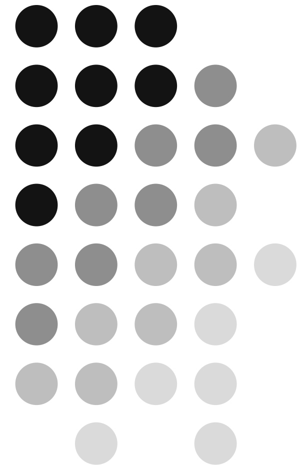
Partición de polígonos

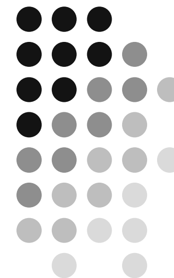
Diseño y Análisis de Algoritmos

Cátedra de Programación
Carrera de Ingeniería de Sistemas
Prof. Isabel Besembel Carrera



Todo el contenido está tomado de J. O'Rourke "Computational Geometry in C"

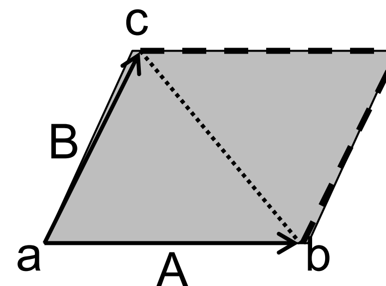
Área de un polígono

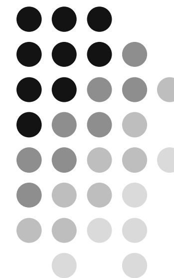


- Área de un triángulo A(T)= 1/2(base x altura)
- No es una fórmula útil si las coordenadas de sus vértices son cualesquiera.
- Producto vectorial de 2 vectores: A×B en 2D (xy) A×B= (AxBy-AyBx)k, es un vector perpendicular al plano del triángulo, A(T)=1/2 (AxBy-AyBx)

Se sustituye A=(b-a) y B=(c-a) y queda

2A(T) = (bx-ax)(cy-ay)-(cx-ax)(by-ay)



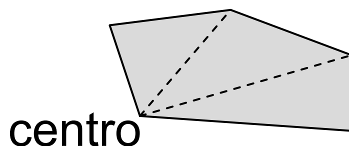


Área de un polígono

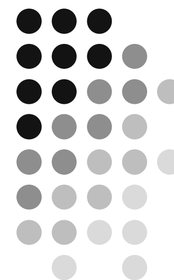
- Producto vectorial según coordenadas: fácilmente generalizable para d-dimensiones
- Lema área triángulo: Dos veces el área de un triángulo T=(a, b, c) está dada por

$$\begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} = (b_x - a_x)(c_y - a_y) - (c_x - a_x)(b_y - a_y) = 2A(T)$$

- Área de un polígono convexo: Cada polígono convexo puede ser triangulado en abanico, con todas sus diagonales incidiendo en un único vértice (centro del abanico)



Área de un cuadrilátero convexo



UNIVERSIDAD DE LOS ANDES

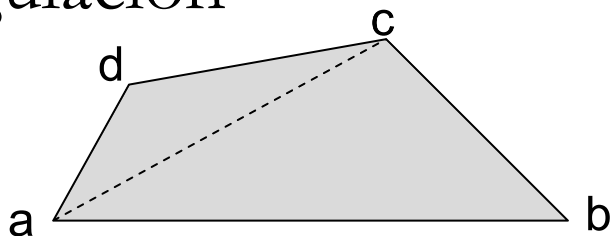
- Área de un polígono con vértices v0, v1, …, vn-1 etiquetados en sentido contrareloj se puede calcular

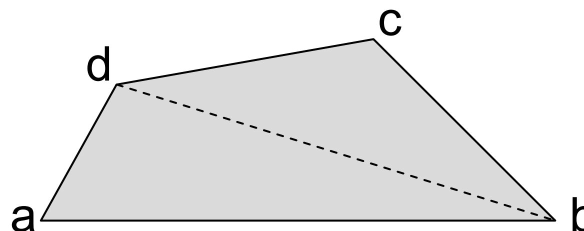
  A(P)=A(v0, v1, v2)+A(v0, v2, v3)+…+A(v0, vn-2, vn-1)

- Área de un cuadrilátero convexo Q=(a, b, c, d) es

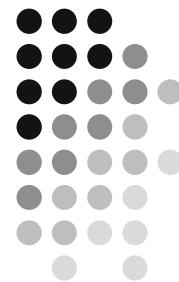
  A(Q)=A(a, b, c)+A(a, c, d)=A(d, a, b)+A(d, b, c)

- Se obtiene la misma expresión independiente de la triangulación





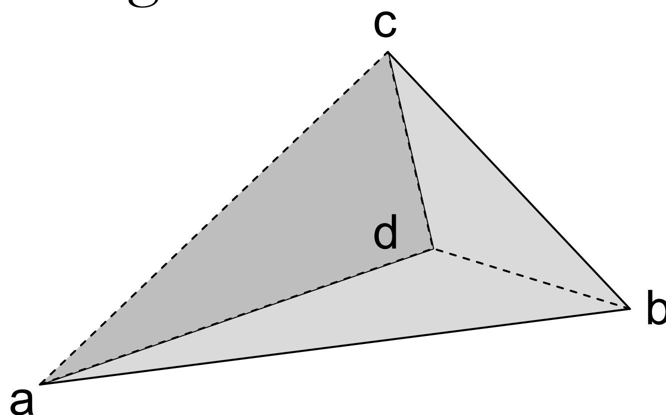
Área de un cuadrilátero no convexo



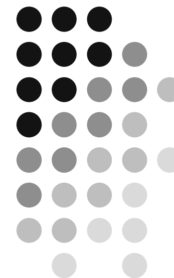
- Generalizando, vi tiene coordenadas xi yi

$$2A(P) = \sum_{i=0}^{n-1} (x_i y_{i+1} - y_i x_{i+1})$$

- Área de un cuadrilátero no convexo: A(Q) se mantiene, aunque la diagonal ac sea externa a Q, su interpretación es que A(a, c, d) es negativa

