기하 알고리즘

기하요소의 표현

- 기하학의 기본 요소로는 점, 선, 다각형, 다면체 등의 1,2,3차원의 물체가 있음
- 간단한 기하문제 예시. 두 선이 교차하는가? 주어진 점이 다각형의 내부에 있는가? 주어진 삼각형이 직각 삼각형인가 등이 있음
- 이러한 기하 문제를 푸는 알고리즘들을 기하 알고리즘(geometric algorithm)이라고 칭함.

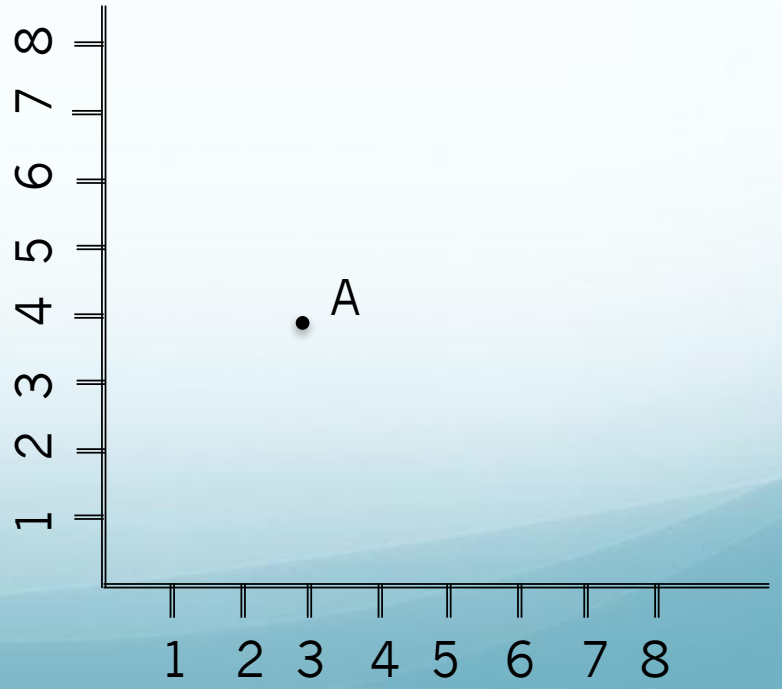
기하요소의 표현

- 점(point)

```
Struct  point{
    int x;                    //점의 x좌표
    int y;                    //점의 y좌표
} FirstPoint, SecondPoint;//point형의 변수들
```

점 A

```
FirstPoint.x = 3;
FirstPoint.y = 4;
```



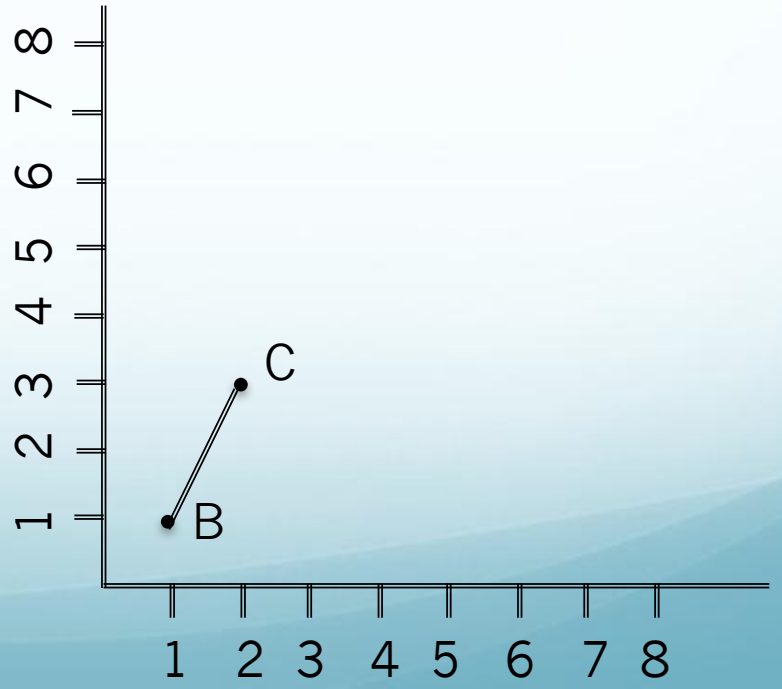
기하요소의 표현

- 선(line)

```
Struct  line{
    struct point pt1;
    struct point pt2;
} TestLine;
```

