직선 상에 있는 세 개 이상의 점들

서로 평행하거나 겹치는 직선 / 선분들

넓이가 0인 다각형들

다각형들의 변들이 서로 겹치는 경우

퇴화 도형의 경우 오류가 발생할 가능성이 크기 때문에 주의해야 한다.

4. 입력 값 제한

sqrt에 아주 작은 음수가 들어가는 경우

```
sqrt( max( 0.0, x ) );                    // 들어가는 값의 범위를 0.0 이상으로 제한한다.
```

acos, asin의 값에 -1 ~ +1 범위 이외의 값이 들어가는 경우

```
acos(max( -1.0, min( 1.0, x ) ) )   // 들어가는 값의 범위를 -1.0 이상 1.0 이하로 제한한다.
```

math.h

수학과 관련된 자주사용되는 함수를 저장한 라이브러리이다.
부동 소수점 (실수) 에서 연산이 이뤄진다.

모든 함수는 허용된 값의 범위가 존재하며 범위를 넘어가면 오류가 발생한다.
결과의 크기가 너무 크면 함수는 HUGE_VAL 값을 리턴하며, errno가 ERANGE로 설정된다.
결과가 너무 작으면 0을 리턴한다.

함수

- 기본 함수 (거듭제곱, 거듭제곱근, 올림, 내림, 절댓값, 나머지)
- 삼각 함수 (사인, 코사인, 탄젠트, 아크 사인, 아크 코사인 등)
- 지수, 대수 함수

상수

math.h (기본함수)

보편적으로 많이 쓰이는 함수들이다.

거듭제곱, 거듭제곱근, 올림, 내림, 절댓값, 나머지에 대한 함수들이다.

반환형	함수	매개변수	설명
double	pow	(double x, double y)	x^y 를 구한다.
double	sqrt	(double x)	$\sqrt{x}$ 를 구한다.
double	ceil	(double x)	x보다 작지 않은 가장 작은 정수를 구한다. (올림)
double	floor	(double x)	x보다 크기 않은 가장 큰 정수를 구한다. (내림)
double	fabs	(double x)	x의 절댓값을 구한다.
double	fmod	(double x, double y)	x를 y로 나눈 나머지를 구한다.

math.h (삼각함수)

삼각함수에서 사용하는 함수들이다.

기본 삼각함수 뿐 아니라 역 삼각함수, 쌍곡선 함수까지도 지원한다.

반환형	함수	매개변수	설명
삼각함수			
double	sin	(double x)	사인 x 를 구한다.
double	cos	(double x)	코사인 x 를 구한다.
double	tan	(double x)	탄젠트 x 를 구한다.
역 삼각함수			
double	asin	(double x)	아크 사인 x를 구한다.
double	acos	(double x)	아크 코사인 x를 구한다.
double	atan	(double x)	아크 탄젠트 x를 구한다. (-π/2 ~ π/2)
double	atan2	(double x, double y)	아크 탄젠트 y/x를 구한다. (-π ~ π)
쌍곡선 함수			
double	sinh	(double x)	하이퍼볼릭 사인 x를 구한다.
double	cosh	(double x)	하이퍼볼릭 코사인 x를 구한다.

math.h (지수, 대수 함수)

지수와 대수 연산에 필요한 함수들이다.

반환형	함수	매개변수	설명
double	exp	(double x)	e^x를 구한다.
double	frexp	(double x, int * exp)	지수와 기수를 나눈다. 지수를 exp가 가리키는 변수에 저장하고 가수를 반환한다.
double	ldemp	(double x, int exp)	$x * 2^{exp}$를 반환한다.
double	log	(double x)	$\log_e x$를 구한다.
double	log10	(double x)	$\log_{10} x$를 구한다.
double	modf	(double x, double * inpart)	정수부와 소수부를 나눈다. 정수부를 intpart가 가리키는 변수에 저장하고 소수부를 반환한다.

math.h (상수)

자주 사용하는 상수, 연산 결과에 대해 저장 되어있다.

이름	설명	이름	설명
HUGE_VAL	계산 결과가 너무 커 오버플로우가 나면 이 값을 반환한다.		
M_E	자연상수 e	M_PI	원주율 π
M_LOG2E	$\log_2 e$	M_PI_2	π / 2
M_LOG10E	$\log_{10} e$	M_PI_4	π / 4
M_LN2	$\log_e 2$	M_1_PI	1 / π
M_LN10	$\log_e 10$	M_2_PI	2 / π
M_SQRT2	√2	M_2_SQRTPI	2 / √π
M_1_SQRT2	1 / √2		

점

점 : 크기가 없고 위치만 있는 도형

유클리드 기하학의 점 : 점은 넓이가 없는 위치이다.

1. 변수로 표현하기

```
int x;      // x 좌표
int y;      // y 좌표
            // 사용할 때마다 변수를 생성 해야하며, 관리가 어렵다.
```

2. 구조체(Class)로 표현하기

```
struct point{
    int x;      // x 좌표
    int y;      // y 좌표
}
point p1 = {1,2};
            // 보다 직관적으로 사용 가능하다.
```

점

3. pair<> STL로 표현하기

```
pair< int , int > point(1,2);

point = pair< int, int >{ 1 , 2 };

point.first = 1;
point.second = 2;
        // 선언 할 필요가 없다, 관리가 용이하다
        // 직관성이 다소 떨어진다.
```

직선

직선 : 두 사이를 가장 짧은 거리로 연결한 선

유클리드 기하학의 직선 : 폭이 없는 길이,
고르게 놓여있는 점 위에 있는 선

직선 방정식

1. 기울기가 m이고 y절편이 n인 직선의 방정식

y = m*x + n

2. 기울기가 m 이고, 점 (x1, y1)을 지나는 직선의 방정식

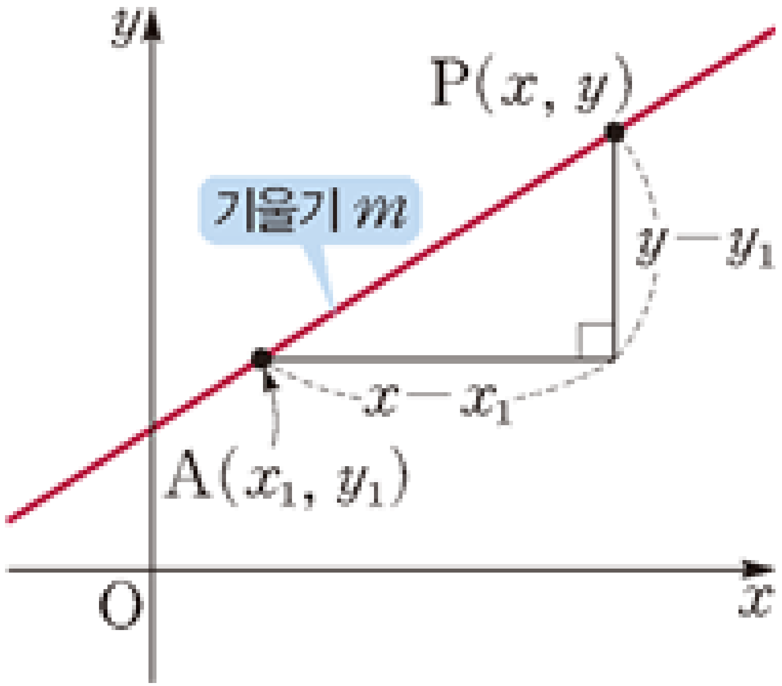
y - y1 = m(x - x1)

두 점의 기울기를 구하는 공식

기울기 = (y2 - y1) / (x2 - x1)

두 점을 지나는 직선의 방정식

(y - y1) = ((y2 - y1) / (x2 - x1)) * (x - x1)



직선

x절편과 y절편이 주어진 직선의 방정식 (a != 0, b != 0 일 때)

$x/a + y/b = 1$

직선을 다음과 같이 나타내었을 때 조건에 따라 직선이 해당 형태를 나타낸다.