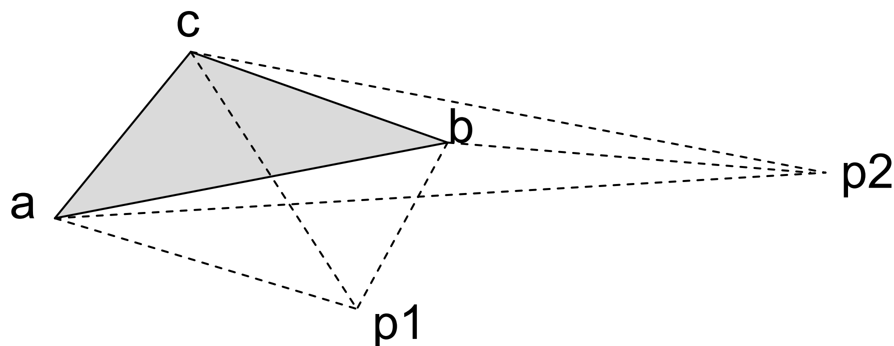


Área desde un centro arbitrario



- Generalización del método de suma de las áreas de los triángulos según un punto arbitrario, quizás en el exterior.
- Lema área T: Sea T=Δabc, con vértices orientados contrareloj y sea p cualquier punto en el plano

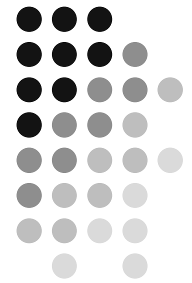
A(T)=A(p, a, b)+A(p, b, c)+A(p, c, a)



A(p2, a, b) y A(p2, b, c) son negativas porque sus vértices están orientados con el reloj

UNIVERSIDAD DE LOS ANDES

Área de un polígono



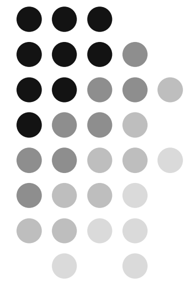
- Teorema área de un polígono: sea un polígono P (convexo o no) con v0, v1, …, vn-1 vértices etiquetados contrareloj y sea p cualquier punto en el plano. Se tiene

$$2A(P) = \sum_{i=0}^{n-1}(x_i y_{i+1} - y_i x_{i+1})$$

  - Prueba: por inducción sobre n. Caso n=3, ya establecido con el lema área T.
  - Suponga que A(P)=A(p, v0, v1)+A(p, v2, v3)+…+A(p, vn-2, vn-1)+A(p, vn-1, v0) es cierta ∀P’ con n-1 vértices y sea P un polígono con n vértices. Por el teorema dos orejas de Meisters, P tiene una oreja. Renumere los vértices de P y sea E=(vn-2, vn-1, v0) la oreja. Sea Pn-1 el polígono obtenido al remover E.

UNIVERSIDAD DE LOS ANDES

Teorema área del polígono



Por la hipótesis de inducción, A(Pn-1)=A(p, v0, v1)+…+ A(p, vn-3, vn-2)+A(p, vn-2, v0)

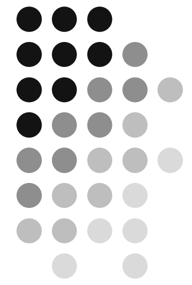
Por el lema área T, A(E)=A(p, vn-2, vn-1)+A(p, vn-1, v0)+ A(p, v0, vn-2)