선분 BC

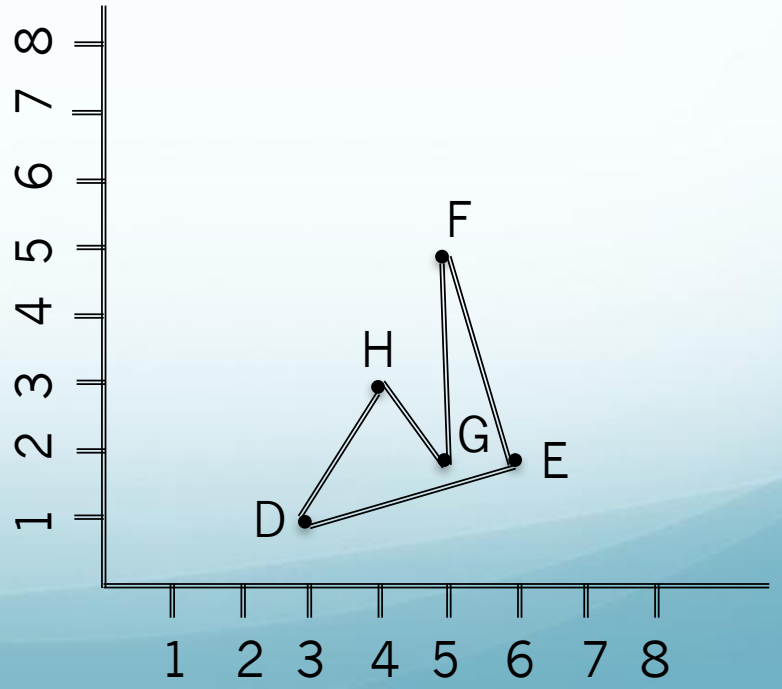
```
TestLine.pt1.x=1;
TestLine.pt1.y=1;
TestLine.pt2.x=1;
TestLine.pt2.y=1;
```



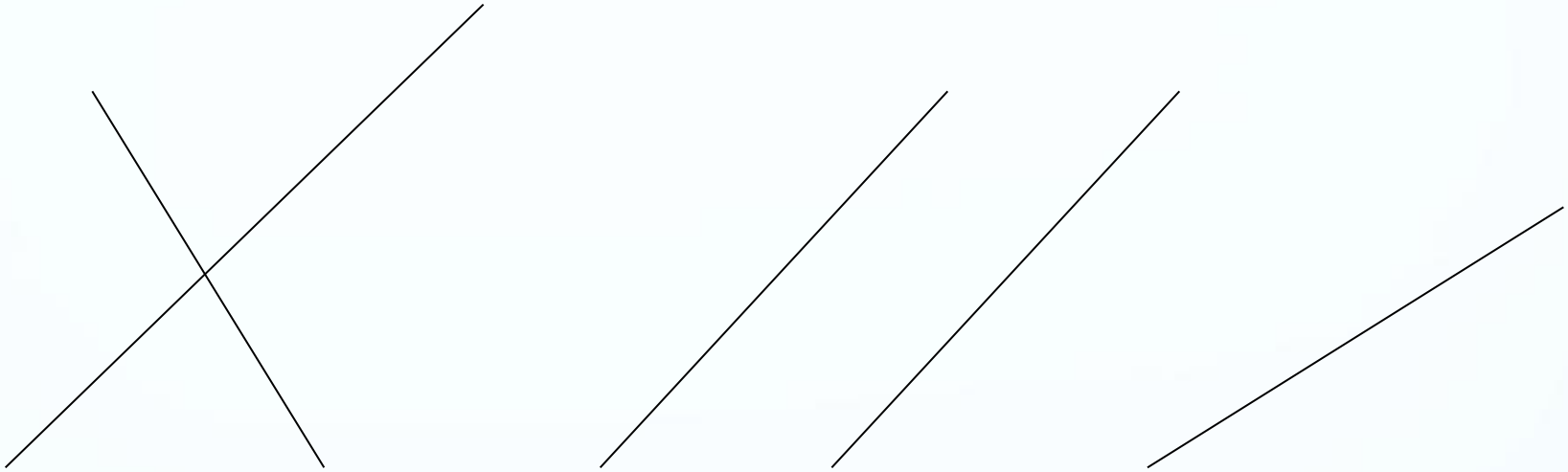
기하요소의 표현

- 다각형: struct point Polygon[n];
- 다각형의 한 꼭지점을 기준점으로 하여 그 점으로부터 다각형의 둘레를 한 방향으로 따라가면서 만나는 점들을 Pgon[0:n-1]에 순서대로 저장.

