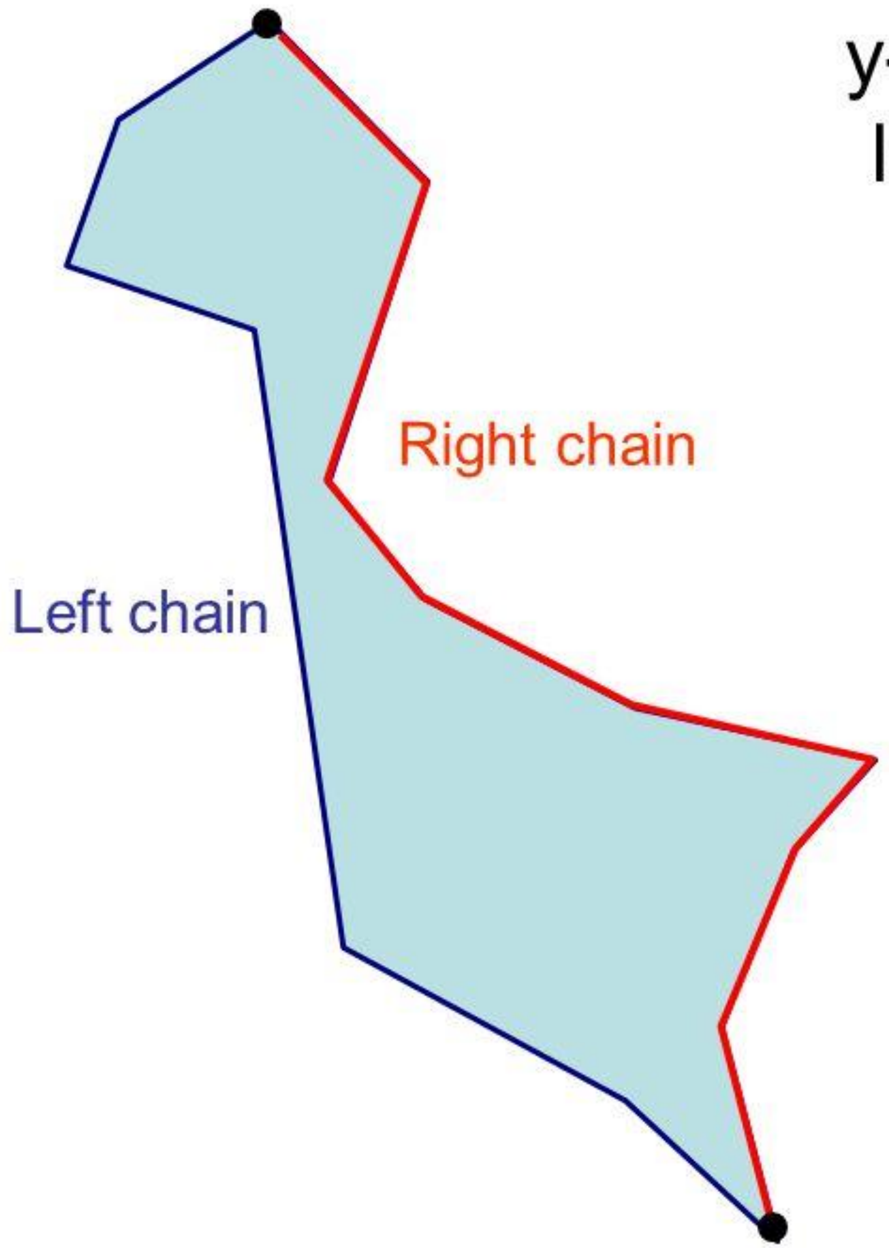
Polygon Partitioning

Partitioning the Polygon to Monotone polygons

A monotone polygon

- A polygon P is said to be monotone with respect to line L if ∂P can be split in to two polygonal chains such that each chain is monotone with respect to L
- The two chains share a vertex at either end

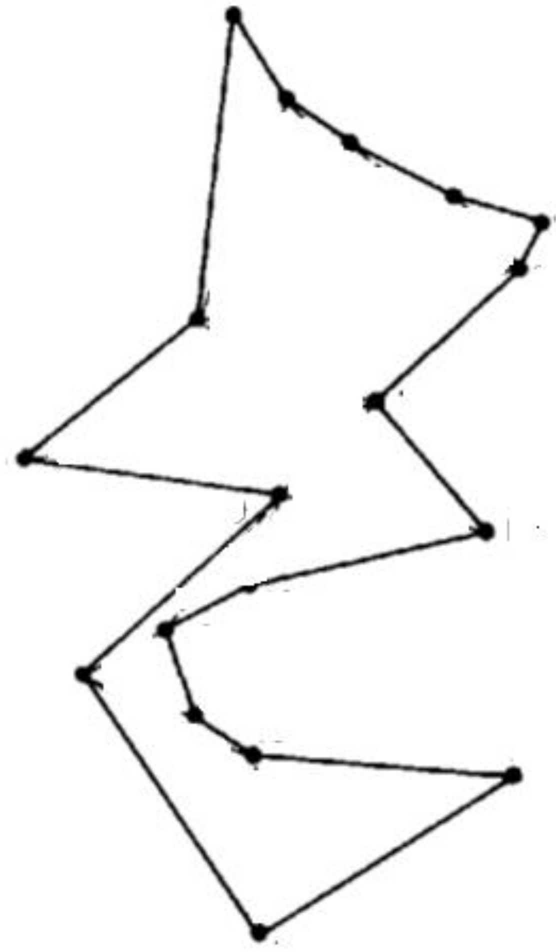
y-monotone polygon: left and right chains

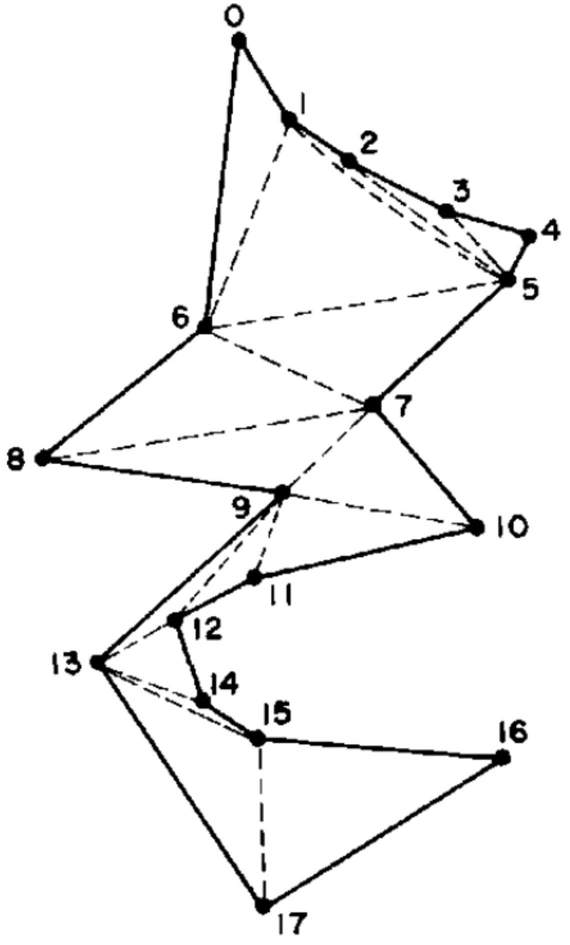


We will also assume that the polygon is strictly y-monotone, i.e. it is y-monotone and has no horizontal edges. Additionally, you may assume that no two vertices have the same y-coordinate