Como A(P)=A(Pn-1)+A(E), se tiene que

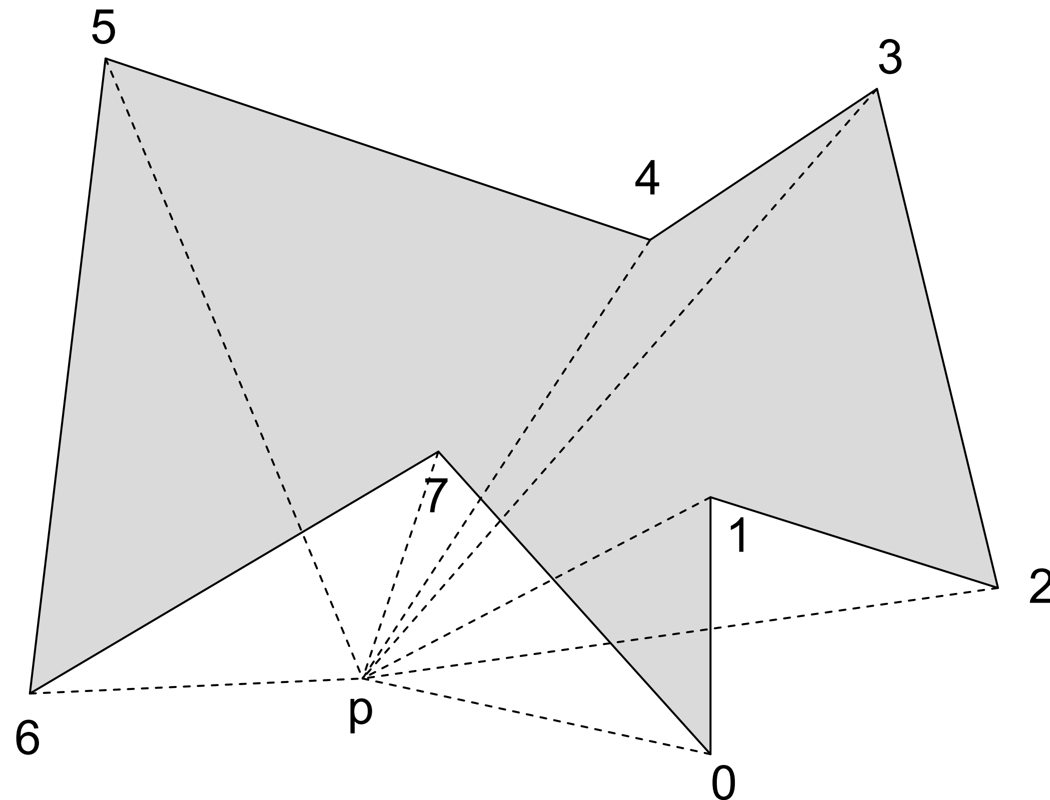
A(P)=A(p, v0, v1)+…+A(p, vn-3, vn-2)+A(p, vn-2, v0)+ A(p, vn-2, vn-1)+A(p, vn-1, v0)+A(p, v0, vn-2)

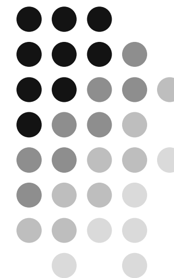
Note que A(p, v0, vn-2)=-A(p, vn-2, v0), se cancelan y se obtienen la ecuación original, la cual se expande y se cancelan los términos, lo que prueba el teorema

UNIVERSIDAD
DE LOS ANDES



Cálculo del área de P





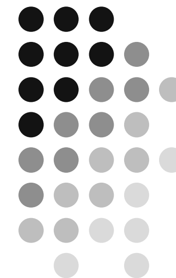
Volumen en d-dimensiones

- 3D: volumen de un tetraedro T de vértices a, b, c, d

$$6V(T)=\begin{vmatrix} a_x & a_y & a_z & 1 \\ b_x & b_y & b_z & 1 \\ c_x & c_y & c_z & 1 \\ d_x & d_y & d_z & 1 \end{vmatrix} = -(a_z-d_z)(b_y-d_y)(c_x-d_x)+\cdots+(a_x-d_x)(b_y-d_y)(c_z-d_z)$$

  Si (a, b, c) forman un circuito en el sentido contrareloj desde el punto d, entonces es positivo

- Por el teorema área de un polígono se generaliza el volumen de un poliedro por la suma (con signo) de los volúmenes de los tetraedros, cuyas caras triangulares están en el poliedro. Todas las caras deben estar orientadas contrareloj desde afuera
- Generalización a d-dimensiones: volumen de un hipertetraedro (tetraedro en dD)



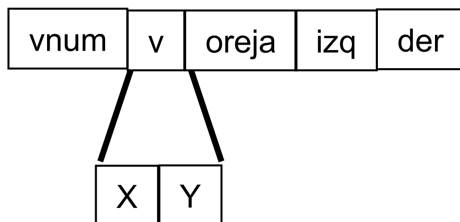
Implementación en 2D

- Punto

```
#define X 0
#define Y 0
typedef enum {FALSE, TRUE} bool;
#define DIM 2
typedef int TipoPuntoI[DIM];          // punto en 2D entero
```

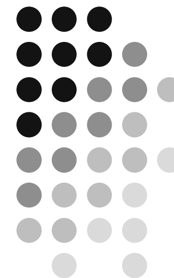
- Polígono

```
typedef struct TipoVertice tVer;
typedef tVer *ptrVer;
struct TipoVertice
{       int             vnum;     // subíndice
        TipoPuntoI      v;        // coordenadas
        bool            oreja;    // True si es una oreja
        ptrVer          izq, der;
};
ptrVer  vertices=NULL;            // cabeza de la lista circular
```



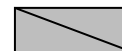
UNIVERSIDAD
DE LOS ANDES

Implementación en 2D



```
#define EXIT FALLA 1
char *malloc();
#define NEW(p, tipo)   \
     if ((p=(tipo *) malloc (sizeof(tipo))) == NULL) \
     {        printf("NEW:  Sin memoria!!!\n"); \
              exit (EXIT FALLA); \
     }
#define ADD(head, p)   \
     if (head)          \
     {        p->der = head; \
              p->izq = head->izq; \
              head->izq = p; \
              p->izq->der = p; \
     }\
     else \
     {        head = p; \
              head->der = head->izq = p; \
     }
#define FREE(p)  if(p)  {free ((char *) p); p = NULL; }
```