Pgon[0] <-D
Pgon[1] <-E
Pgon[2] <-F
Pgon[3] <-G
Pgon[4] <-H

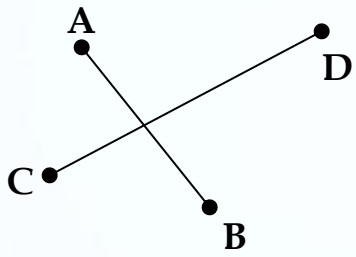


평면 위에 있는 두 직선의 상대 위치

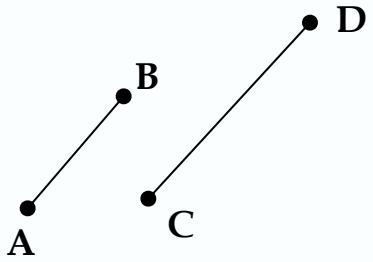


(a) 교차 (b) 평행 (c) 일치

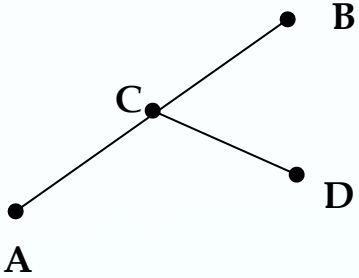
평면 위에 있는 두 직선의 상대 위치



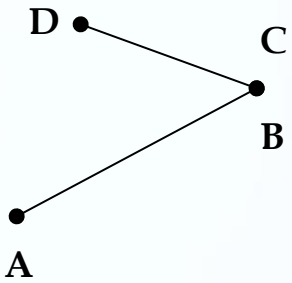
(a) 교차



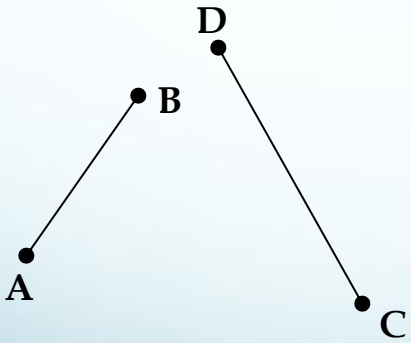
(b) 평행



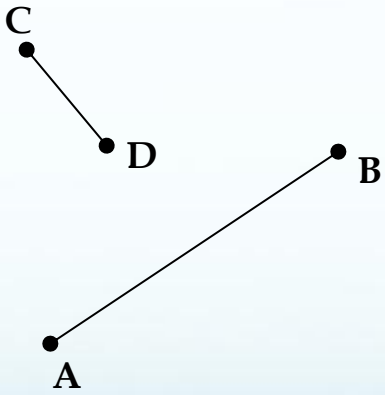
(c) 한끝점교차



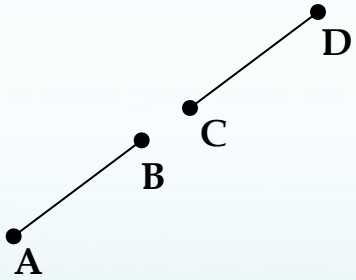
(d) 양끝점교차



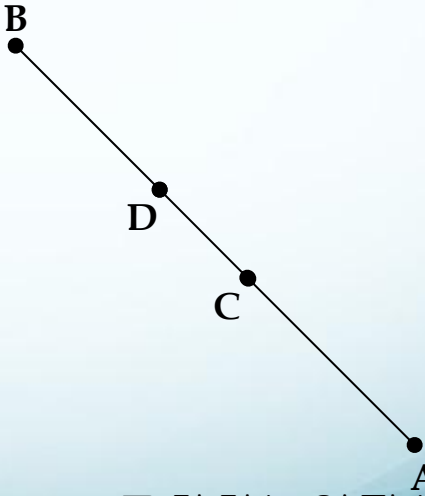
(e) 양연장선교차



(f) 한연장선교차

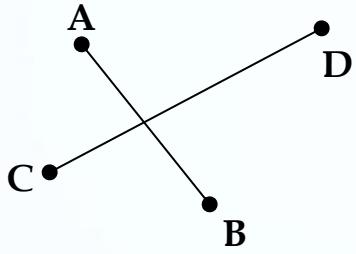


(g) 교차없는일직선
두 선분은 일직선상에
있으나 교차하지 않음

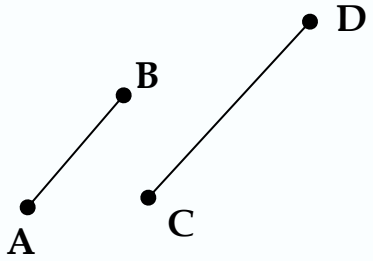


(h) 교차하는일직선
한 선분이 다른