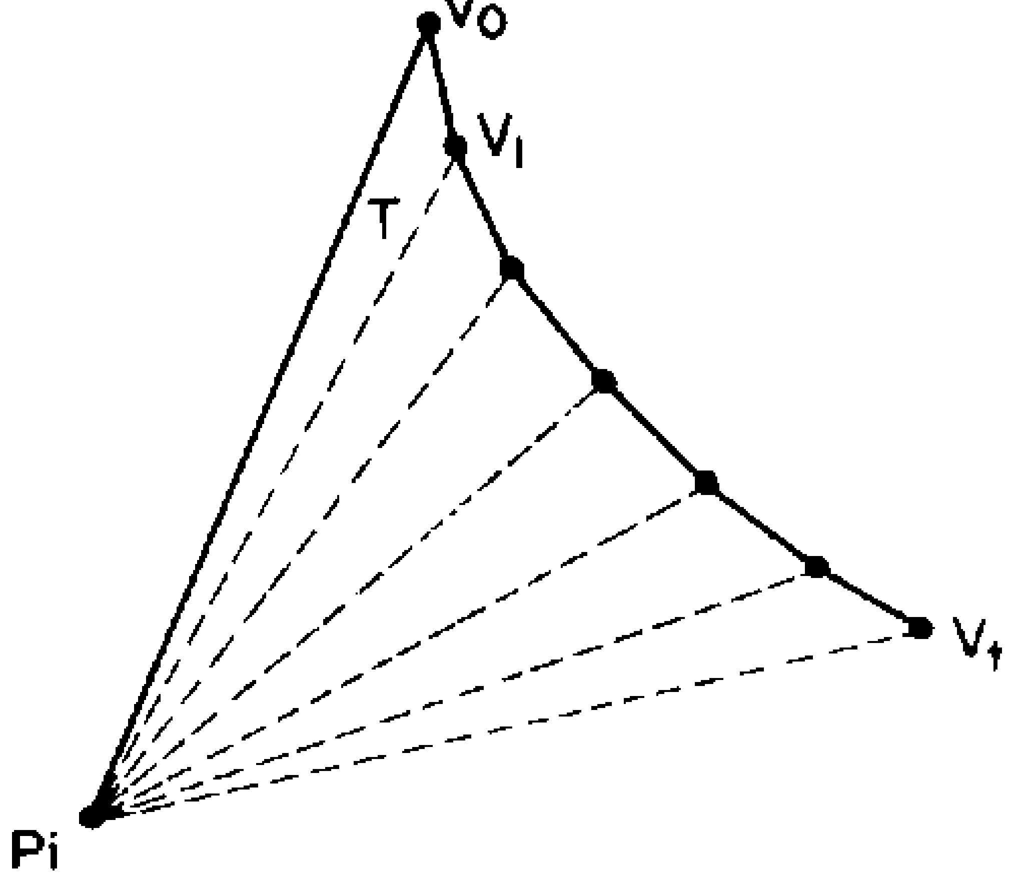
Triangulation of a monotone polygon

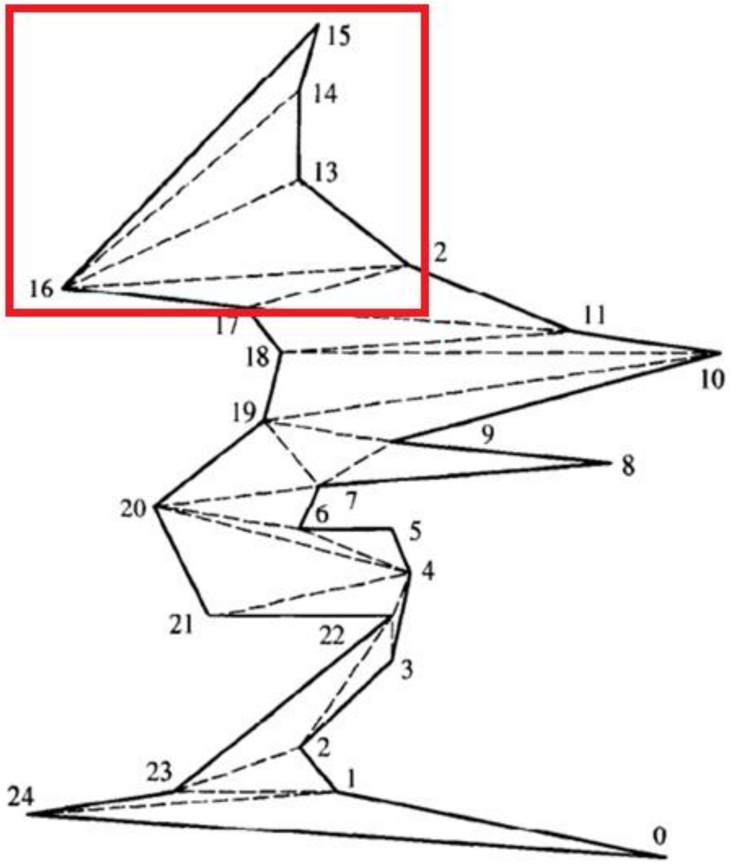
Input and Output





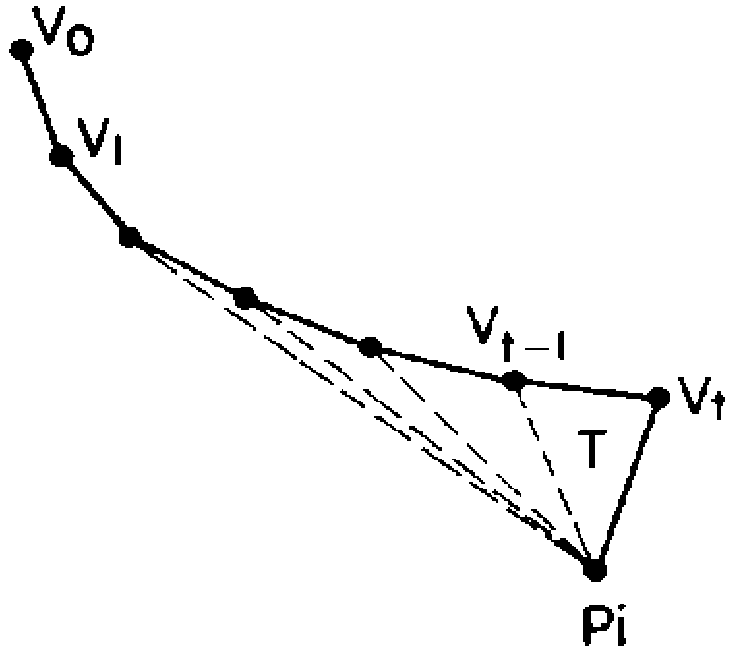
To join the vertices to form triangles: Case-1