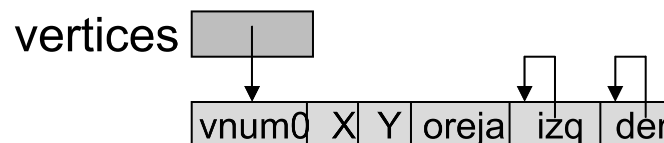
vertices



NEW(p, tVer)

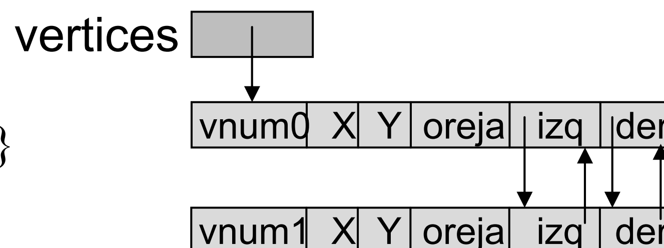
ADD(vertices, p)

vertices



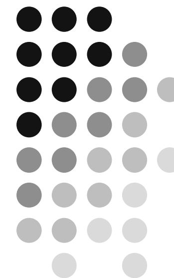
ADD(vertices, p)

vertices



UNIVERSIDAD
DE LOS ANDES

Área

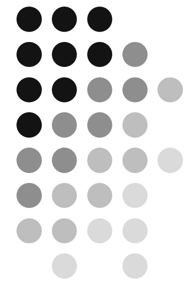


- Si las coordenadas son muy grandes, la multiplicación de ellas puede causar un desborde, que no se reporta en C

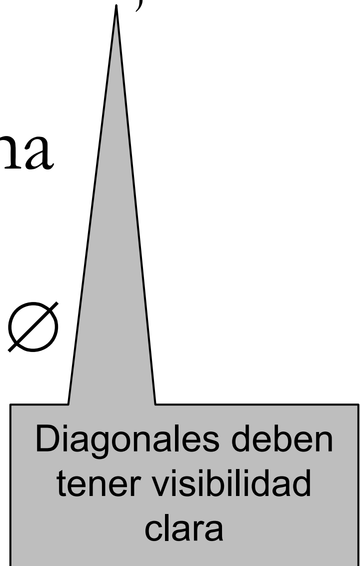
```
int  area2(TipoPuntoI a, TipoPuntoI b, TipoPuntoI c)
{    return (b[X]-a[X])*(c[Y]-a[Y])-(c[X]-a[X])*(b[Y]-a[Y]);
}
int  areaPoli2(void)
{    int      sum=0;
     ptrVer              p, a;
     p=vertices;
     a=p->der;
     do
     {        sum +=area2(p->v, a->v, a->der->v);
              a=a->der;
     }while(a->der != vertices);
     return sum;
}
```

UNIVERSIDAD DE LOS ANDES

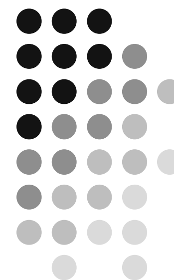
Diagonal de P



- Diagonal de un polígono es una línea directa entre dos vértices
- Segmento $v_i v_j$ no es una diagonal si está bloqueado por una porción del borde del polígono => $v_i v_j$ intercepta una arista del polígono
- Lema segmento: El segmento $s = v_i v_j$ es una diagonal de P si y sólo si
  - $\forall$ arista e de P no incidente en v_i o v_j, $s \cap e = \emptyset$
  - s es interna a P en la vecindad de v_i y v_j

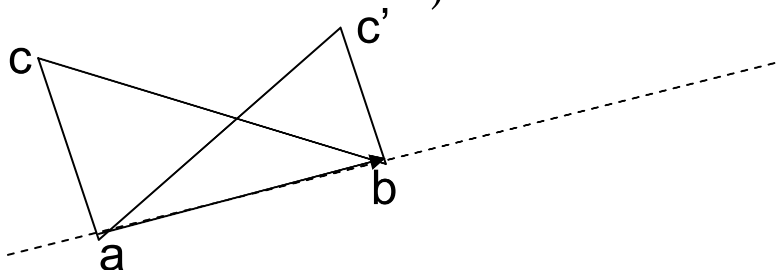


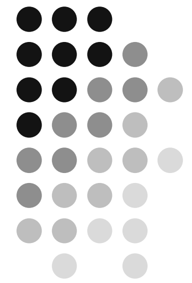
A la izquierda



- Si dos segmentos se interceptan, se puede saber con un predicado "a la izquierda", que determina si un punto está a la izquierda del segmento
- Un segmento se determina con 2 puntos dados en un orden particular (a, b)
- Un punto c está a la izquierda de (a, b) si (a, b, c) forma un circuito contrareloj

c está a la izquierda de ab si y sólo si Δabc tiene área positiva



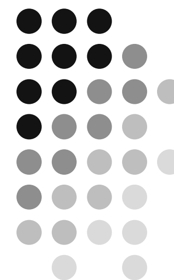


UNIVERSIDAD DE LOS ANDES

A la izquierda

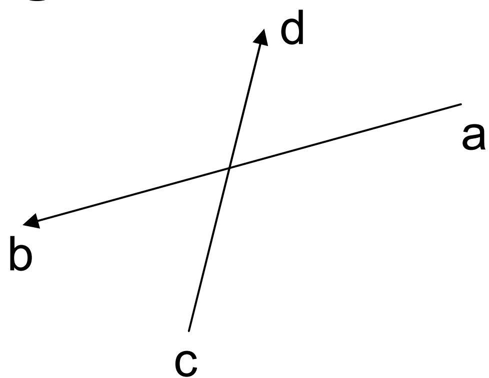
- Qué pasa si c es colineal con ab? El área de Δabc es 0
- Todas las comparaciones son con 2 veces el área, pues no interesa el valor exacto del área para decidir si está a la izquierda o no, o si son colineales
- Además al dividir por 2 se deja el mundo de los Enteros

