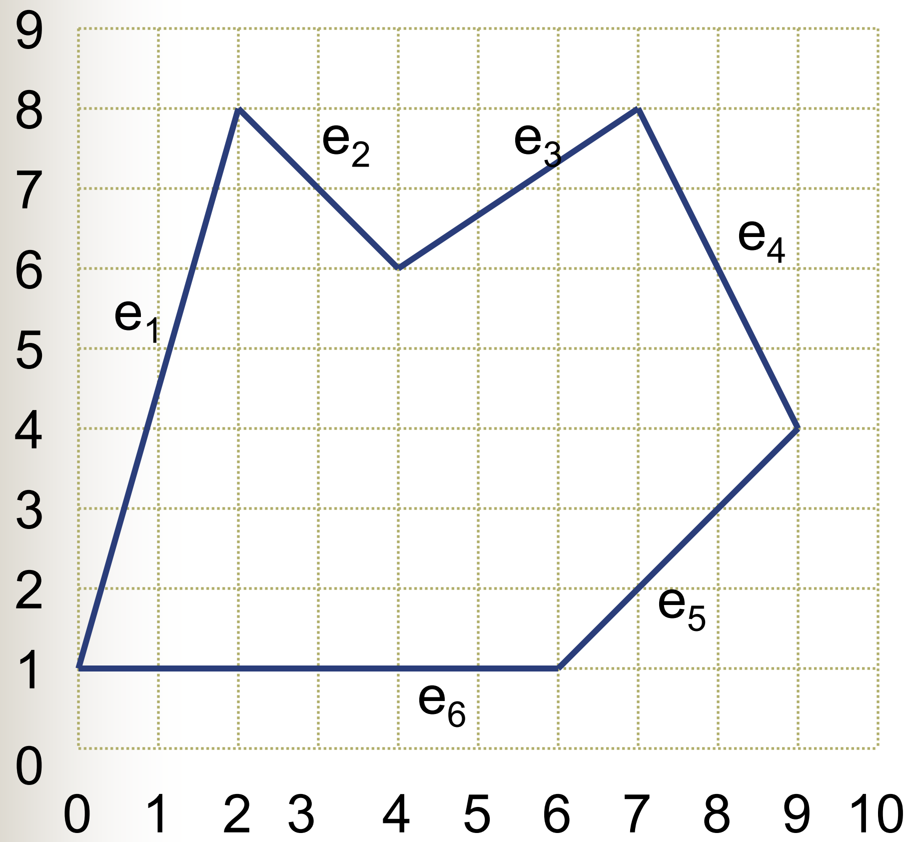
Algoritmo Scanline

Maria Cristina F. de Oliveira

Junho 2019

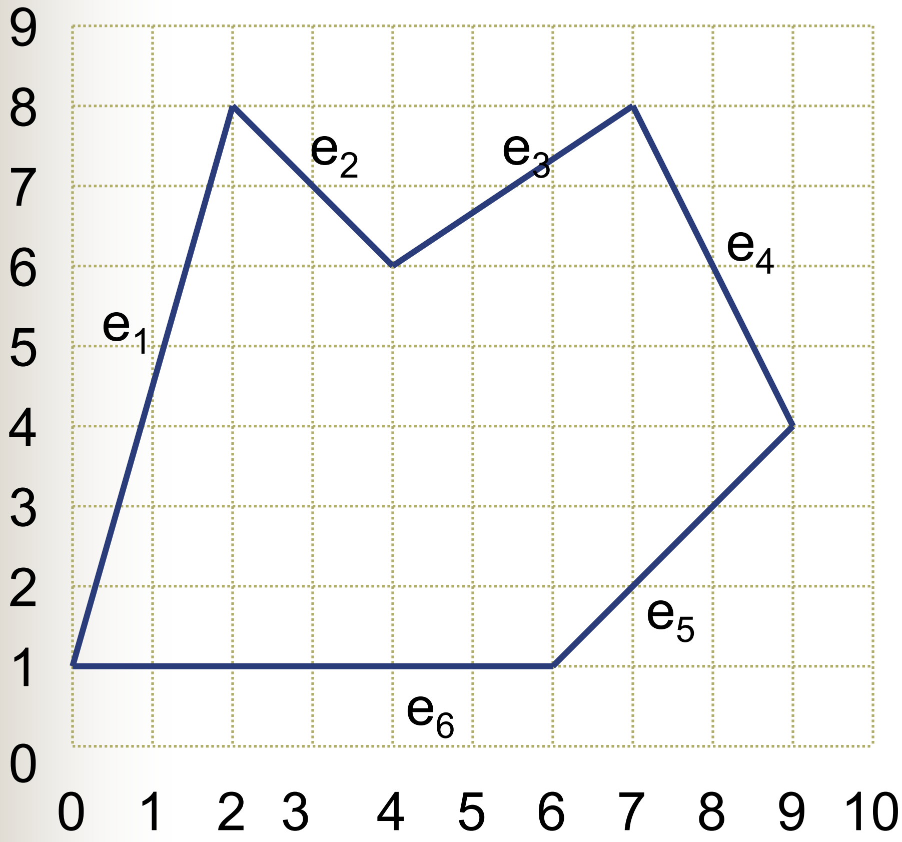
Exemplo



Este polígono tem como vértices:
(0,1)
(2,8)
(4,6)
(7,8)
(9,4)
(6,1)

e 6 arestas
e_1, .. e_6

Algoritmo Scanline

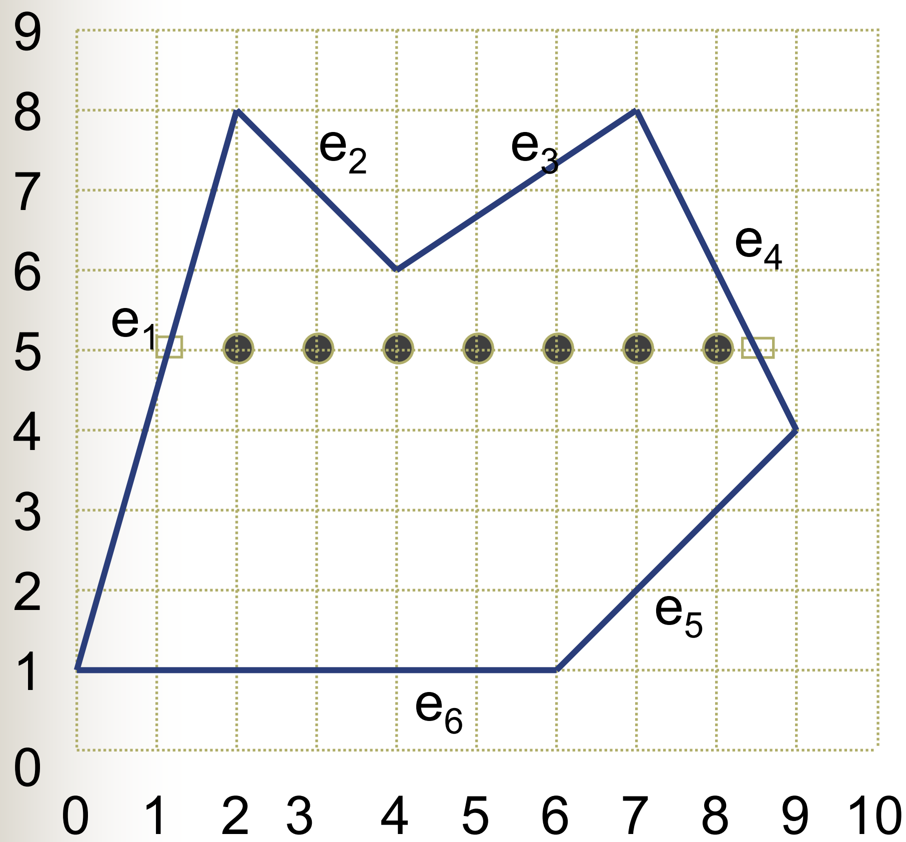


Para preencher a área, processa as linhas de varredura (*scanlines*)