$ax + by + c = 0$

a != 0, b != 0 일 때

기울기 : -a / b 절편 : -c / b

a != 0, b == 0 일 때

y축에 평행한 직선

a == 0, b != 0 일 때

x축에 평행한 직선

두 점사이의 거리

피타고라스 정리에 따라

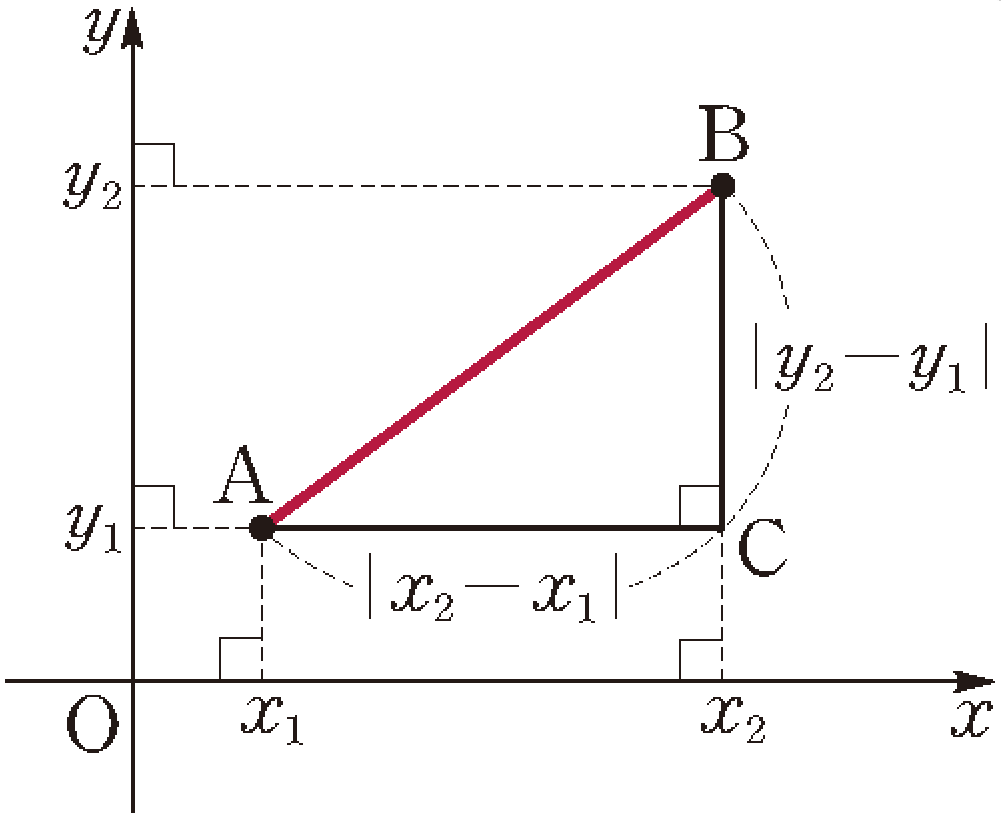
직선 AB의 길이 = √ ((x2-x1)² + (y2 - y1)²)

코드

```
sqrt(pow(x2 - x1 , 2) + pow(x2 - x1 , 2))
```

원점과 한 점의 거리

√ ((x1)² + (y1)²)



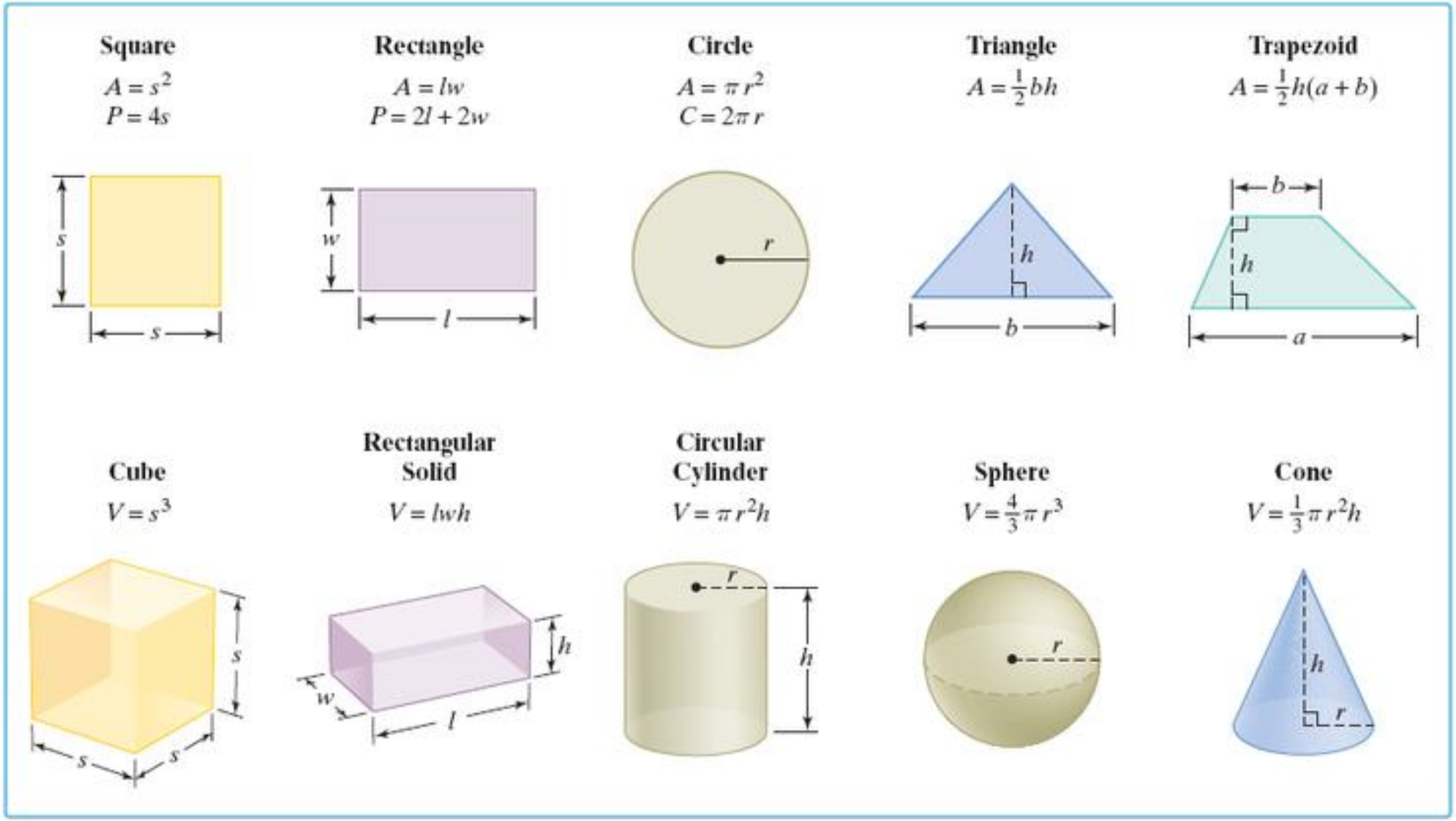
면, 도형, 다각형, 다면체

면은 길이와 폭을 갖고 있다.