선분을 포함 함

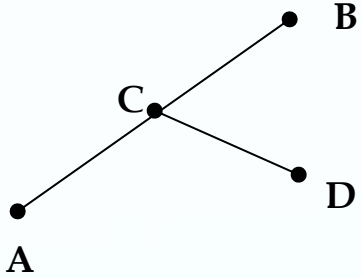
평면 위에 있는 두 직선의 상대 위치



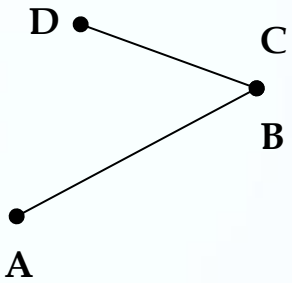
(a) 교차



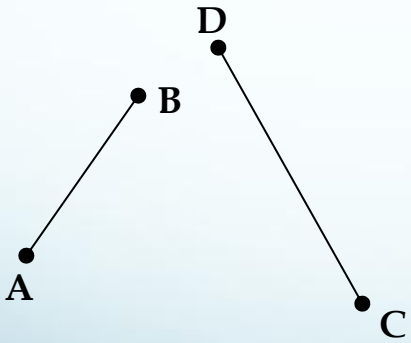
(b) 평행



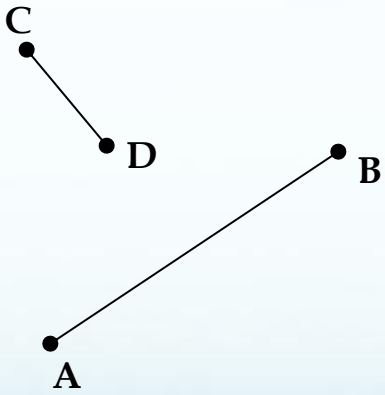
(c) 한끝점교차



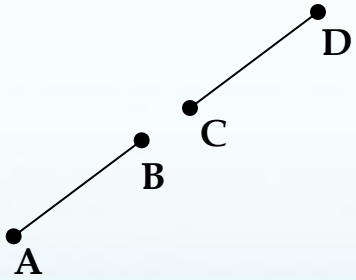
(d) 양끝점교차



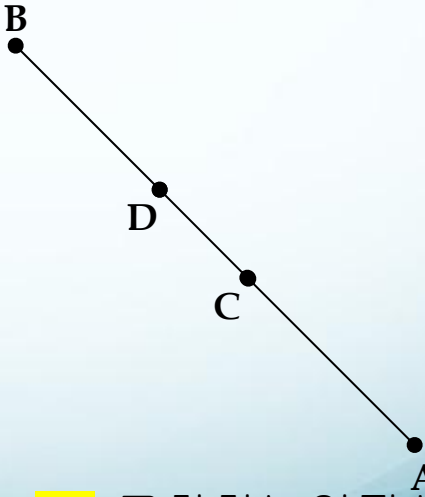
(e) 양연장선교차



(f) 한연장선교차



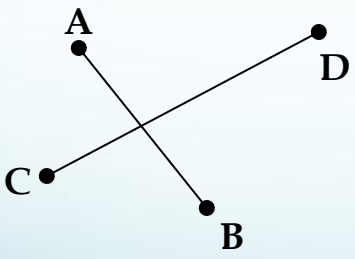
(g) 교차없는일직선
두 선분은 일직선상에 있으나 교차하지 않음



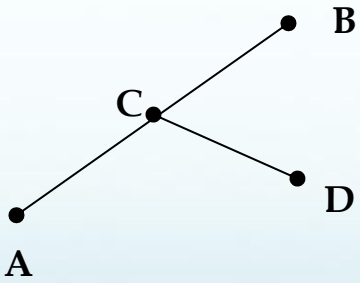
(h) 교차하는일직선
한 선분이 다른 선분을 포함 함

두 선분의 교차여부 문제

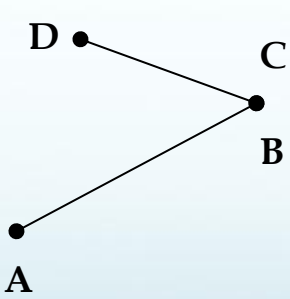
- (a) - 선분 AB를 기준으로 점 C와 D가 서로 반대편이고 선분 CD를 기준으로 점 A와 B가 서로 반대편에 있음
- (c), (d), (h) – 한 선분의 끝점이 다른 선분 상에 있음