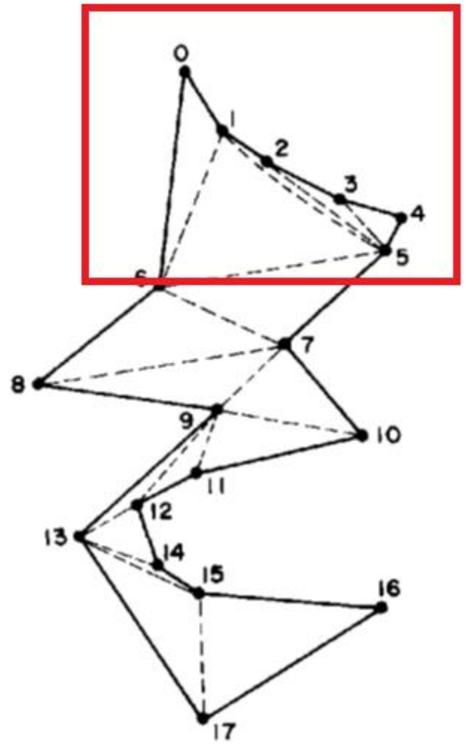




To join the vertices to form triangles

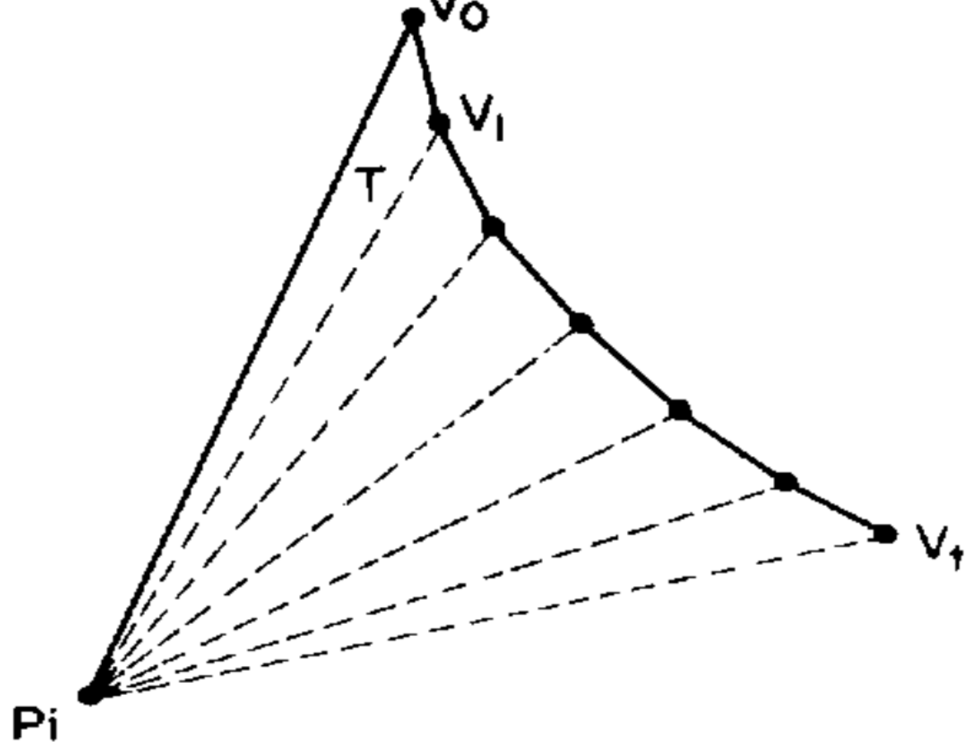
- Case-2 :

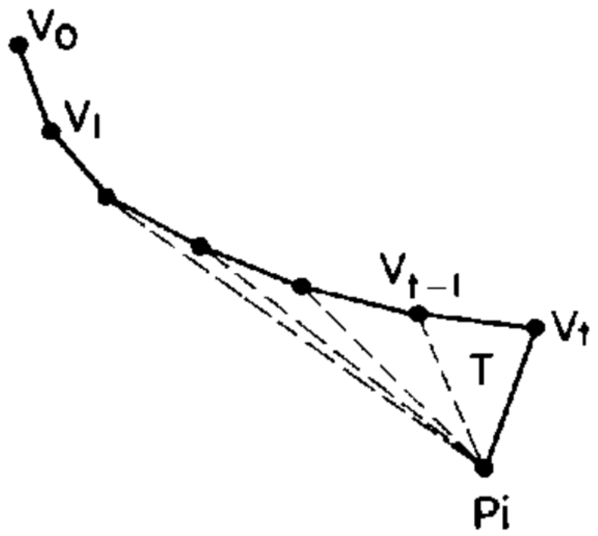




Data Structure

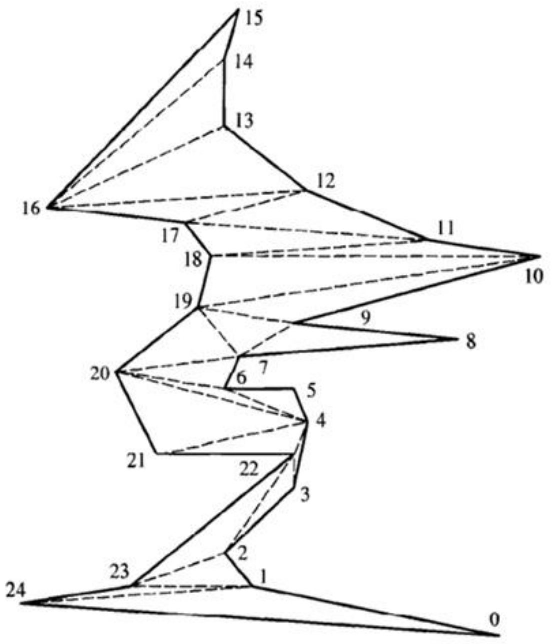
- Stack
- v_0, v_1, ….., v_t - vertices in the stack with v_t as the stack top
- p_i is the vertex to be processed

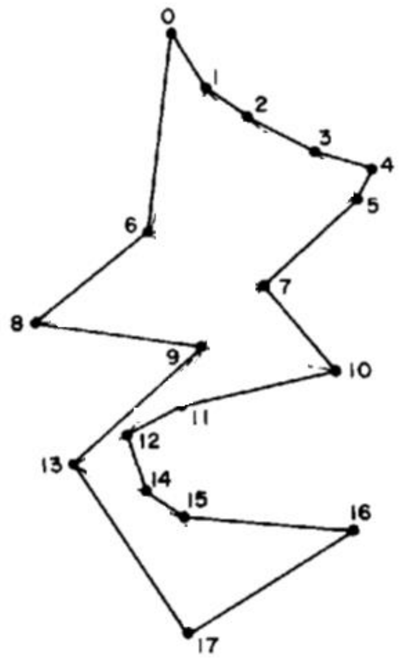




Assumptions/ Notations used in the Algorithm

- Let *p_0, . . . ,p_n* be the vertices in sorted order, with *p_0* being the next vertex to be processed.

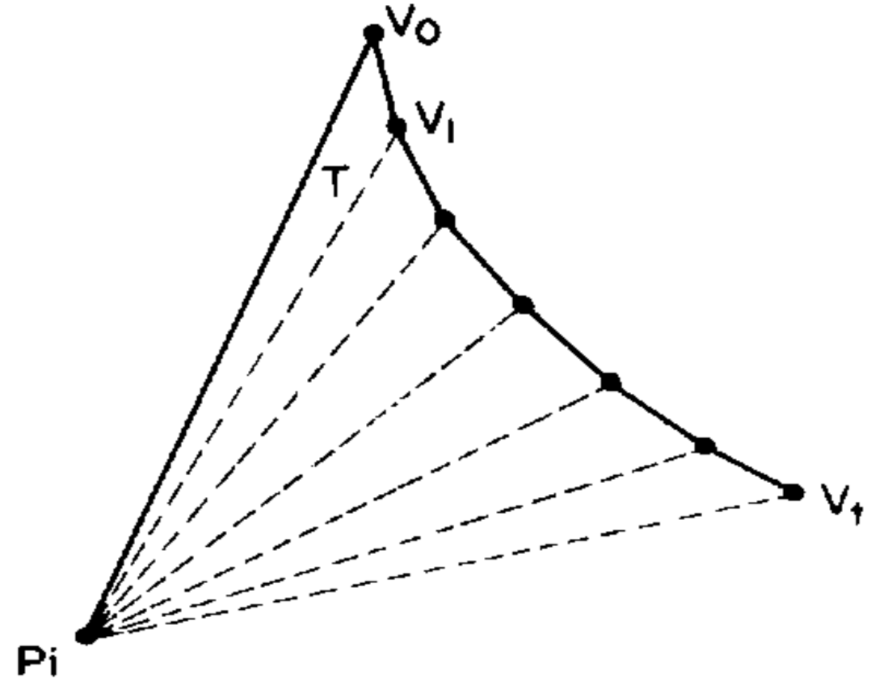


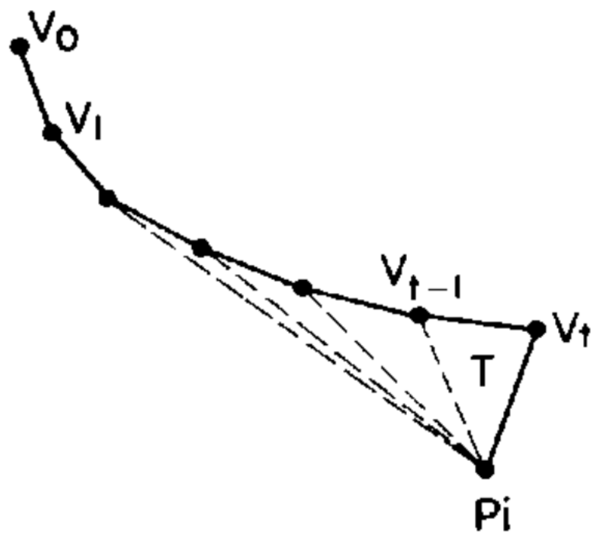


- Assume that no two vertices have the same *y*-coordinate (to simplify the presentation of algo)
- The algorithm successively reduces polygon *P* by chopping triangles off the top.