면의 끝은 선으로 이루어져 있다.

도형은 경계를 갖고 있는 것이다.

다각형은 직선으로 이루어진 것이다.



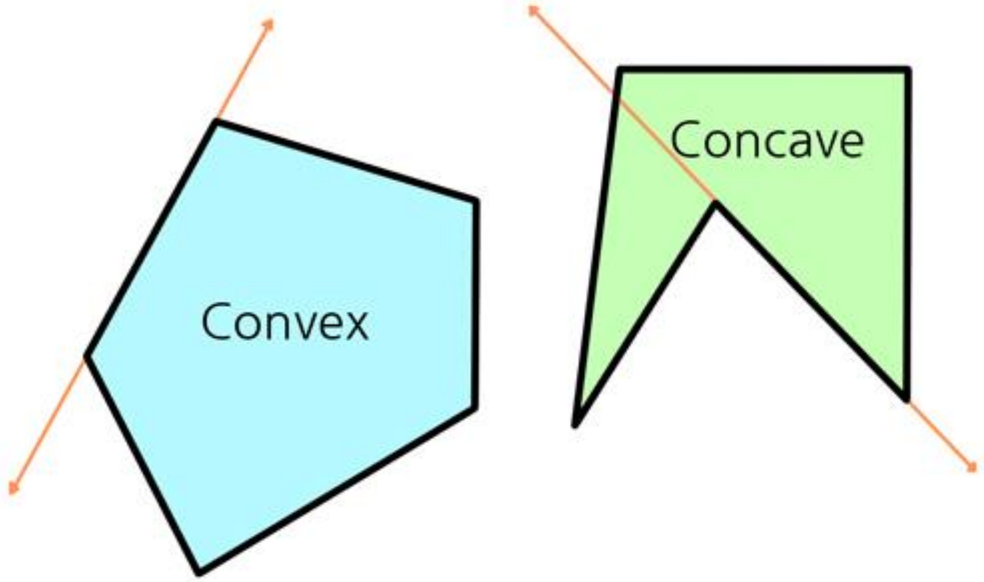
다각형

다각형 : 점들의 집합
이웃한 점들은 직선으로 연결되고, 처음과 마지막 점을 직선으로 연결한 닫힌 모양의 도형

블록 다각형 : 모든 내각이 180도 미만인 다각형
두 볼록 다각형의 교집합은 항상 볼록 다각형이다.

오목 다각형 : 180가 넘는 내각을 갖는 다각형

단순 다각형 : 연속한 두 변 이외에는 어느 두 변도 교차하지 않는 다각형
경계가 스스로 교차하지 않는 다각형



다각형 클리핑 알고리즘,
블록 껍질 알고리즘,
회전하는 캘리퍼스 알고리즘

원

원 : 한 점으로부터 가장자리까지의 거리가 일정한 도형

중점 : 원의 중심에 있는 기준이 되는 점

반지름 r : 중점으로 부터 한 점까지의 거리

지름 : 원의 한 점으로부터 중점을 지나가는 선분의 길이

원주율 : 원의 둘레와 지름의 비 (π = 3.14 M_PI 에 정의되어 있음)

원주 : 원의 둘레

원주 공식

2 * M_PI * r

원 넓이 공식

M_PI * pow(r , 2) || M_PI * r * r

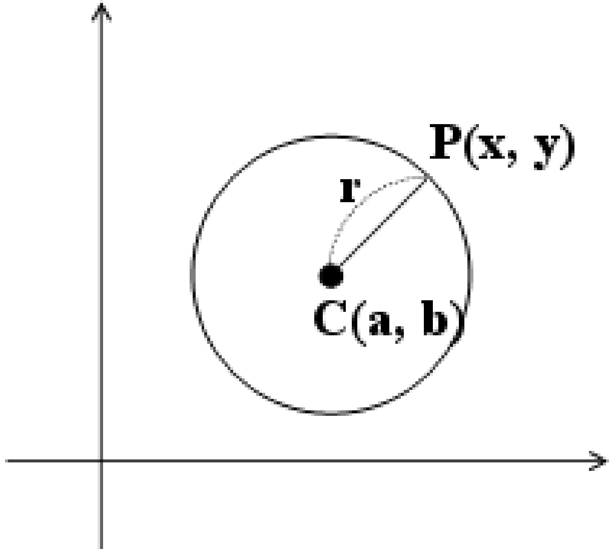
부채꼴 호의 길이와 넓이

호의 길이 l = 원주 * x / 360 넓이 = 넓이 * x / 360 = 1/2 * r * l

원 방정식

중심이 a, b이고, 반지름의 길이가 r인 원의 방정식

$(x - a)^2 + (y - b)^2 = r^2$



직선과 원의 관계

중점과 수선의 거리 또는 판별식을 통해 판정이 가능하다.

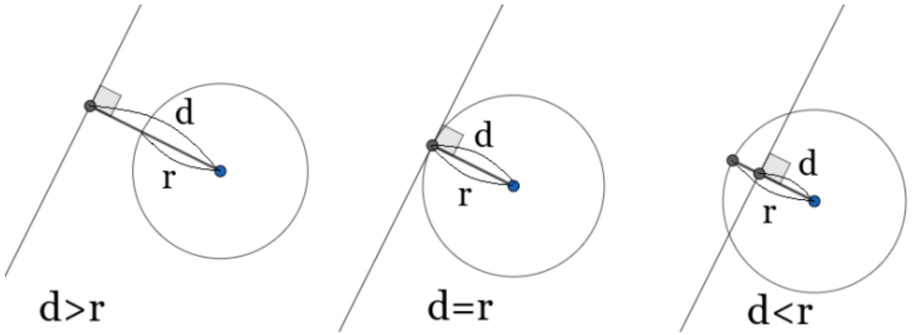
1. 만나지 않는다. $D < 0$ 허근
2. 접한다. $D = 0$ 중근
3. 두 점에서 만난다. $D > 0$ 서로 다른 두 실근

판별식을 이용한 방식

판별식 D : b^2 - 4*a*c (a*x^2 + b*x + c = 0)

y = -ax + n 으로 정의 한 후 원 방정식에 대입한다.