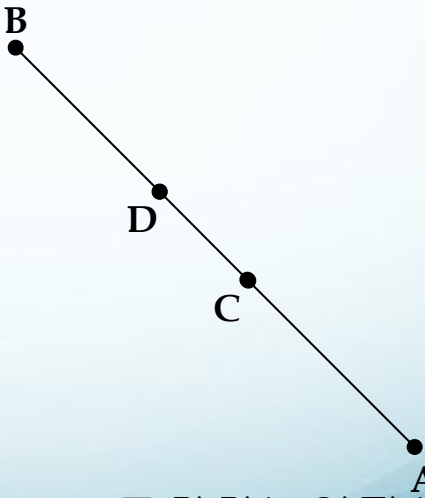
(a) 교차



(c) 한끝점교차



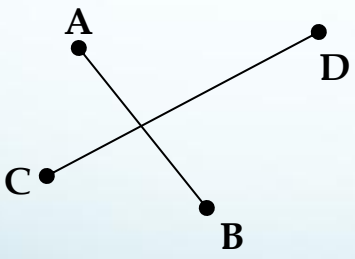
(d) 양끝점교차



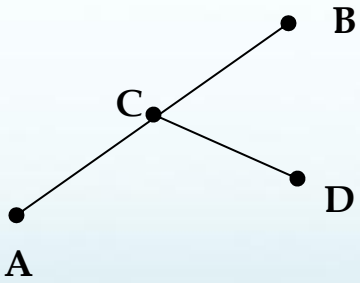
(h) 교차하는일직선 한 선분이 다른 선분을 포함 함

두 선분의 교차여부 문제

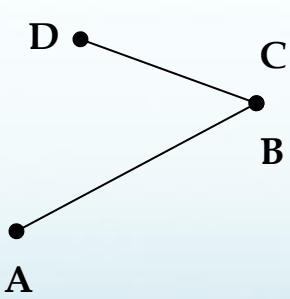
- 선분 AB를 기준으로 점 C와 D가 서로 반대편이고 &
  선분 CD를 기준으로 점 A와 B가 서로 반대편에 있음
- 한 선분의 끝점이 다른 선분 상에 있음



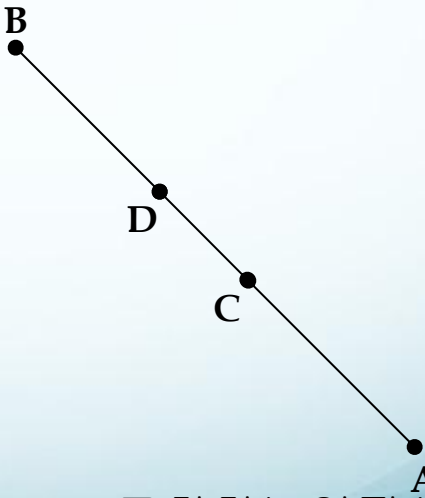
(a) 교차



(c) 한끝점교차



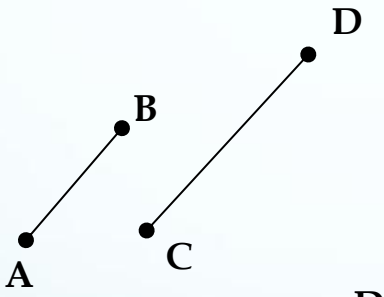
(d) 양끝점교차



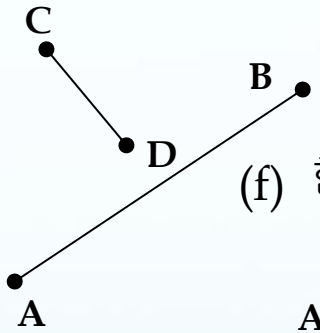
(h) 교차하는일직선
한 선분이 다른
선분을 포함 함

두 선분의 교차여부 문제

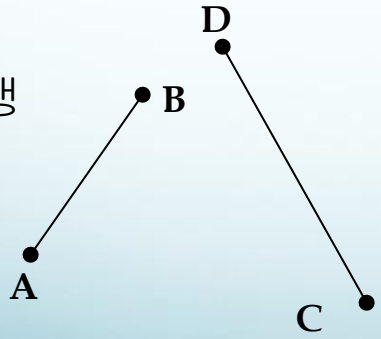
- 선분 AB를 기준으로 점 C와 D가 서로 반대편이고 &
  선분 CD를 기준으로 점 A와 B가 서로 반대편에 있음
- 한 선분의 끝점이 다른 선분 상에 있음