Assumptions/ Notations used in the Algorithm –contd.

- At all times it maintains a stack of all the vertices examined so far but not yet completely processed.
- Let $v_0, \ldots, v_t$ be the vertices on the stack, with v_0 on the bottom and v_t on the top of the stack, and let P_i be the polygon remaining as the step commences.

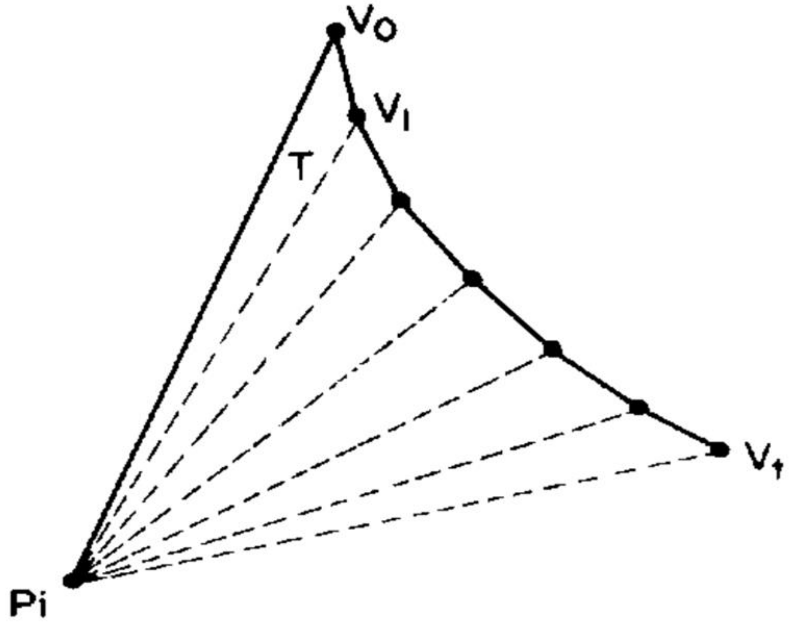


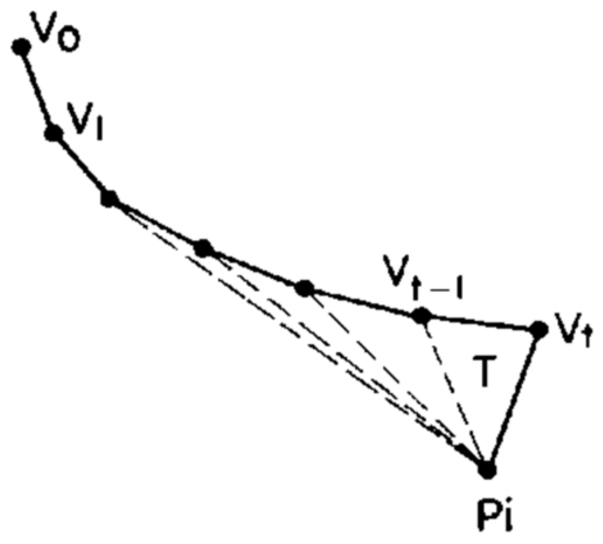


Stack properties

Maintained throughout the processing:

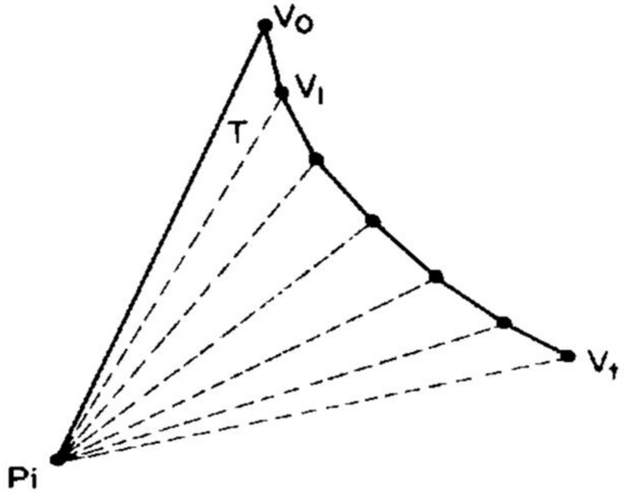
- *v0, . . . , vt* decrease by height, *vt* lowest.
- *v0 . . . , vt* form a chain of consecutive vertices on the boundary of the polygon *Pi*
- *v1 . . . , vt-1* are reflex vertices.
- The next vertex *pi* to be processed is adjacent via a polygon edge of polygon *Pi* to either *v0* or *vt* (or to both).

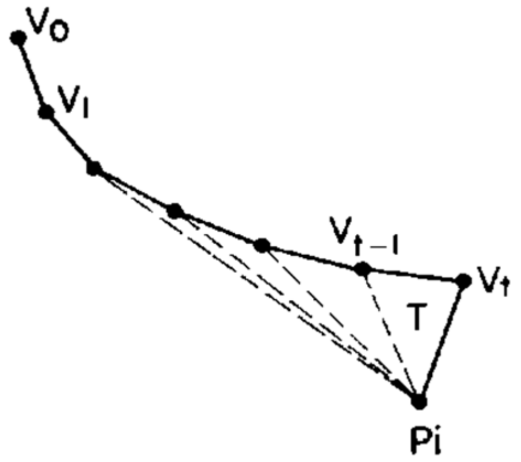




Algorithm – in a nutshell

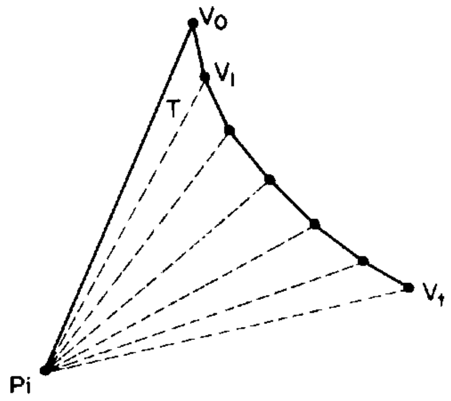
- The algorithm connects diagonals from the next vertex to the vertices on the top of the stack, pops these off the stack, and pushes the just processed vertex onto the stack.
- For example: The algorithm connects diagonals from *pi* to *vt*, *vt-1 ,..,* pops these off the stack, and pushes *pi* onto the stack.

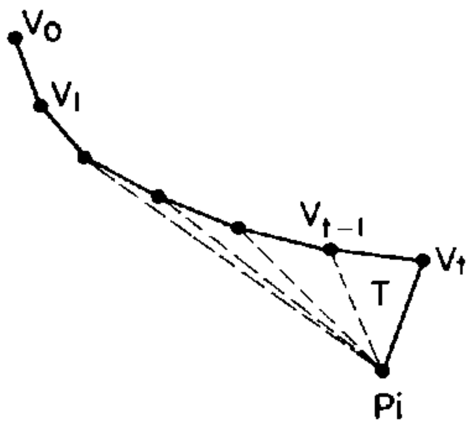




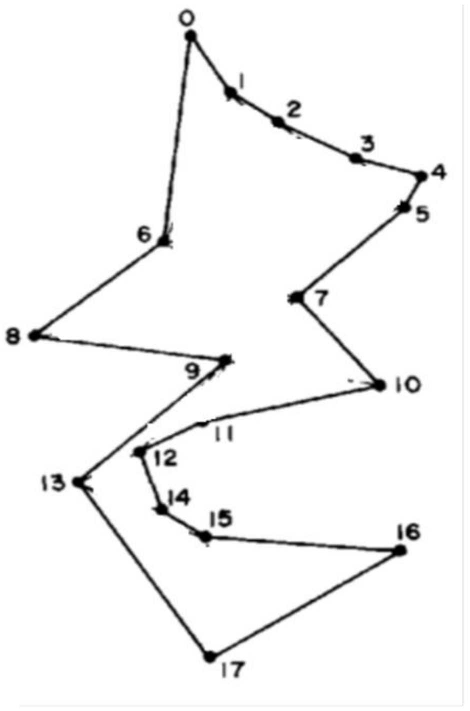
***Algorithm :* Triangulation of a Monotone Polygon**