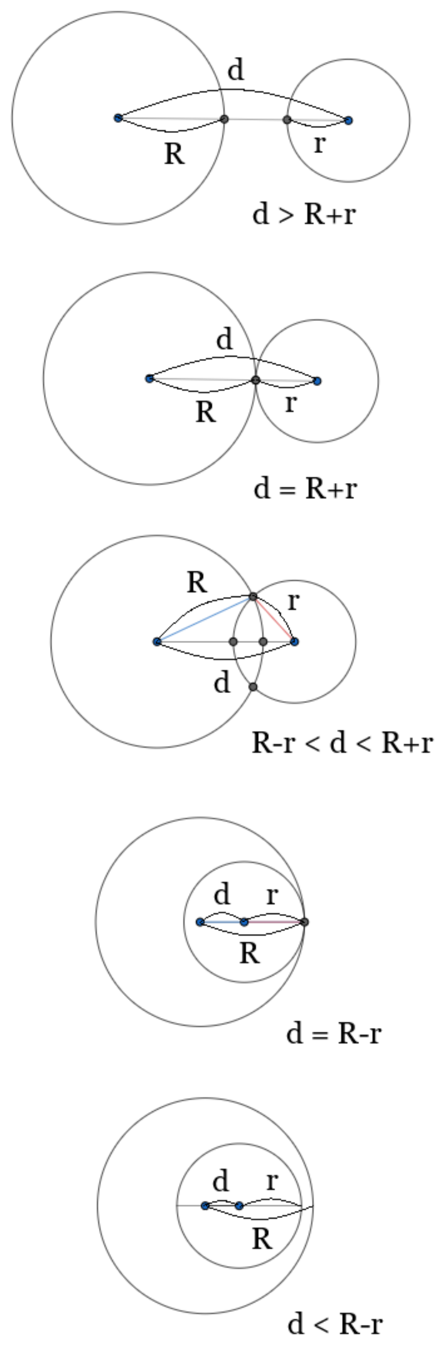
이후 판별식에 넣어 확인한다.



원과 원의 관계

두 점의 중점과 반지름의 합의 관계로 파악한다.

1. 외부에 있다.
2. 외접한다.
3. 두 점에서 만난다.
4. 내접한다.
5. 내부에 있다.



세 점을 지나는 원의 방정식

$x ^ 2 + y ^ 2 + a * x + b * y + c = 0$ 형태로 바꾼 후 연립 방정식

접선의 방정식

원의 중심과 접점을 이은 반지름은 접선과 수직이다

수직인 선분의 기울기의 곱은 -1이다.

기울기를 알 때

y = mx + n 의 식에 기울기를 대입한 후 원의 방정식에 대입한다.

판별식이 0이 되는 x,y 값을 찾는다.   (이 때 결과값은 두가지 경우가 나온다.)

접점을 알 때

두 직선이 수직이면 기울기의 곱 = -1 이라는 것을 이용한다.

원에서 접점 기울기= (y - b) / (x - a)

점선의 기울기 = -((x - a) / (y - b))

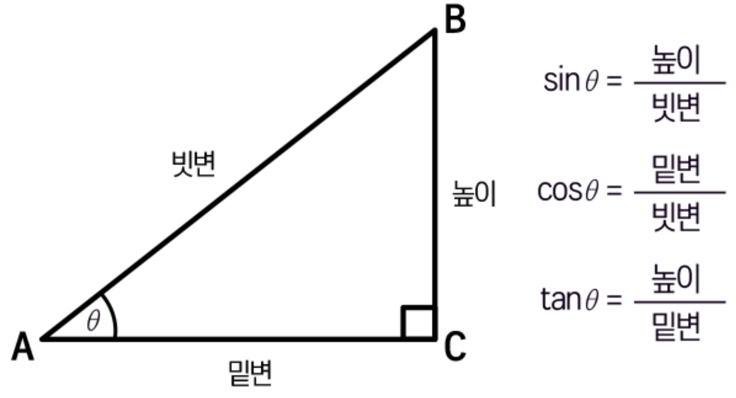
접선의 방징식 = y - y1 = ((x1 - a) / (y1 - b)) * (x - x1)

= (x1 - a) * (x – a) + (y1 – b) * (y – b) = r ^ 2

삼각함수

삼각비 : 직각삼각형에서 두 변에 대한 길이의 비를 나타내는 것

삼각 함수 : 직각 삼각형에 대하여 일정한 비의 관계를 함수로 나타낸 것



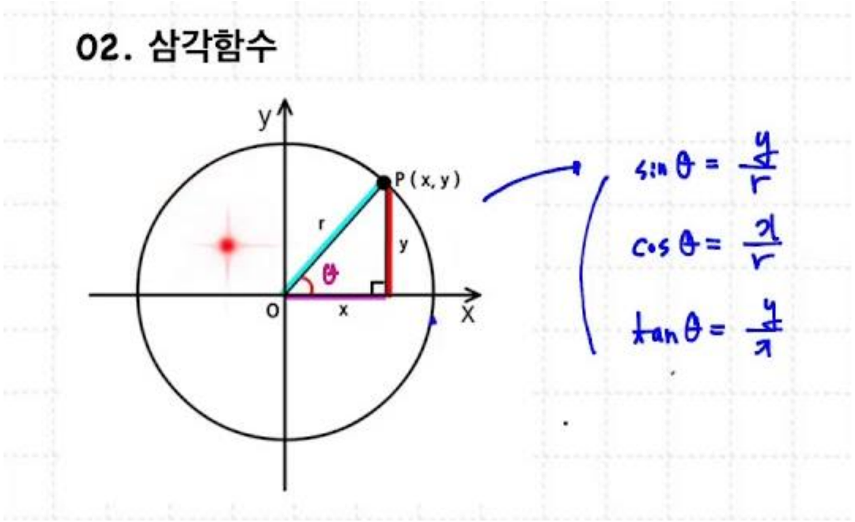
sinΘ : y / r

cosΘ : x / r

tanΘ : y / x

사용처 : 거리 측정, 측량, 음악, 파동

거리 길이, 주기적인 변화에 사용한다.



각도

60분법 : 원을 360으로 나눠 표현하는 방법

단위 : 도 °

호도법 : 길이 비율에 따라 각도를 표현하는 방법

$\theta : 360^\circ = r : 2\pi r$

단위 : 라디안 rad

$1^\circ = \pi / 180$ rad

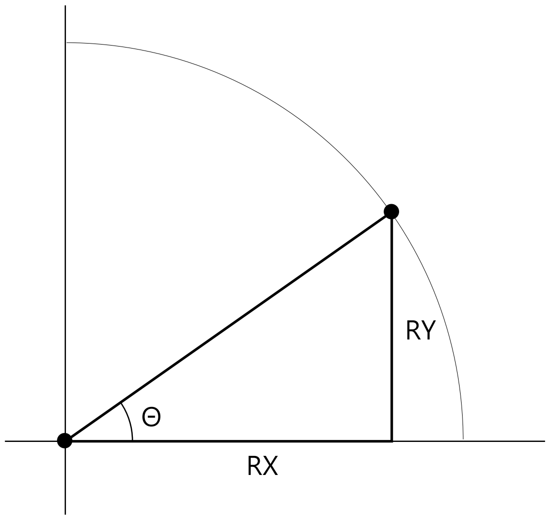
$180^\circ = \pi$ rad

$360^\circ = 2\pi$ rad

선분의 각도 계산

두 점의 좌표가 주어 졌을 때 상대적인 각도를 측정하는 방법이다.