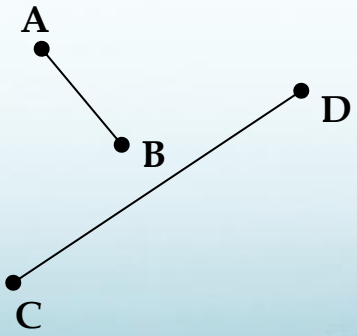
(b) 평행



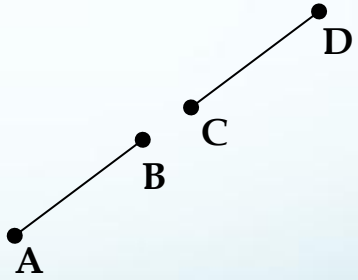
(f) 한연장선교차



(e) 양연장선교차



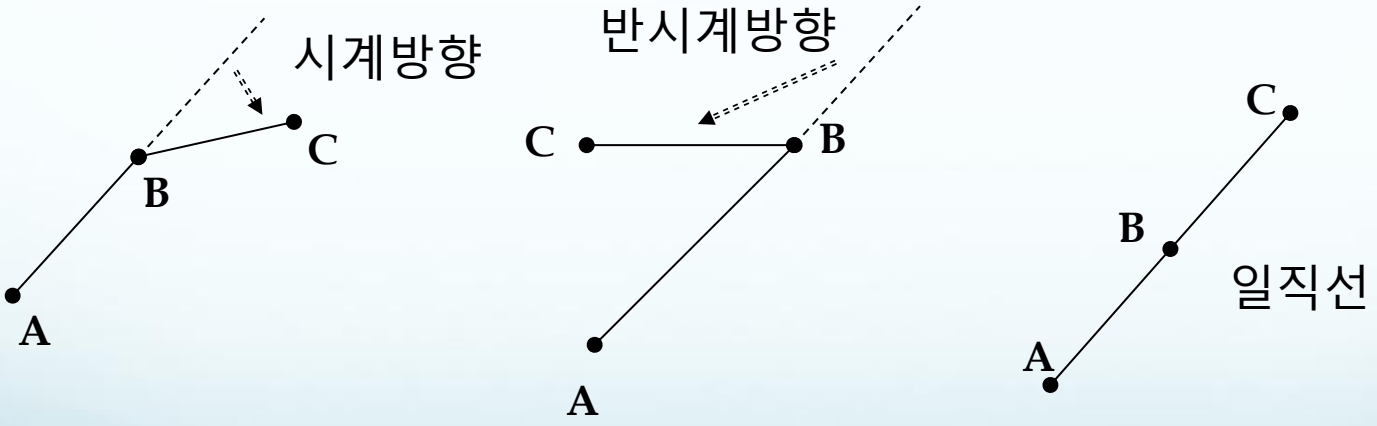
(f) 한연장선교차



(g) 교차없는일직선
두 선분은 일직선상에
있으나 교차하지 않음

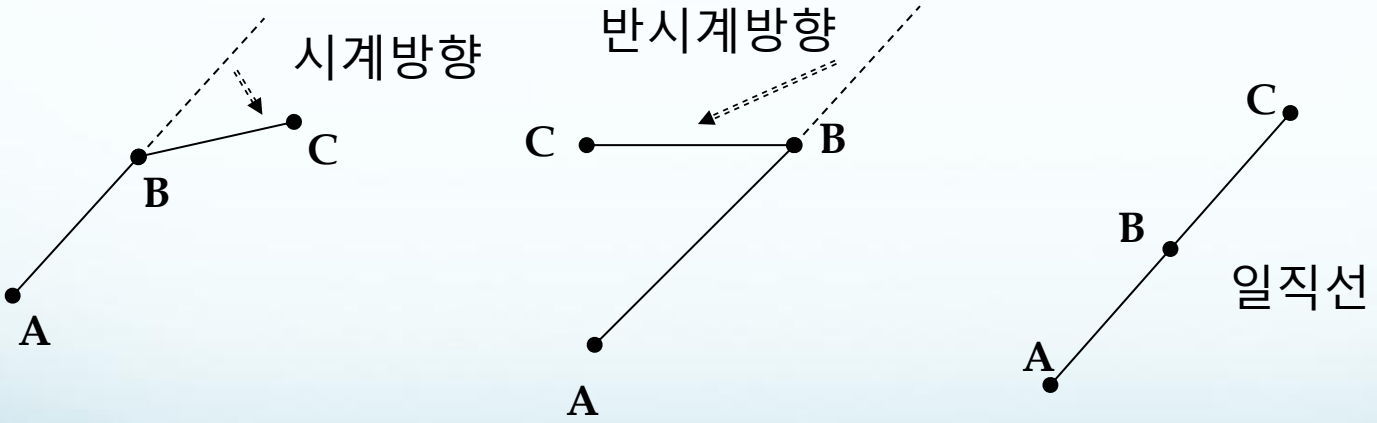
두 선분의 교차여부 문제

- 선분 AB와 두 점 C, D가 주어졌을 때, 이 두 점이 선분 AB를 기준으로 서로 반대편에 위치하고 있다는 것을 어떻게 알 수 있을까?
- ABC의 꺾이는 방향을 고려하자.



두 선분의 교차여부 문제

- 선분 AB와 두 점 C, D가 주어졌을 때, 이 두 점이 선분 AB를 기준으로 서로 반대편에 위치하고 있다는 것을 어떻게 알 수 있을까?
- ABC의 꺾이는 방향을 고려하자.



- ABC의 방향과 ABD의 방향이 서로 다르면 선분 AB를 기준으로 C와 D의 방향이 반대임.