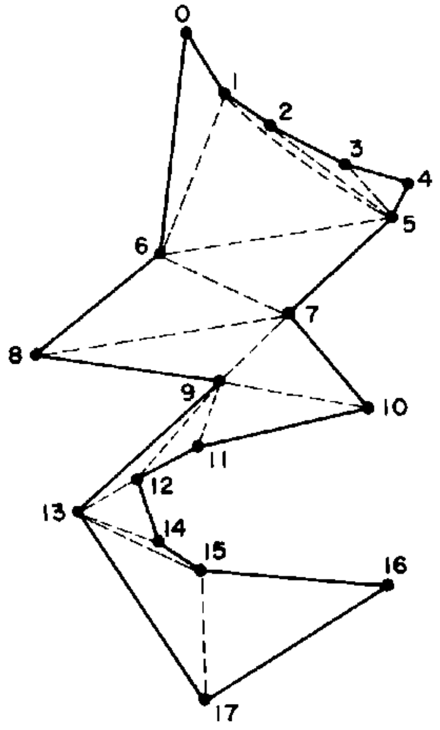
```
Sort vertices by decreasing y-coordinate, resulting in $p_0, . . . , p_n$.
Push $p_0$
Push $p_1$
for i = 2 to $n$ - 1 do
if $p_i$ is adjacent to $v_0$ then
begin
    while $t > 0$ do
    begin
    Draw diagonal $p_i$ —>$v_t$.
    Pop
    end
    Pop
    Push $v_t$.
    Push $p_i$
end
else if $p_i$ is adjacent to $v_t$ then
begin
    while $t > 0$ and $v_t$ is not reflex do
    begin
    Draw diagonal $p_i$ —> $v_{t-1}$
    Pop
    end
    Push $p_i$.
end
```



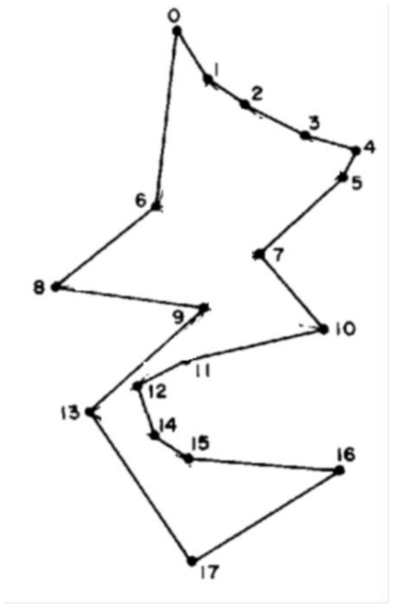


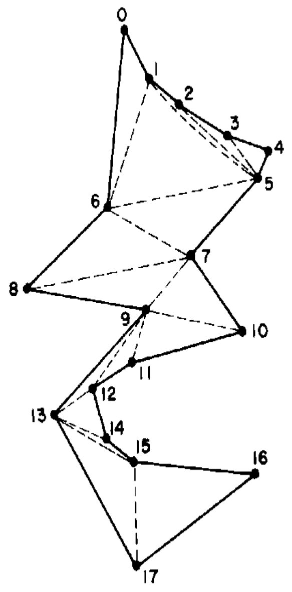
```
• Algorithm : Triangulation of a Monotone Polygon
• Sort vertices by decreasing y-coordinate, resulting in $p_0, . . . , p_n$.
• Push $p_0$.
• Push $p_1$.
• for i = 2 to $n$ - 1 do
• if  $p_i$ is adjacent to $v_0$ then
• begin
    – while $t$ > 0 do
    – begin
    – Draw diagonal $p_i$ —>$v_t$.
    – Pop
    – end
    – Pop
    – Push $v_t$
    – Push $p_i$
• end
• else if $p_i$ is adjacent to $v_t$ then
• begin
    – while $t$ > 0 and $v_t$ is not reflex do
    – begin
    – Draw diagonal $p_i$ —> $v_{t-1}$
    – Pop
    – end
    – Push $p_i$
• end
```





```
•   Algorithm : Triangulation of a Monotone Polygon
•   Sort vertices by decreasing y-coordinate, resulting in p0, . . . , pn.
•   Push p0.
•   Push p1.
•   for i = 2 to n - 1 do
•   if  pi is adjacent to v0 then
•   begin
        –   while t > 0 do
        –   begin
        –   Draw diagonal pi —>vt.
        –   Pop
        –   end
        –   Pop
        –   Push vt
        –   Push pi
•   end
•   else if pi is adjacent to vt then
•   begin
        –   while t > 0 and vt is not reflex do
        –   begin
        –   Draw diagonal pi —> vt-1
        –   Pop
        –   end
        –   Push pi
•   end
```

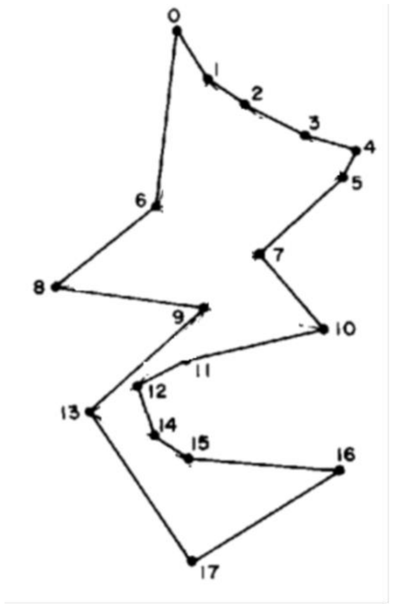


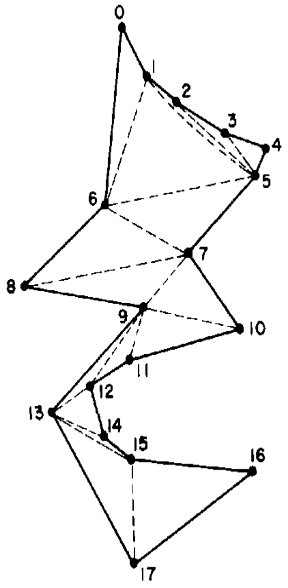


i	stack	condn	while	diag
2				
3				
4				
5				

Algorithm : Triangulation of a Monotone Polygon

```
• Sort vertices by decreasing y-coordinate, resulting in p0, . . . , pn.
• Push p0.
• Push p1.
• for i = 2 to n - 1 do
• if  pi is adjacent to v0 then
• begin
    – while t > 0 do
    – begin
    – Draw diagonal pi —>vt.
    – Pop
    – end
    – Pop
    – Push vt
    – Push pi
• end
• else if pi is adjacent to vt then
• begin
    – while t > 0 and vt is not reflex do
    – begin
    – Draw diagonal pi —> vt-1
    – Pop
    – end
    – Push pi
• end
```

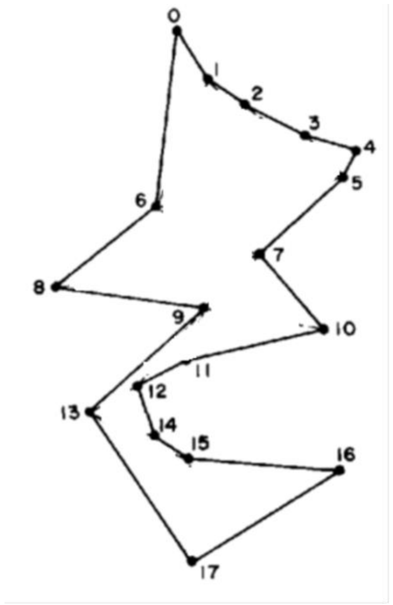


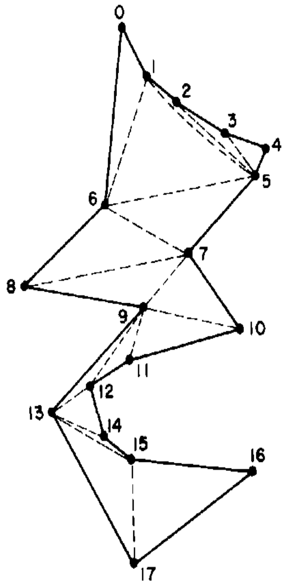


i	stack	condn	while	diag
2	0,1	else	No (1 refl)	
3	0,1,2	else	No(2 refl)	
4				
5				