Para um dado valor y; acha intersecções com arestas

Pares de intersecções sucessivas definem um bloco (*span*) de pixels

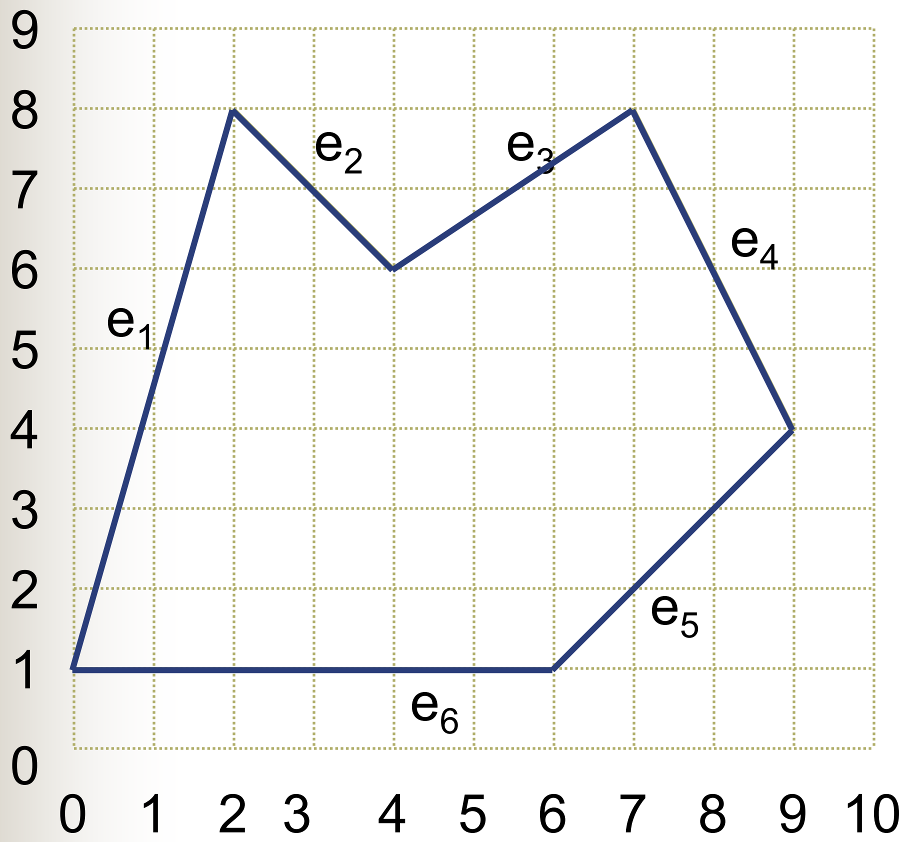
Casos a tratar: 1. normal



Scanline 5:
interseção e_1 = 1.14
interseção e_4 = 8.5

Pixels 2-8 na *scanline* são preenchidos

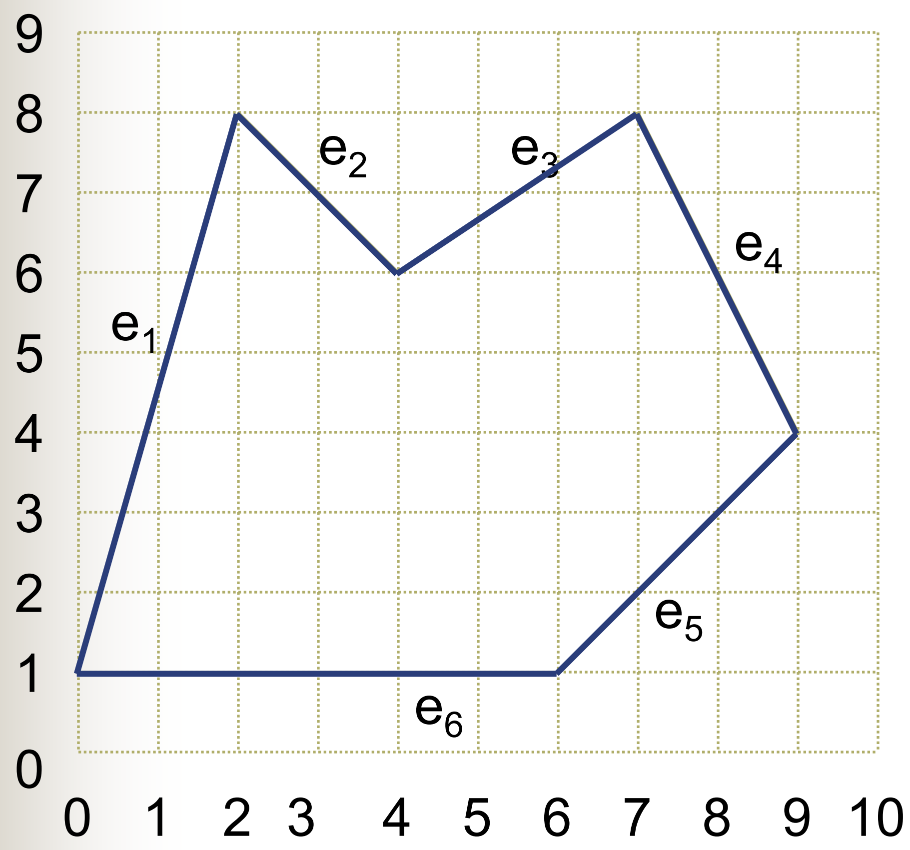