두 선분의 교차여부 문제

- 세 점 A,B,C가 주어졌을 때 꺾은 선 ABC의 방향을 결정하는 함수 Direction(A,B,C)

Direction(A,B,C)

(1) A, B, C가 일직선상에 있지 않을 때, 꺾은선 ABC의 방향이

① 시계 방향인 경우 : 1

② 시계 반대 방향인 경우 : -1

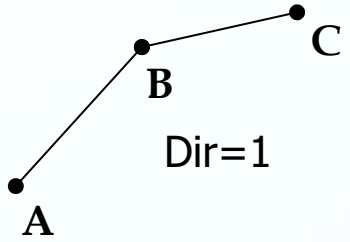
(2) 점 A, B, C가 일직선상에 있을 때,

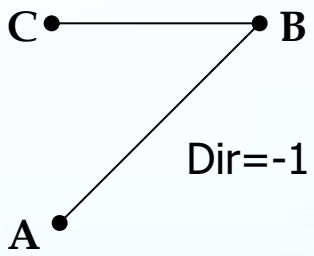
① C가 가운데 있는 경우 : 0

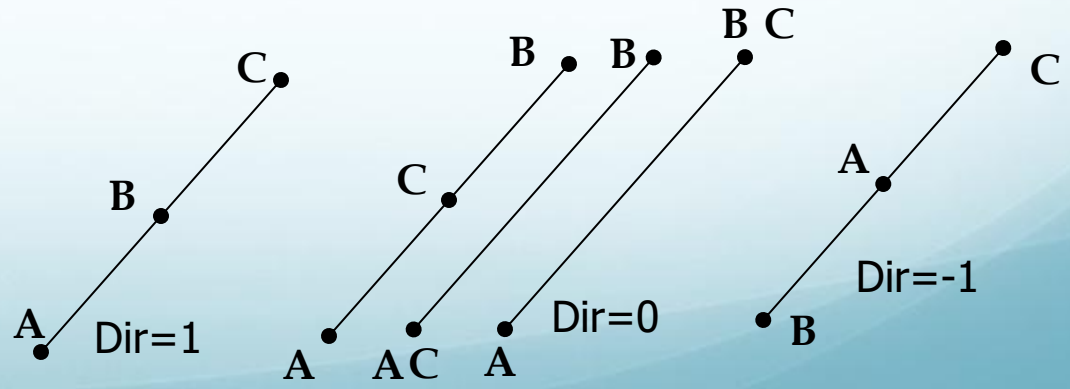
C=A 또는 C=B인 경우 : 0

② B가 가운데 있는 경우 : 1

③ A가 가운데 있는 경우 : -1







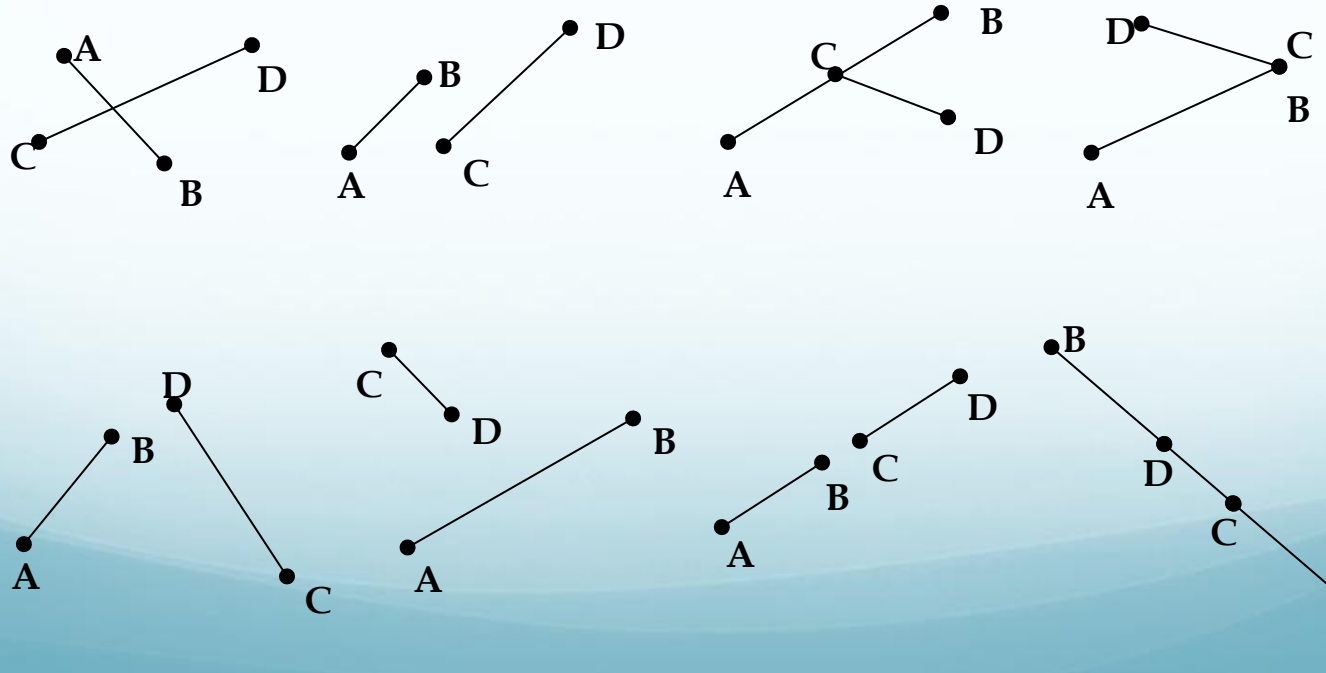
<함수 Direction이 복귀시키는 값>

Intersection (line AB, line CD)

A≠B and C≠D 이면 다음 두 조건이 만족될 때 두 선분이 교차한다.