- ***Algorithm* : Triangulation of a Monotone Polygon**
- Sort vertices by decreasing y-coordinate, resulting in *p0, . . . , pn*.

```
Push p0.
Push p1.
for i = 2 to n - 1 do
if  pi is adjacent to v0 then
begin
    while t > 0 do
    begin
    Draw diagonal pi —>vt.
    Pop
    end
    Pop
    Push vt
    Push pi
end
else if pi is adjacent to vt then
begin
    while t > 0 and vt is not reflex do
    begin
    Draw diagonal pi —> vt-1
    Pop
    end
    Push pi
end
```

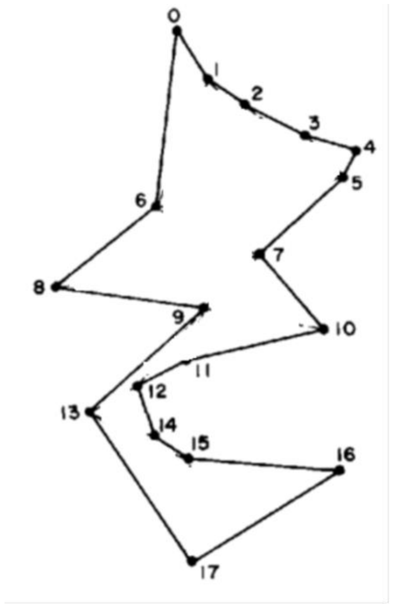


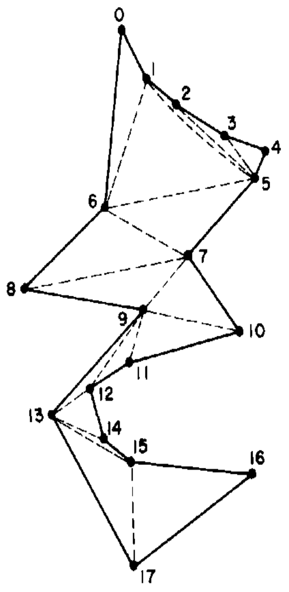


i	stack	condn	while	diag
2	0,1	else	No (1 refl)	
3	0,1,2	else	No(2 refl)	
4	0,1,2,3	else	No(3 refl)	
5	0,1,2,3, 4	else	Yes(4 notref)	5,3
5				
5				

Algorithm : Triangulation of a Monotone Polygon

```
• Sort vertices by decreasing y-coordinate, resulting in p0, . . . , pn.
• Push p0.
• Push p1.
• for i = 2 to n - 1 do
• if  pi is adjacent to v0 then
• begin
    – while t > 0 do
    – begin
    – Draw diagonal pi —>vt.
    – Pop
    – end
    – Pop
    – Push vt
    – Push pi
• end
• else if pi is adjacent to vt then
• begin
    – while t > 0 and vt is not reflex do
    – begin
    – Draw diagonal pi —> vt-1
    – Pop
    – end
    – Push pi
• end
```

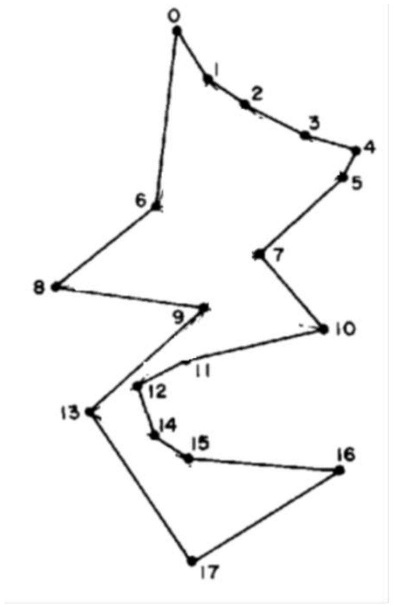


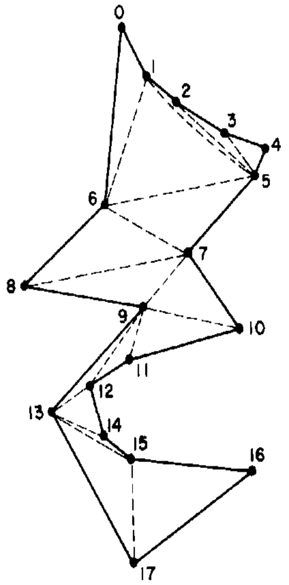


i	stack	condn	while	diag
2	0,1	else	No (1 refl)	
3	0,1,2	else	No(2 refl)	
4	0,1,2,3	else	No(3 refl)	
5	0,1,2,3, 4	else	Yes(4 notref)	5,3
5	0,1,2,3	Yes(angle 532 or angle 3 *not reflex*)		5,2
5	0,1,2	Yes(angle 521 or angle 2 *not reflex*)		5,1

Algorithm : Triangulation of a Monotone Polygon

```
• Sort vertices by decreasing y-coordinate, resulting in p0, . . . , pn.
• Push p0.
• Push p1.
• for i = 2 to n - 1 do
• if  pi is adjacent to v0 then
• begin
    – while t > 0 do
    – begin
    – Draw diagonal pi —>vt.
    – Pop
    – end
    – Pop
    – Push vt
    – Push pi
• end
• else if pi is adjacent to vt then
• begin
    – while t > 0 and vt is not reflex do
    – begin
    – Draw diagonal pi —> vt-1
    – Pop
    – end
    – Push pi
• end
```

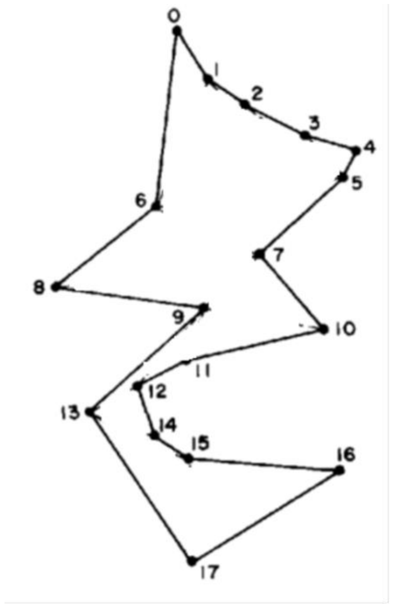


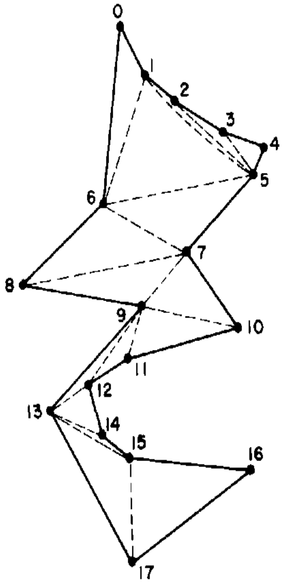


i	stack	condn	while	diag
2	0,1	else	No (1 refl)	
3	0,1,2	else	No(2 refl)	
4	0,1,2,3	else	No(3 refl)	
5	0,1,2,3, 4	else	Yes(4 notref)	5,3
5	0,1,2,3	Yes(angle 532 or angle 3 *not reflex*)		5,2
5	0,1,2	Yes(angle 521 or angle 2 *not reflex*)		5,1
5	0,1		No(1 ref)	
5	0,1,5			
6	0,1,5	if	Yes	6,5