두 점의 가로 세로 길이를 알기 때문에 각도를 도출해 낼 수 있다.



tan Θ = Ry / Rx

Θ = tan ^ -1 (Ry / Rx)

Θ = atan(Ry / Rx)

Θ = atan2(Ry , Rx)

결론 : 중심각은 역탄젠트를 활용하여 구한다.

atan(Ry / Rx) // -π/2 ~ π/2

atan2(Ry , Rx) // -π ~ π

mod(atan2(Ry, Rx) + M_2_PI, M_2_PI); // 0 ~ 2π

벡터

벡터 : 크기와 방향을 가지는 물리량 (a 또는 Va 와 같이 표기)

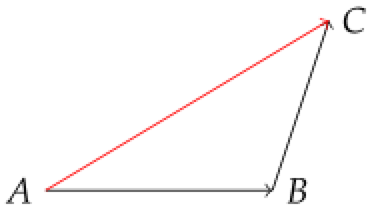
스칼라 : 크기만 가지고 있는 물리량 (|a| 또는 |Va| 와 같이 표기)

단위 벡터 : 크기가 1인 벡터

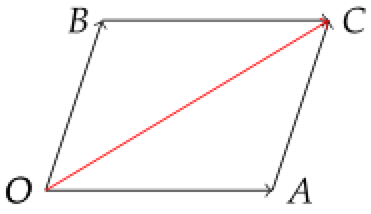
역 벡터 : 벡터에 대해 방향이 반대이고 길이 또는 크기가 같은 벡터

Va (x1, y1), Vb(x2, y2)

덧셈	Va + Vb	= (x1+x2 , y1+y2)
곱셈	Va * n	= (x1 * n , y1 * n)



$\overrightarrow{AB} + \overrightarrow{BC} = \overrightarrow{AC}$



$\overrightarrow{OA} + \overrightarrow{OB} = \overrightarrow{OC}$

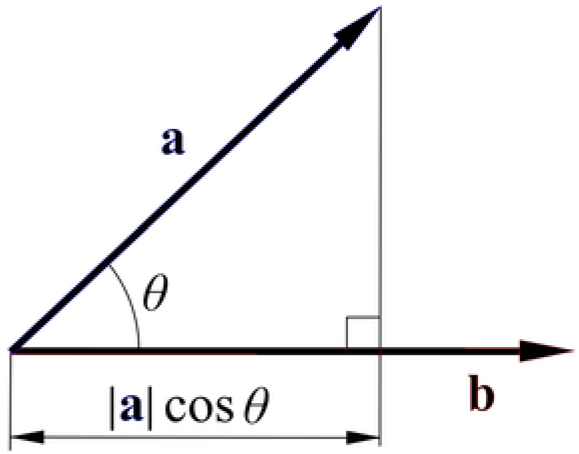
사용처 : 물체의 운동 방향, 사이각 계산

벡터 내적

벡터에서 방향이 일치하는 만큼만 곱하는 방식이다.

벡터를 곱하는 방법 중 하나이다.

결과값은 스칼라 값이 나온다.



Va · Vb = x1*x2 + y1 * y2 = |Va| * |Vb| * cosΘ

벡터 내적

벡터의 사이각 구하기

angle = acos((a · b) / (|a| * |b|)) // 값이 두 경우로 나온다.

angle = min(angle, M_PI - angle) // 두 값 중 작은 값으로 나온다.

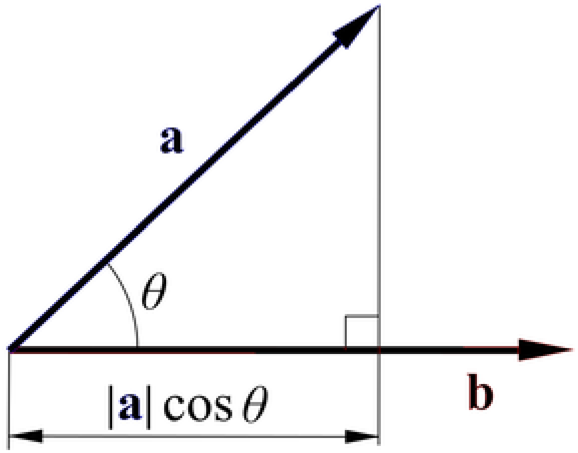
벡터의 직각 여부 확인하기.

두 벡터의 내적이 0이라면 항상 직각이다.

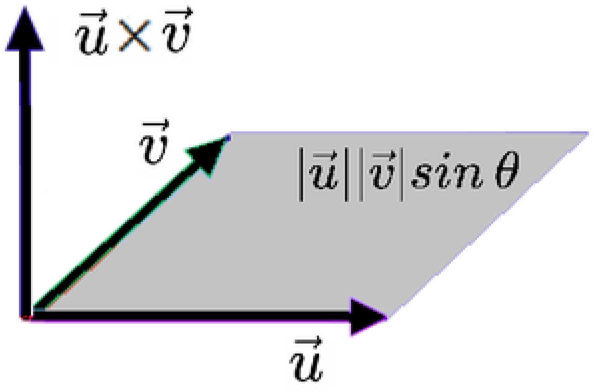
a · b 가 0이라면 항상 직각이다.

벡터의 사영

|a| cos Θ = (a · b) / |b|



벡터 외적



두 벡터가 이루는 정사각형의 넓이를 구하는 방법이다.

벡터를 곱하는 방법 중 하나이다.

결과값은 벡터이다.

3차원 벡터에서 정의된다. (z축을 0으로 계산하여 2차원에서 사용한다.)

오른손 법칙을 따른다.

벡터 외적의 결과값은 a벡터와 b벡터가 만드는 평행 사변형의 넓이이다.

Va * Vb = ((y1 * z2 – z1 * y2), (z1*x2 – x1 * z2), (x1 * y2 - y1 * x2))

= |Va| * |Vb| * sin Θ

2차원의 결과값 = (0, 0, (x1 * y2 - y1 * x2))

외적 값을 절반으로 나누면 두 벡터가 만드는 삼각형의 크기를 계산 할 수 있다.

벡터 외적

면적 계산

외적의 절대값은 Va, Vb를 두 변으로 하는 평행 사변형의 넓이이다.

따라서 외적값을 절반으로 나누면 두 벡터를 선분으로 하는 삼각형의 넓이가 나온다.

삼각형의 넓이 = |Va * Vb| / 2

두 벡터의 방향 판별

외적 결과값의 부호에 따라 방향성을 알 수 있다.