(1) Direction(A,B,C) x Direction(A,B,D) ≤ 0

(2) Direction(C,D,A) x Direction(C,D,B) ≤ 0



int Intersection(struct line AB, struct line CD)

```
{
   /* 입력: AB, CD: 교차 상태를 검사할 두 선분,
      출력: AB와 CD가 서로 교차하면 true 아니면 false. */
1    int LineCrossing;
2    if ((Direction(AB.A, AB.B, CD.C) *
3      Direction(AB.A, AB.B, CD.D) < = 0) &&
4      (Direction(CD.C, CD.D, AB.A) *
5      Direction(CD.C, CD.D, AB.B) < = 0))
6      LineCrossing = TRUE;
7    else LineCrossing = FALSE;
8    return LineCrossing;
}
```

Direction(A,B,C)

반시계방향 AB기울기<AC기울기

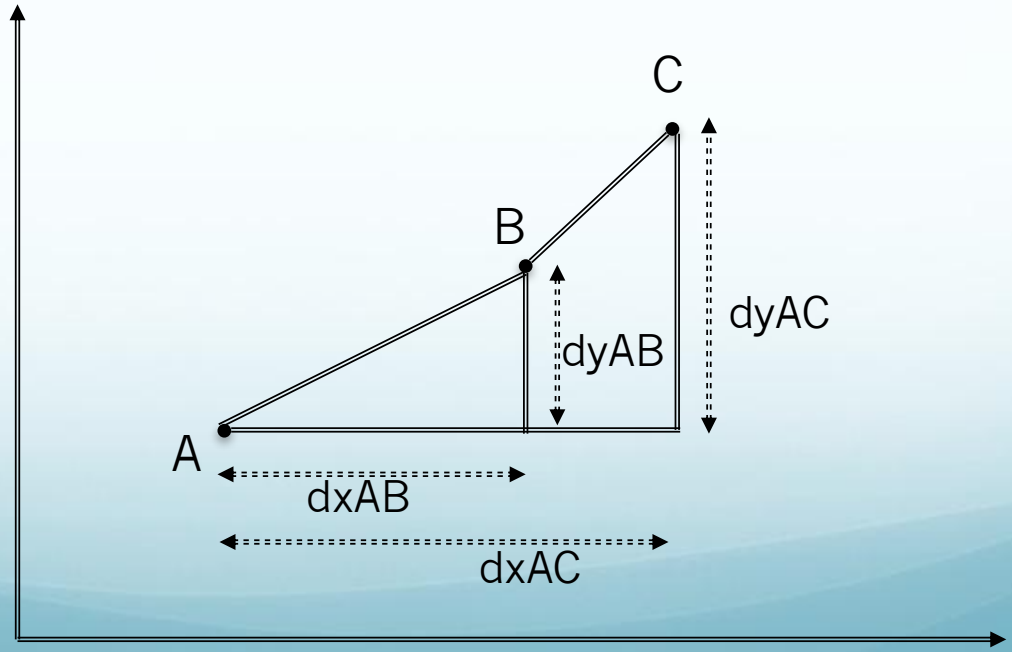
$$\frac{dyAB}{dxAB} < \frac{dyAC}{dxAC}$$

$$dyABdxAC < dyACdxAB$$

시계방향 AB기울기>AC기울기

$$\frac{dyAB}{dxAB} > \frac{dyAC}{dxAC}$$

$$dyABdxAC > dyACdxAB$$



int Direction(struct point A, struct point B, struct point C)

```
{/* 입력 : A, B, C - 세 점의 좌표,  출력 : Dir - 그림 4에서 정의된 값 */
        int dxAB, dxAC, dyAB, dyAC, Dir;
 1      dxAB = B.x - A.x;  dyAB = B.y - A.y;
 3      dxAC = C.x - A.x;  dyAC = C.y - A.x;
 5      if (dxAB * dyAC < dyAB * dxAC) Dir = 1;       //AB기울기 > AC기울기
 6      if (dxAB * dyAC > dyAB * dxAC) Dir = -1;    //AB기울기 < AC기울기
 7      if (dxAB * dyAC == dyAB * dxAC) {           //AB기울기 = AC기울기
 8         if (dxAB == 0 && dyAB == 0) Dir = 0;
 9         if ((dxAB * dxAC < 0) || (dyAB * dyAC < 0)) Dir = -1;
10         else if ((dxAB * dxAB + dyAB * dyAB >=
                (dxAC * dxAC + dyAC * dyAC)  Dir = 0;
12         else    Dir = 1;
13      }
14      return Dir;
}
```