- ***Algorithm* : Triangulation of a Monotone Polygon**
- Sort vertices by decreasing y-coordinate, resulting in *p0, . . . , pn*.

```
Push p0.
Push p1.
for i = 2 to n - 1 do
if  pi is adjacent to v0 then
begin
    while t > 0 do
    begin
    Draw diagonal pi —>vt.
    Pop
    end
    Pop
    Push vt
    Push pi
end
else if pi is adjacent to vt then
begin
    while t > 0 and vt is not reflex do
    begin
    Draw diagonal pi —> vt-1
    Pop
    end
    Push pi
end
```

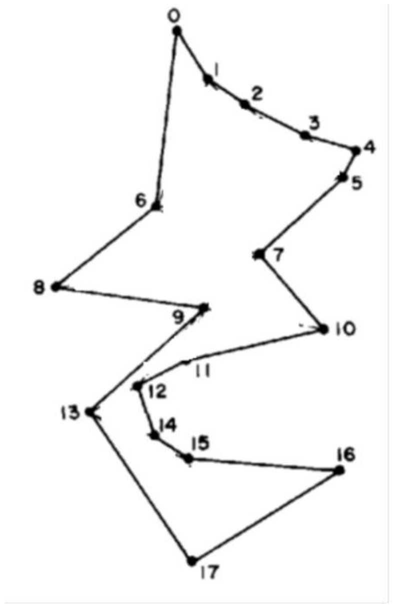


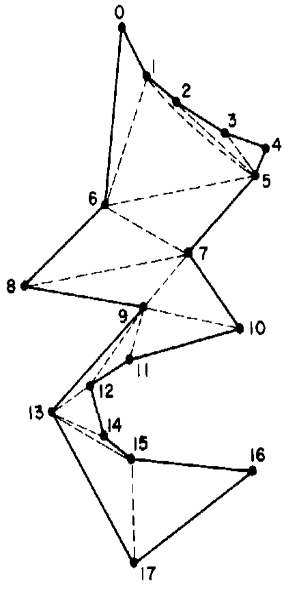


i	stack	condn	while	diag
2	0,1	else	No (1 refl)	
3	0,1,2	else	No(2 refl)	
4	0,1,2,3	else	No(3 refl)	
5	0,1,2,3, 4	else	Yes(4 notref)	5,3
5	0,1,2,3	Yes(angle 532 or angle 3 *not reflex*)		5,2
5	0,1,2	Yes(angle 521 or angle 2 *not reflex*)		5,1
5	0,1		No(1 ref)	
5	0,1,5			
6	0,1,5	if	Yes	6,5
6	0,1		Yes	6,1
6	0		No	
6	5,6			

Algorithm : Triangulation of a Monotone Polygon

```
Sort vertices by decreasing y-coordinate, resulting in p0, . . . , pn.
Push p0.
Push p1.
for i = 2 to n - 1 do
if  pi is adjacent to v0 then
Begin
    while t > 0 do
    begin
    Draw diagonal pi —>vt.
    Pop
    end
    Pop
    Push vt
    Push pi
end
else if pi is adjacent to vt then
begin
    while t > 0 and vt is not reflex do
    begin
    Draw diagonal pi —> vt-1
    Pop
    end
    Push pi
end
```

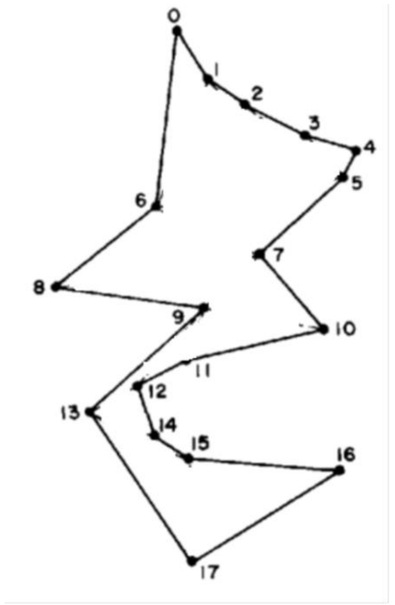


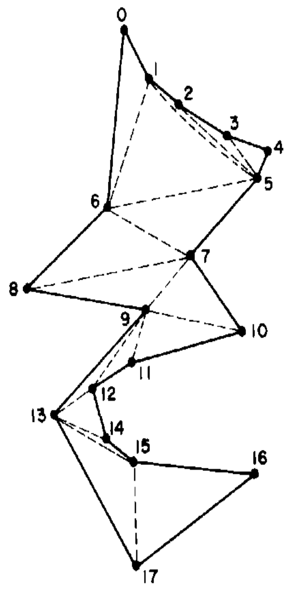


i	stack	condn	while	diag
7	5,6	if	yes	7,6
7	5		no	
7	6,7			
8	6,7	if	yes	8,7
8	6		no	
8	7,8			
9	7,8	Else, Yes(angle 987 or angle 8 not reflex)		9,7
9	7	No (t=0)		
9	7,9			
10	7,9	if	yes	10,9
10	7	No (t=0)		
10	9,10			
11	9,10	Else, Yes(angle 11,10,9 or angle 10 not reflex)		11,9

Algorithm : Triangulation of a Monotone Polygon

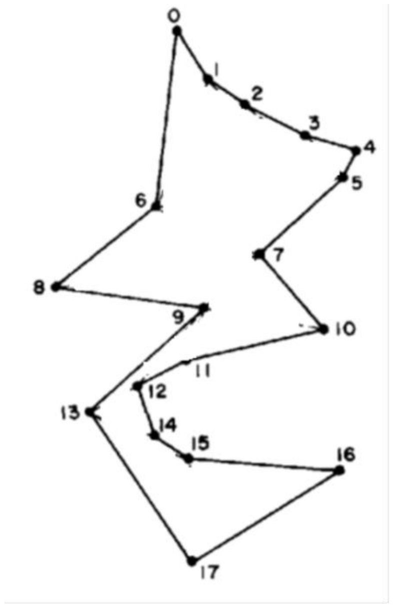
```
Sort vertices by decreasing y-coordinate, resulting in p0, . . . , pn.
Push p0.
Push p1.
for i = 2 to n - 1 do
if  pi is adjacent to v0 then
Begin
    while t > 0 do
    begin
    Draw diagonal pi —>vt.
    Pop
    end
    Pop
    Push vt
    Push pi
end
else if pi is adjacent to vt then
begin
    while t > 0 and vt is not reflex do
    begin
    Draw diagonal pi —> vt-1
    Pop
    end
    Push pi
end
```

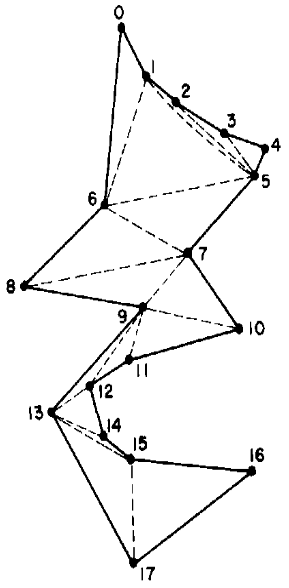




i	stack	condn	while	diag
11	9		No, t=0	
11	9,11			
12	9,11	Else, Yes(angle 12,11,9 or angle 11 not reflex)		12,9
12	9		No, t=0	
12	9,12			
13	9,12	if	yes	13,12
13	9		No, t=0	
13	12,13			
14	12,13	if	yes	14,13
14	12		No, t=0	
14	13,14			
15	13,14	Else, Yes(angle 15,14,13 or angle 14 not reflex)		15,13