Casos a tratar: 2. Intercepta vértices



scanline 6 intercepta e_1,e_2,e_3,e_4 em 1.4, 4, 4, 8 respectiva/e
- dois spans traçados

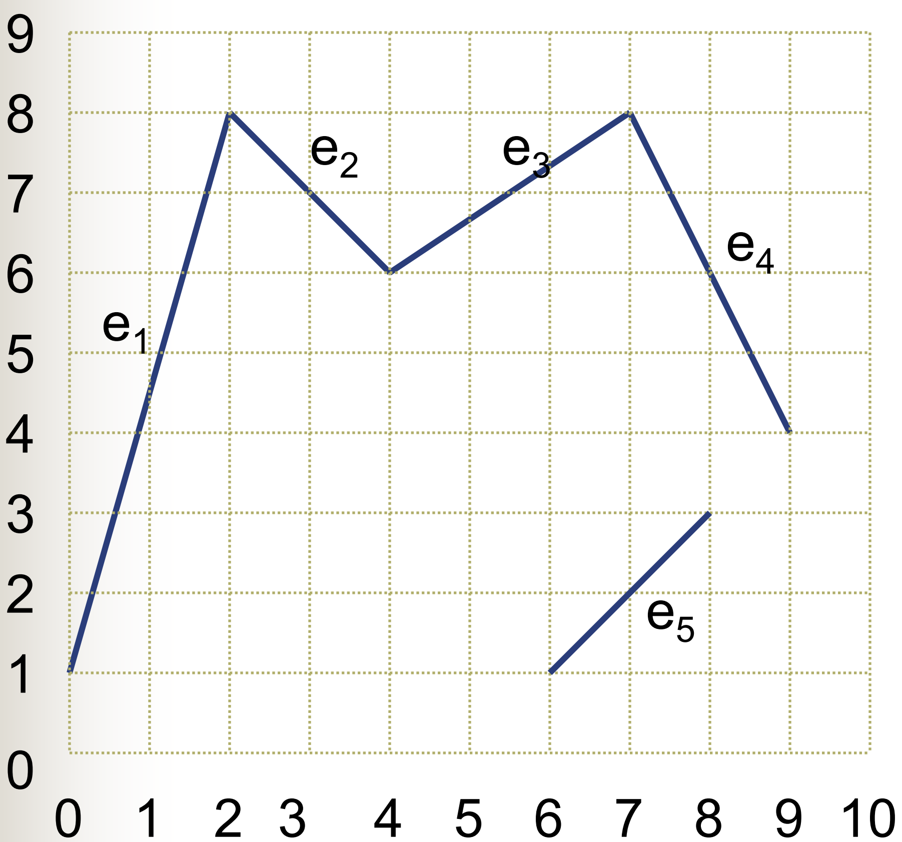
scanline 4 intercepta e_1,e_4,e_5 em 0.9, 9, 9 respectiva/e
- linha intercepta duas arestas, considera só a superior, i.e. e_4