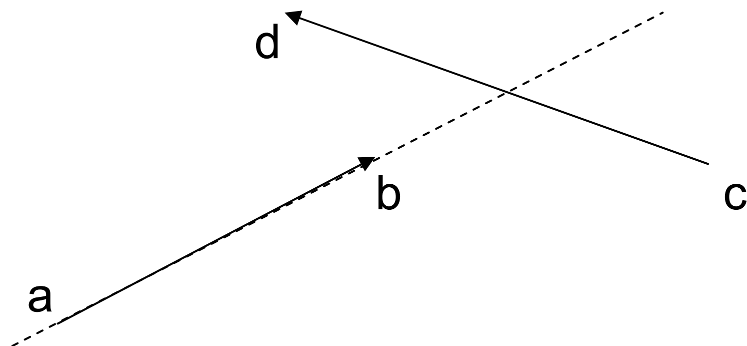
```
bool    aLaIzq(TipoPuntoI a, TipoPuntoI b, TipoPuntoI c)
{   return (area2(a, b, c) > 0);
}
bool    sobreLaIzq(TipoPuntoI a, TipoPuntoI b, TipoPuntoI
    c)
{    return (area2(a, b, c) >= 0);
}
bool    colineal(TipoPuntoI a, TipoPuntoI b, TipoPuntoI c)
{   return (area2(a, b, c) == 0);
}
```



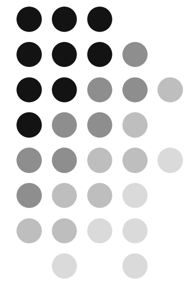
Intersección de segmentos

- Si dos segmentos ab y cd se interceptan en su interior, entonces c y d están separados por la línea L1 que contiene a y b: c está de un lado y d del otro, o viceversa.
- Ninguna de esas condiciones es suficiente para garantizar la intersección





Intersección de segmentos

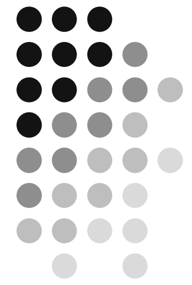


Ideal para calcular la visibilidad clara

```
bool      interseccionPropia(TipoPuntoI a, TipoPuntoI b, TipoPuntoI c, TipoPuntoI d)
{   if(colineal(a, b, c) || colineal(a, b, d) || colineal(c, d, a) || colineal(c, d, b) )
              return FALSE;
    return oExc( aLaIzq(a, b, c), aLaIzq(a, b, d)) && oExc(aLaIzq(c, d, a), aLaIzq(c, d, b));
}
bool      oExc(bool x, bool y)
{   return !x ^ !y;
}
```

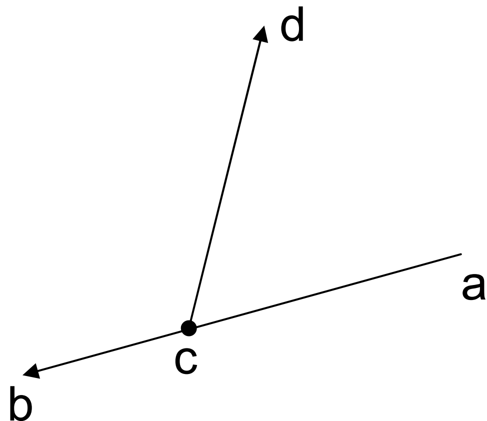
^	0	1
0	0	1
1	1	0

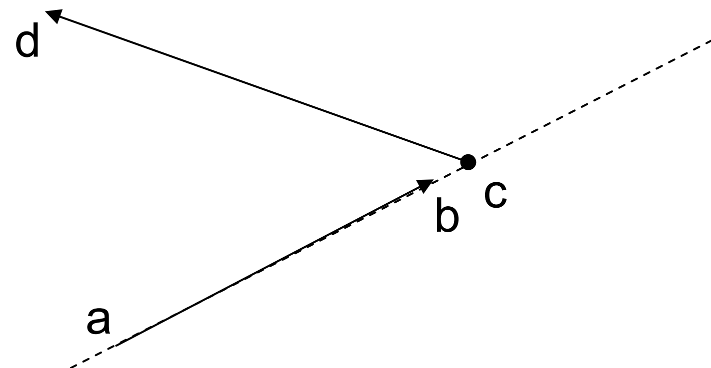
- Se puede implementar area2 para que regrese 1, 0 ó -1 y así evitar el uso de la función en otras funciones que puedan obtener un desborde en el cálculo

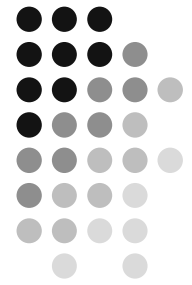


Intersección de segmentos

- Caso especial: cuando un punto borde (c) de un segmento está sobre el otro segmento (ab)
  - Solo pasa si a, b y c son colineales
  - Pero esa no es una condición suficiente, lo que se necesita realmente es saber si c está entre a y b