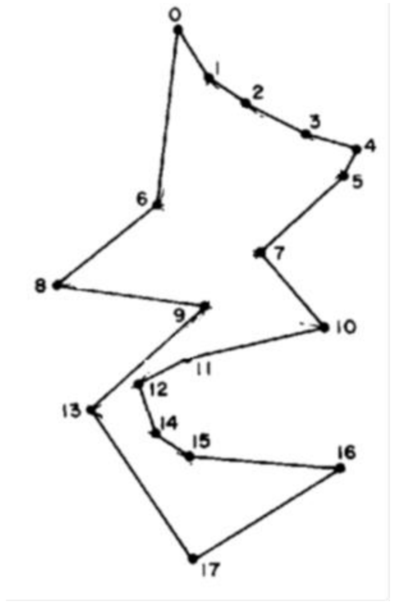
- ***Algorithm :* Triangulation of a Monotone Polygon**
- Sort vertices by decreasing y-coordinate, resulting in *p0, . . . , pn.*
- Push *p0.*
- Push *p1.*
- **for** i = 2 **to** *n* - 1 **do**
- **if** *pi* is adjacent to *v0* **then**
- **Begin**
  - **while** *t* > 0 **do**
  - **begin**
  - Draw diagonal *pi —>vt.*
  - Pop
  - **end**
  - Pop
  - Push *vt*
  - Push *pi*
- **end**
- **else if** *pi* is adjacent to *vt* **then**
- **begin**
  - **while** *t* > 0 and *vt* is not reflex **do**
  - **begin**
  - Draw diagonal *pi —> vt-1*
  - Pop
  - **end**
  - Push *pi*
- **end**

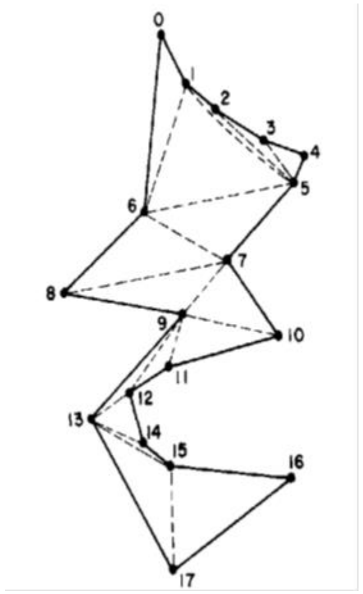


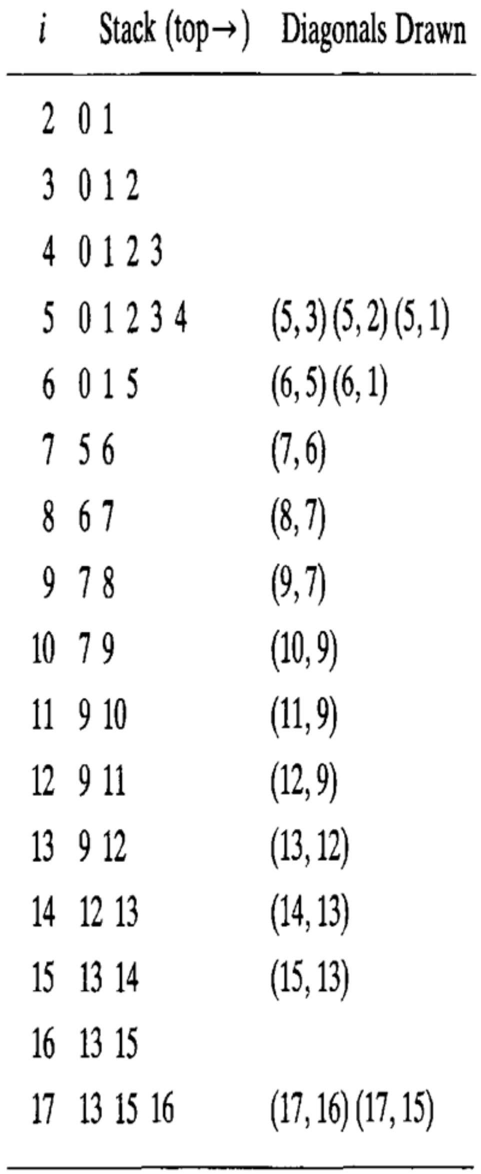


i	stack	condn	while	diag
15	13		No, t=0	
15	13,15			
16	13,15	Else, No(angle 16,15,13 or angle 15 reflex)		
16	13,15, 16			
17	13,15, 16	if	yes	17,16
17	13,15		yes	17,15
17	13		no	
17	16,17			

Summary

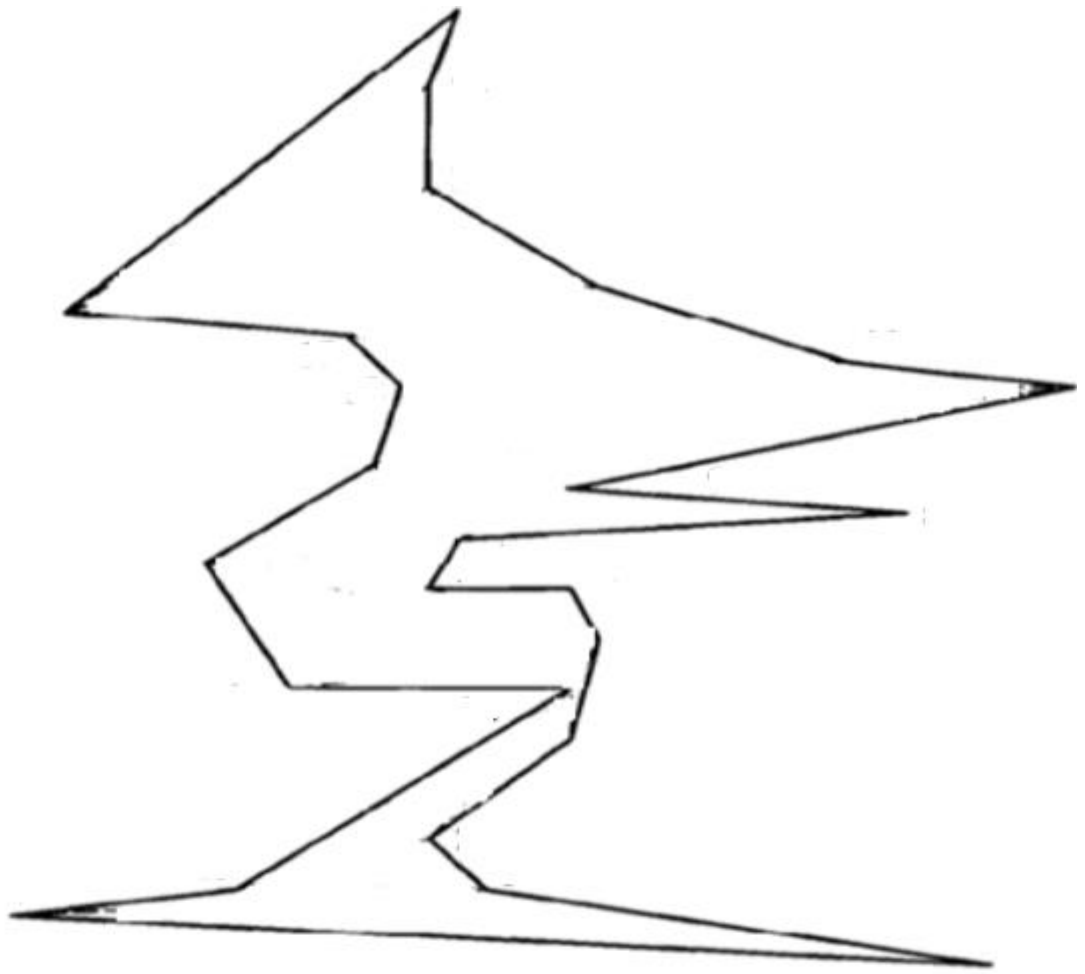






i	Stack (top→)	Diagonals Drawn
2	0 1	
3	0 1 2	
4	0 1 2 3	
5	0 1 2 3 4	(5, 3) (5, 2) (5, 1)
6	0 1 5	(6, 5) (6, 1)
7	5 6	(7, 6)
8	6 7	(8, 7)
9	7 8	(9, 7)
10	7 9	(10, 9)
11	9 10	(11, 9)
12	9 11	(12, 9)
13	9 12	(13, 12)
14	12 13	(14, 13)
15	13 14	(15, 13)
16	13 15	
17	13 15 16	(17, 16) (17, 15)

Exercise : Triangulate the given polygon using the algorithm



References

- J. O’Rourke, *Computational Geometry in C*, 2/e, Cambridge University Press, 1998
- J. O'Rourke: Art Gallery Theorems and Algorithms

Thank you