오른손 법칙에 따라

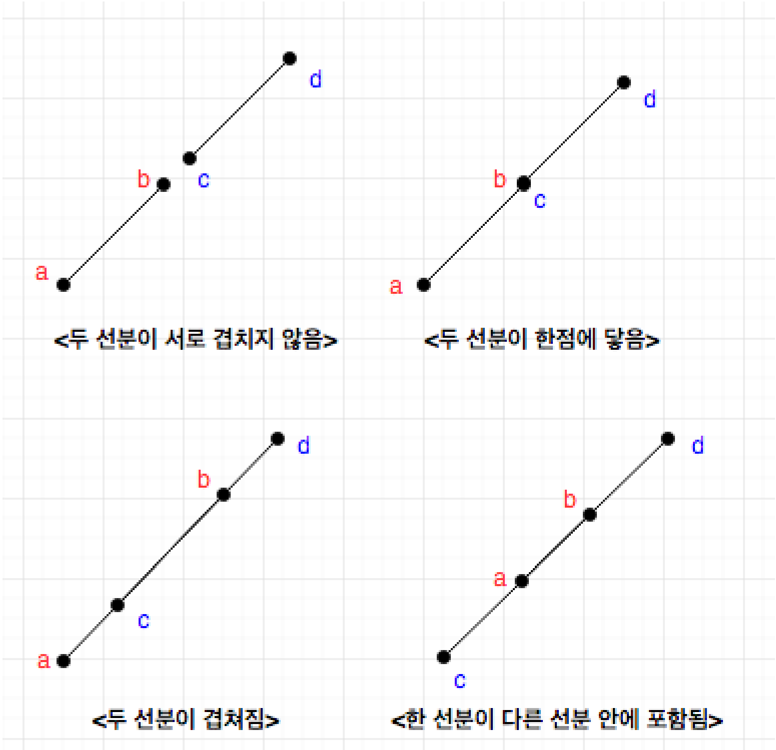
양수 : Vb가 Va의 반시계 방향에 있음

음수 : Vb가 Va의 시계 방향에 있음

선분과 선분의 교차

한 직선에 평행한 두 선분이 있을 때 선분들의 관계는 넷 중 하나이다.

1. 서로 겹치지 않음
2. 한 점에서 닿음
3. 겹쳐짐
4. 한 선분이 다른 선분 안에 포함 됨



선분 AB에 대하여 선분 CD의 교차관계

A < C < B 또는 A < D < B 라면 교차한다.

단 A < B 이고, C < D 이며, CCW(A, B, C) != 0 일 때

A < B 이고 C < D 일 때 다음 경우를 먼저 걸러야 한다.

```
if(ccw(a, b, c) != 0 || b < c || d < a) return false;
```

polar 메소드도 존재 하지만 시간이 오래걸린다.

행렬

수 또는 다항식 등을 직사각형 모양으로 배열한 것

행 : 가로줄 Row

열 : 세로줄 Column

차원 : 행렬의 크기로 행의 개수 * 열의 개수로 나타냄 m*n차원 행렬

행벡터 : 행렬이 하나의 행으로 구성되어 있는 경우

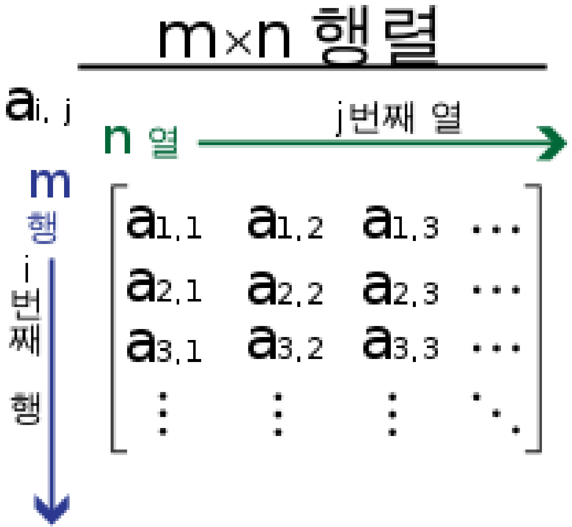
열벡터 : 행렬이 하나의 열로 구성되어있는 경우

상등 : 두 행렬의 모든 원소가 같다면 상등한다 라고 표현한다.

영행렬 : 행렬의 모든 원소가 0으로 이뤄진 경우

전치행렬 : 원래의 행렬의 행과 열을 바꾼 행렬

사용처 : 방정식 계산 , 모델링(현상 수식화), 암호화, 그래픽 처리(이미지 변환)



행렬

행렬 A = (a1 a2)
(a3 a4)

행렬 B = (b1 b2)
(b3 b4)

덧셈 A + B = (a1+b1 a2+b2)
(a3+b3 a4+b4)

곱셈 K * A = (K*a1 K*a2)
(K*a3 K*a4)

행렬의 곱

행렬 곱

행렬을 곱하기위해선 A행렬의 열의 수와 B 행렬의 행의 수가 같을 때 행렬 곱 AB가 정의 될 수 있다.

교환 법칙	: 다음 두 경우를 제외하고 일반적으로 성립하지 않음
	단위행렬과의 교환 법칙
	스칼라 곱의 교환 법칙
결합법칙	: 성립함
분배법칙	: 성립합

A = (a1 a2) B = (b1)
(b2)

AB = (a1*b1 a2*b2)

A = (a1 a2)
(a3 a4)

B = (b1 b2)
(b3 b4)

AB = (a1*b1+a2*b3 a1*b2+a2*b4)
(a3*b1+a4+b3 a3*b2+a4*b4)

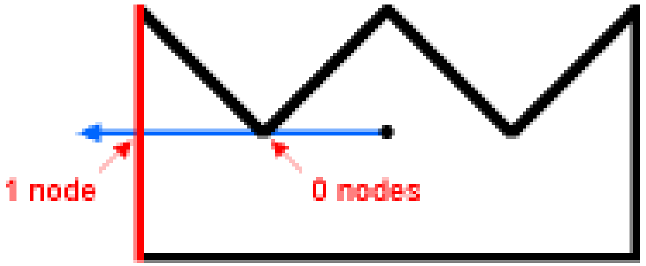
목차

점과 다각형의 상대 위치 검사

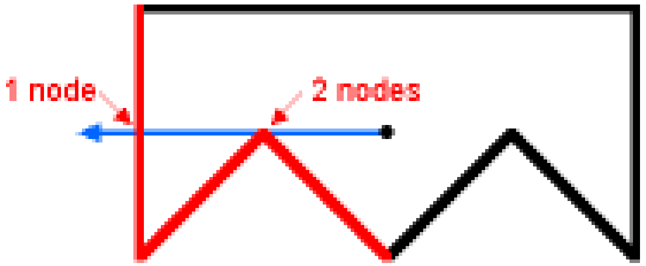
다각형과 한 점이 주어질 때 점이 다각형의 내부에 있는지 외부에 있는지 확인하는 방법

점에 한 방향으로 반직선을 긋는다.
해당 반직선이 다각형과 만나는 점의 개수에 따라 결과를 판별할 수 있다.

홀수 : 다각형의 내부에 있는 점이다.
0 또는 짝수 : 다각형의 외부에 있는 점이다.



점으로 판단하는 것 이 아닌 선분으로 판단 하기 때문에
다음과 같은 경우에도 올바르게 판별이 가능하다.



양쪽으로 판별하여 양쪽 모두 0 또는 짝수인 경우에만 외부에 있는 점이다.

CCW (Counter ClockWise)

평면 위에 놓여진 세 점의 방향 관계를 구하는 알고리즘

벡터의 외적을 이용하여 세 점의 방향성을 구한다.

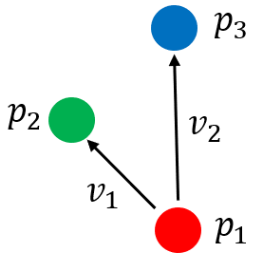
A점을 기준으로 두고 AB 벡터 * AC 벡터 를 구한다.