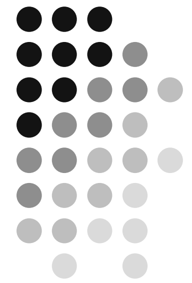


Entre

- Si ab no es vertical => c está sobre ab si y sólo si la coordenada X de c está en el intervalo de las coordenadas X de a y b
- Si ab es vertical => se realiza el mismo chequeo para la coordenada Y

```
bool    entre(TipoPuntoI a, TipoPuntoI b, TipoPuntoI c)
{   TipoPuntoI   ba, ca;
    if( ! colineal(a, b, c) )            return FALSE;
    if (a[X] != b[X])
        return (   (a[X] <= c[X]) && (c[X] <= b[X]) ) ||
               (   (a[X] >= c[X]) && (c[X] >= b[X]) );
    else
        return (   (a[Y] <= c[Y]) && (c[Y] <= b[Y]) ) ||
               (   (a[Y] >= c[Y]) && (c[Y] >= b[Y]) ) ;
}
```

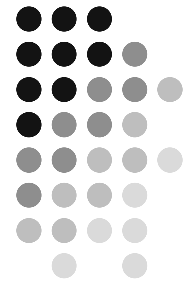
Intersección de segmentos



- Dos segmentos se interceptan si y sólo si ellos tienen una intersección propia o el punto borde de un segmento está entre los dos puntos bordes del otro segmento

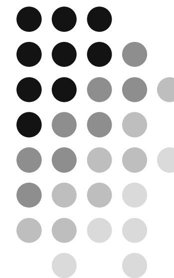
```
bool      interseccion(TipoPuntoI a, TipoPuntoI b, TipoPuntoI c, TipoPuntoI d)
{   if( interseccionPropia(a, b, c, d )   return TRUE;
    else if( entre(a, b, c)  | |
             entre(a, b, d)  | |
             entre(c, d, a)  | |
             entre(c, d, b)     )            return TRUE;
    else
            return FALSE;
}
```

Implementación de la triangulación



- Paso 1. Encontrar una diagonal de P, dos condiciones:
  - No intersección con aristas de P y
  - Ser interior en P
- Si se ignora la segunda condición => solo se aplica el código de intersección, para cada arista e de P no incidente sobre la ***posible*** diagonal s (e intercepta s)
- Aun puede pasar que una de las aristas incidentes en un punto borde de s puede ser colineal con s

Diagonal interna-externa



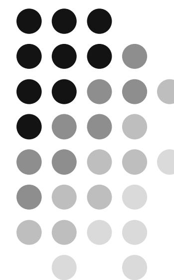
UNIVERSIDAD DE LOS ANDES

```
bool      diagonalIE(ptrVer a, ptrVer b)
{   ptrVer            c, c1;
    c = vertices;
    do
    {       c1 = c->der;
            if( (c != a ) && ( c1 != a) && (c != b ) && ( c1 != b )
               && interseccion(a->v, b->v, c->v, c1->v )  )          return FALSE;
             c = c->der;
    }while( c != vertices );
    return TRUE;
}
```

- Segunda condición del lema segmento, hay que distinguir entre una diagonal interna y una externa
- Un vector B está en el cono abierto contrareloj entre otros dos vectores A y C.

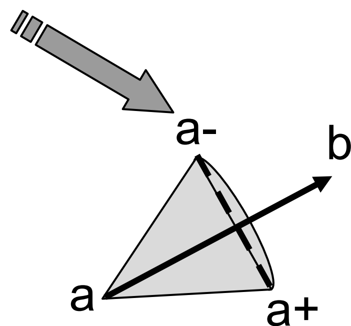
UNIVERSIDAD
DE LOS ANDES

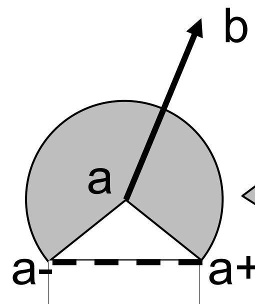
Diagonal dentro del cono



- B está en la diagonal, A y C están sobre las aristas consecutivas de P
- Caso vértice convexo: diagonal s=ab es interna en P si y sólo si s es interna en el cono cuyo ápice o punta es a y cuyos lados pasan por a- y a+

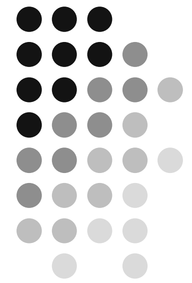
  a- debe estar a la izquierda de ab y a+ debe estar a la izquierda de ba





Caso vértice cóncavo: el exterior de la vecindad de a es un cono donde a es el vértice convexo
Un vértice cóncavo luce convexo si el interior y el exterior se intercambian

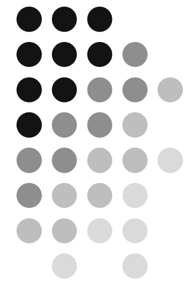
Diagonal dentro del cono



- Convexo: a es convexo si y sólo si a- está a la izquierda o sobre aa+
- Si (a-, a, a+) son colineales, el ángulo interno en a es π, que se definió como convexo

```
bool        enElCono(ptrVer a, ptrVer b)
{   ptrVer          a0, a1;
    a0 = a->der;
    a1 = a->izq;
    if( aLaIzq(a->v, a1->v, a0->v) )                // a es convexo
            return  aLaIzq(a->v, b->v, a0->v) && aLaIzq( b->v, a->v, a1->v );
    return  !( aLaIzq(a->v, b->v, a1->v ) && aLaIzq( b->v, a->v, a0->v )  );
}
```

Diagonal

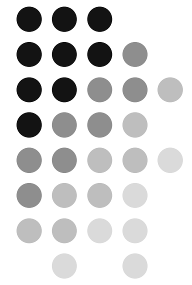


UNIVERSIDAD DE LOS ANDES

```
bool        diagonal(ptrVer a, ptrVer b)
{
    return enElCono(a, b) && enElCono(b, a) && diagonalIE(a, b);
}
```

- Escogencia del orden de invocación de las funciones
  - La función enElCono( ) está en O(1)
  - La función diagonalIE( ) está en O(n) donde n es el número de aristas de P
  - Si alguna de las dos llamadas de la función enElCono regresa falso => no se invocará la función más costosa

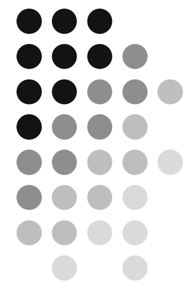
Ejercicios



1. En una máquina con enteros de $\pm 2^{31}$, ¿qué tan grandes pueden ser a, b y c para evitar el desborde en el cálculo de area2?
2. Si se elimina la sentencia if, de la función interseccionPropia( ), ¿qué calcula exactamente? Aun así intersección( ) trabaja apropiadamente?
3. Ejecute a mano el código de interseccion( ) y determine el número más grande de llamadas a area2 que se podrían tener. Diseñe una nueva versión que evite los duplicados.
4. Proponga un esquema para evitar la duplicación de la prueba de intersección de dos segmentos. ¿Cúal es la complejidad en tiempo y espacio del nuevo algoritmo?

Triangulación por remoción de las orejas

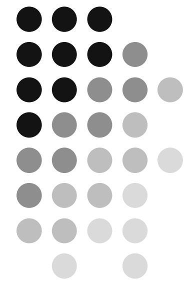
UNIVERSIDAD DE LOS ANDES



- Método: siguiendo la prueba del teorema de triangulación, repetir:
  - Encontrar una diagonal
  - Picar P en dos pedazos
- Análisis: está en $O(n^4)$
  - Hay $O(n^2)$ candidatas a diagonales, prueba de cada una está en $O(n)$
  - Repitiendo $O(n^3)$ por cada una de las n-3 diagonales

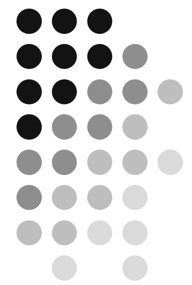
Es ineficiente y por ello se utiliza el teorema de las dos orejas de Meisters

Triangulación por remoción de las orejas

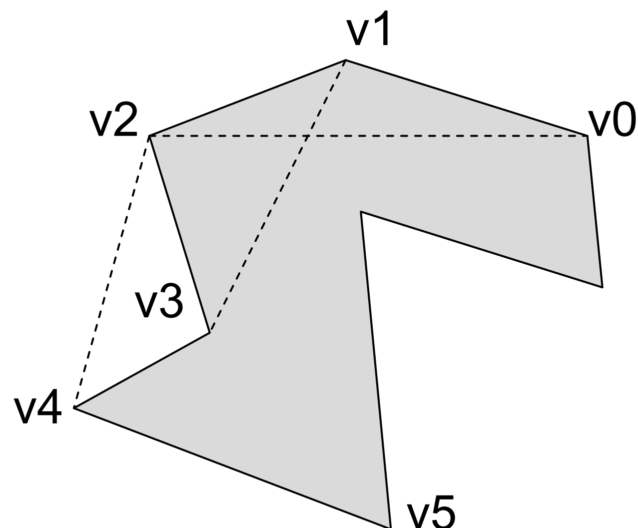


- Se sabe que hay una diagonal interna y que la misma separa una oreja
- Hay O(n) candidatas a diagonales de orejas: (v_i, v_{i+2}) con i=0,…, n-1
- La complejidad en el peor de los casos es $O(n^3)$
- La función diagonal( ) está en O(n), si se invoca n veces, se logra tener una complejidad $O(n^2)$
- La idea clave es que la eliminación de una oreja no cambia P

Triangulación por remoción de las orejas



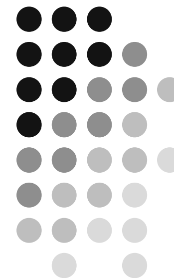
- Primero se determina para cada vértice v_i si es una posible punta de oreja en el sentido donde v_{i-1}, v_{i+1} es una diagonal, lo cual está en $O(n^2)$, pero este paso solo se realiza una vez



V2 es una punta de oreja E2=Δ(v1, v2, v3)
Si E2 se elimina, v1 y v3 podrían cambiar de status, pero los deja intactos, ya que sólo se elimina el vértice v2 que es convexo
En caso que v3v5 sea una diagonal, v4 sería una punta de oreja, pero la eliminación de E2 no cambia su status