결과값에 따른 방향성

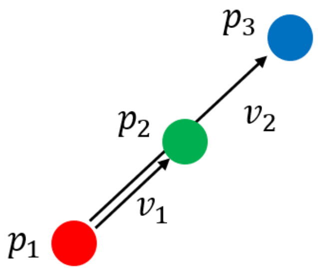
CCW() < 0 : 시계방향

CCW() = 0 : 일직선

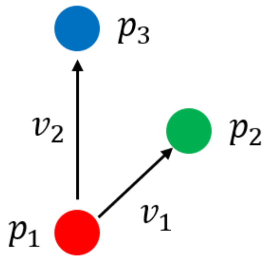
CCW() > 0 : 반시계 방향



시계 방향 (음수)



일직선 (0)



반시계방향 (양수)

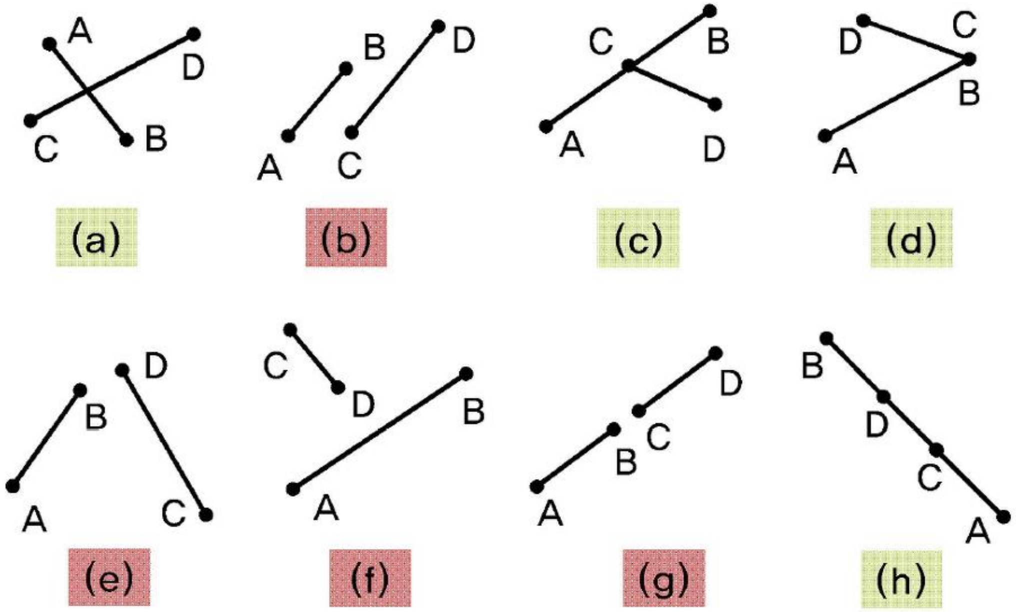
CCW 코드

```
// p1 p2 벡터 * p1 p2 벡터
// x1y2 + x2y3 + x3y1 - (x2y1 + x3y2 + x1y3)

int ccw(pair<int, int> p1, pair<int, int> p2, pair<int, int> p3){
        int op   =   p1.first * p2.second + p2.first * p3.second + p3.first * p1.second;
        op       -= (p1.second * p2.first + p2.second * p3.first + p3.second * p1.first);

        if ( op > 0 )            return 1;
        else if ( op == 0 )      return 0;
        else                     return -1;
}
```

두 선분의 교차 검사

A,B,C,D 점 에서 선분 AB , 선분 CD가 주어질 때 두 점이 교차하는지를 판단하는 방법이다.
CCW 알고리즘을 사용하여 이를 판단한다.

선분 AB에 대하여 C, D 점이 각각 좌측 우측에 하나씩 있어야 한다.
따라서 CCW 값을 곱하여 음수가 되면 교차한다 할 수 있다.
위의 조건을 만족하더라도 선분 CD에 대하여 점A, 점B가 만족하지 않는 경우가 있다.
따라서 선분 CD에 대하여 A, B점을 한번 더 검사한다.
이 값이 각각 음수라면 두 선분은 교차한다.



두 선분의 교차 검사

```
bool is_cross(point A, point B, point C, point D){

        int ta =   ccw ( A, B, C ) * ccw ( A, B, D );
                   // 선분 AB에 대하여 점 C, 점 D 가 각각 좌 우측에 배치되어 있는지 확인
        int tb =   ccw ( C, D, A ) * ccw ( C, D, B );
                   // 선분 CD에 대하여 점 A, 점 B 가 각각 좌 우측에 배치되어 있는지 확인

        return (ta < 0 && tb < 0 )
                   // 두 결과값이 모두 음수라면 서로 교차한다.
}
```

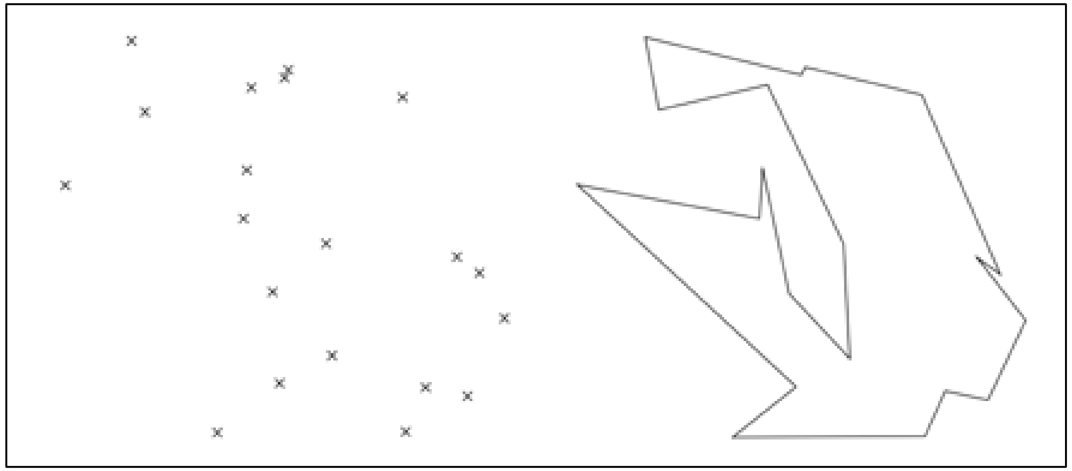
단순 폐쇄 경로 찾기

N개의 점이 주어졌을 때 이 점들을 모두 경유하고 출발점에 되돌아오는 교차하지 않는 경로를 찾는 알고리즘

단순 다각형 : 연속한 두 변 이외에는 어느 두 변도 교차하지 않는 다각형
경계가 스스로 교차하지 않는 다각형

단순 폐쇄 경로에 의해 만들어지는 다각형은 단순 다각형이다.

1. 임의의 기준점을 잡는다.
2. 각각의 점까지의 각도를 구한다.
3. 오름차순으로 정렬한다.
4. 각 점을 오름차순으로 잇는다.



기준점에 따라 다각형의 모양이 달라진다.

예시 문제 : 단순 다각형 https://www.acmicpc.net/problem/3679

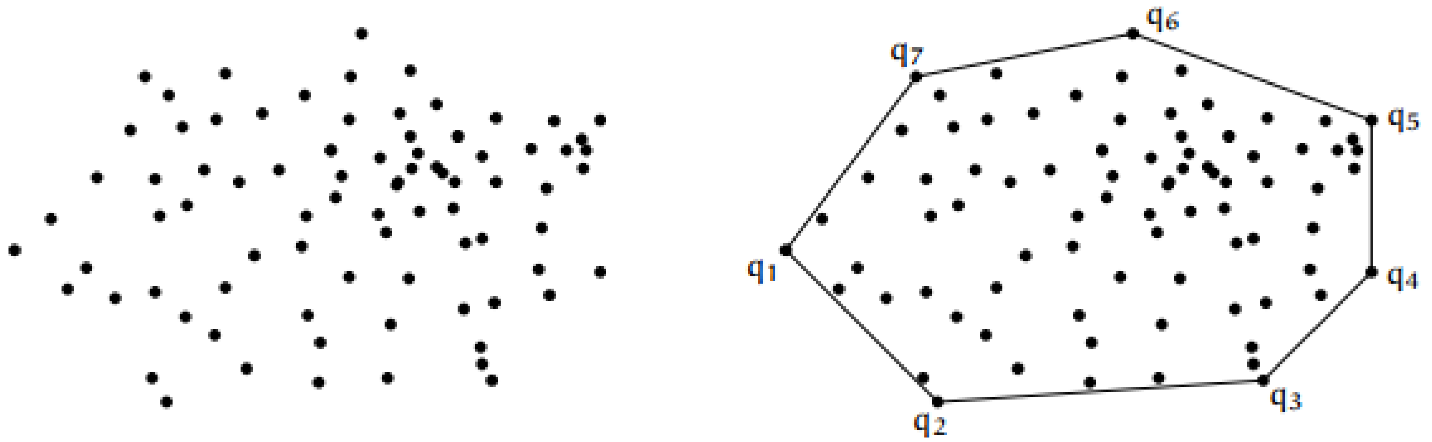
블록 껍질 찾기

유한한 점의 집합 X에 대해 그 점을 모두 포함하는 가장 작은 볼록 다각형을 볼록 껍질 이라 한다.

해당 문제를 풀 수 있는 알고리즘으로는

선물 포장 알고리즘, 그레이엄 스캔, 퀵 헐, 분할 정복, 모노톤 체인, 점진적 볼록 껍질 알고리즘 등이 있다.

알고리즘에 따라 복잡도가 크게 달라진다.



(a) Input. (b) Output.

짐꾸러기 알고리즘

선물 포장 알고리즘, 자비스 행진 알고리즘 이라고도 한다. https://www.youtube.com/watch?v=ZnTiWcIznEQ

CCW를 사용한다.

O(nh) (h : 블록껍질을 이루는 점의 개수) 입력값의 영향을 많이받는 알고리즘이다.

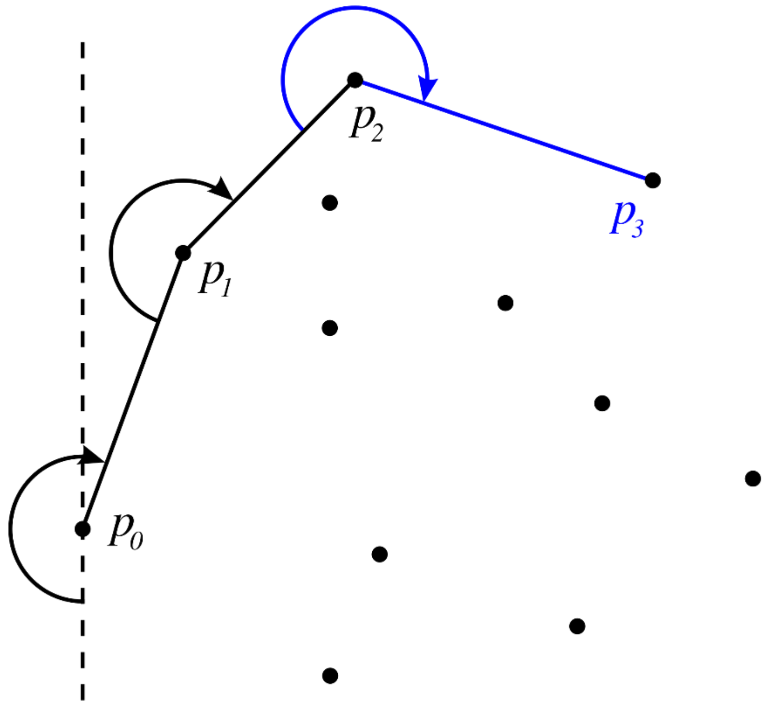
기준점에 대하여 모든 점의 각도를 계산하여 가장 큰 각도를 갖고 있는 점을 결과에 집어넣는 알고리즘이다 .

기준점을 잡는다.

기준점으로부터 모든 점에 대한 각도를 조사한다.

가장 큰 또는 가장 작은 값을 결과에 집어 넣는다.

다음 점이 기준점 이면 알고리즘을 종료한다.



그레이엄 스캔 (그라함 스캔)

O(N log N)의 시간 복잡도를 갖는 볼록 껍질 알고리즘이다.

CCW, 스택을 이용한다.

https://www.youtube.com/watch?v=Ps1idzOx6LA

x값 또는 y값이 가장 작은 점을 기준으로 잡는다.

기준점에서 모든 점에 대한 각도를 측정한다.

각도를 기준으로 오름차순 정렬한다.(반 시계방향으로 선을 정렬)

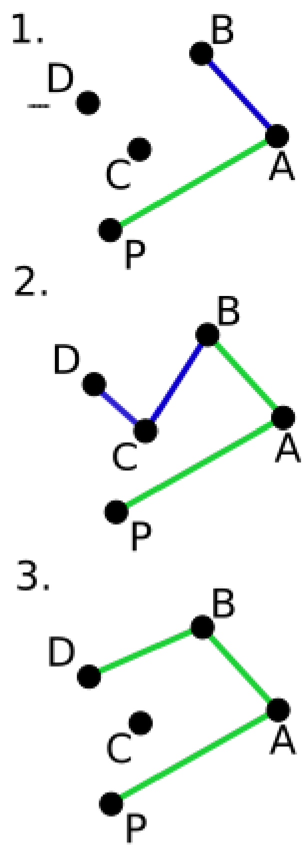
0번 점과 1번 점을 스택에 넣는다.

가장 최근의 두 점을 기준으로 다음 점이 선분의 좌측인지 우측인지 판별한다.

좌측 또는 선분 위에 있는 경우 : 스택에 해당 점을 집어 넣는다.

우측에 있는 경우 : 스택에서 pop 연산을 한다.

모든 점을 판별한 경우 알고리즘이 종료된다.



참조

벡터

- https://owlyr.tistory.com/8
- https://hellogaon.tistory.com/37
- https://nsgg.tistory.com/85
- https://devhwan.tech/53

IBM math.h 문서

- https://www.ibm.com/docs/ko/i/7.4?topic=files-mathh

동명대학교 조미경 교수님의 ppt

- http://cfs12.tistory.com/original/24/tistory/2008/10/13/19/53/48f328b5cb9d0

삼각함수의 실생활 응용 사례

- https://easytoread.tistory.com/entry/%EC%82%BC%EA%B0%81%ED%95%A8%EC%88%98%EC%9D%98-%EC%8B%A4%EC%83%9D%ED%99%9C-%ED%99%9C%EC%9A%A9-%EC%82%AC%EB%A1%80#GPS%EC%9D%98_%EC%9C%84%EC%B9%98_%EA%B3%84%EC%82%B0