Triangulación por remoción de las orejas

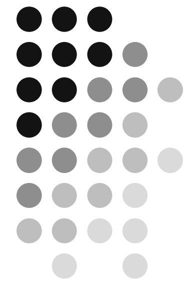
UNIVERSIDAD DE LOS ANDES



<table>
<tr><td colspan="3">Junio, 2005
triangulacion()
{pre: n > 3 } {pos: }</td></tr>
<tr><td>1
2</td><td>iniciarOreja( )
(n > 3) [v2=localizar una punta de oreja
marcar la diagonal v1v3
eliminar v2
actualizar puntaOreja de v1 y v3]</td><td>➢n: Natural. Número de vértices de P
➢iniciarOreja( ): inicia el status de cada vértice de P.</td></tr>
</table>

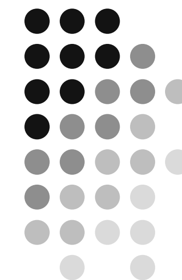
```
void        iniciarOreja(void)
{   ptrVer              v0, v1, v2;
    v1 = vertices;
    do
    {       v2  = v1->der;
            v0 = v1->izq;
            v1->oreja = diagonal(v0, v2);
            v1 = v1->der;
    }while(v1 != vertices);
}
```

Triangulación por remoción de las orejas

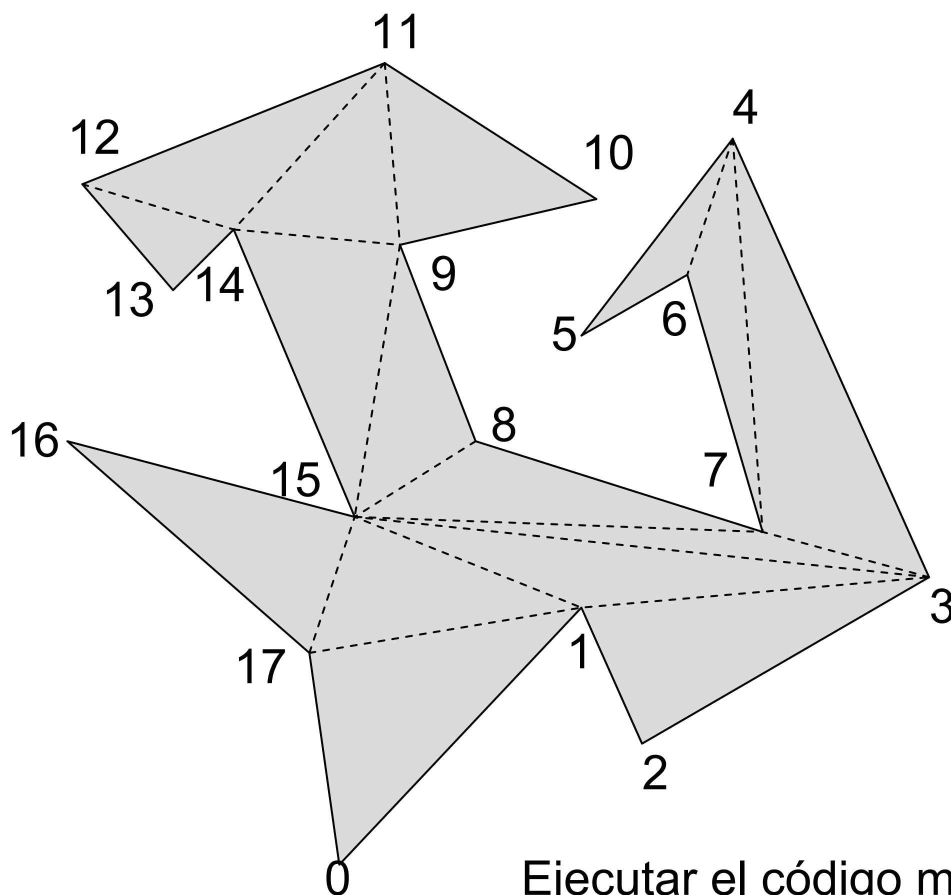


```
void        triangulacion(void)
{   ptrVer          v0, v1, v2, v3, v4;    // cinco vértices consecutivos
    int             n = vertices;
    iniciarOreja();
    while( n > 3 )
    {       v2  = vertices;
            do
            {       if(v2->oreja)          // se encontró una oreja
                    {       v3 = v2->der;        v4 = v3->der;        // llenar variables
                            v1 = v2->izq;        v0 = v1->izq;
                            imprimirDiagonal(v1, v3);
                            v1->oreja = diagonal(v0, v3);             //actualizar status
                            v3->oreja = diagonal(v1, v4);
                            v1->der = v3;                    // cortar la oreja v2
                            v3->izq = v1;
                            vertices = v3;                   // si head es v2
                            n--;
                            break;
                    }
            v2 = v2->der;
            }while(v2 != vertices);
    }
}
```

UNIVERSIDAD
DE LOS ANDES



Ejemplo (tomado de O'Rourke)




```
main()
{           leerVertices();
            escribirVertices();
            triangulacion();
}
```

i	(x,y)	i	(x,y)
0	(0,0)	9	(6,14)
1	(10,7)	10	(10,15)
2	(12,3)	11	(7,10)
3	(20,8)	12	(0,16)
4	(13,17)	13	(1,13)
5	(10,12)	14	(3,15)
6	(12,14)	15	(5,8)
7	(14,9)	16	(-2,9)
8	(8,10)	17	(5,5)

Ejecutar el código main y mostrar la salida