Casos a tratar: 3. Arestas horizontais



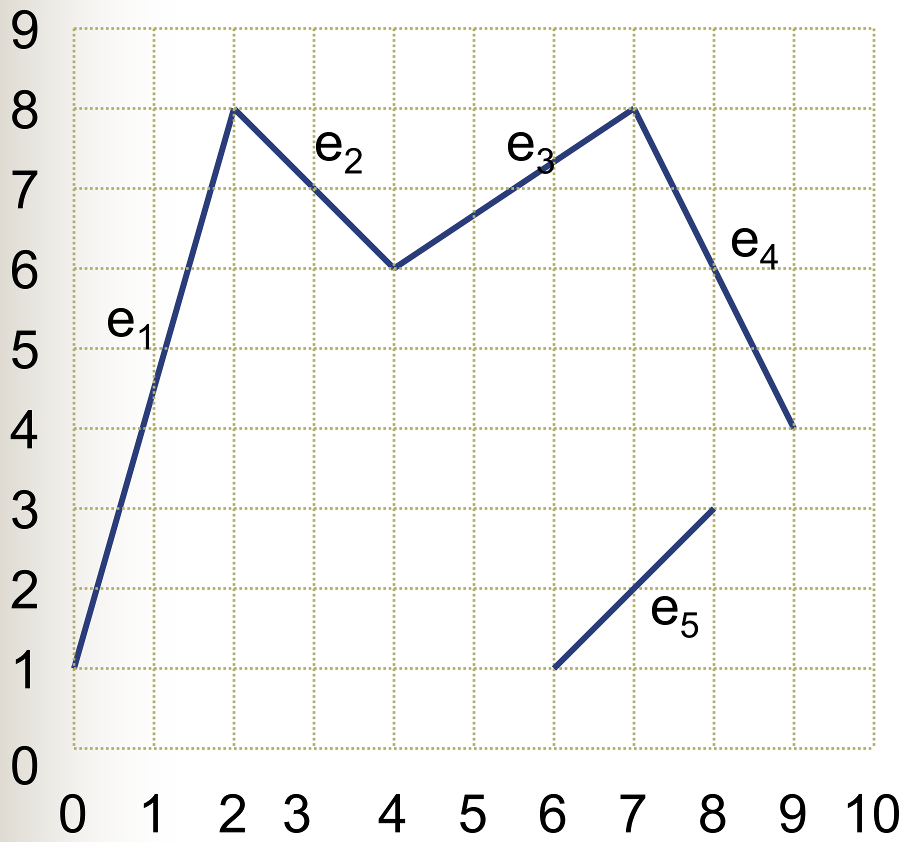
Uma aresta horizontal, como e_6 pode ser Ignorada: vai ser traçada automaticamente

Pré-Processamento do Polígono



Removemos
e_6, e encurtamos
e_5 em uma *scanline*

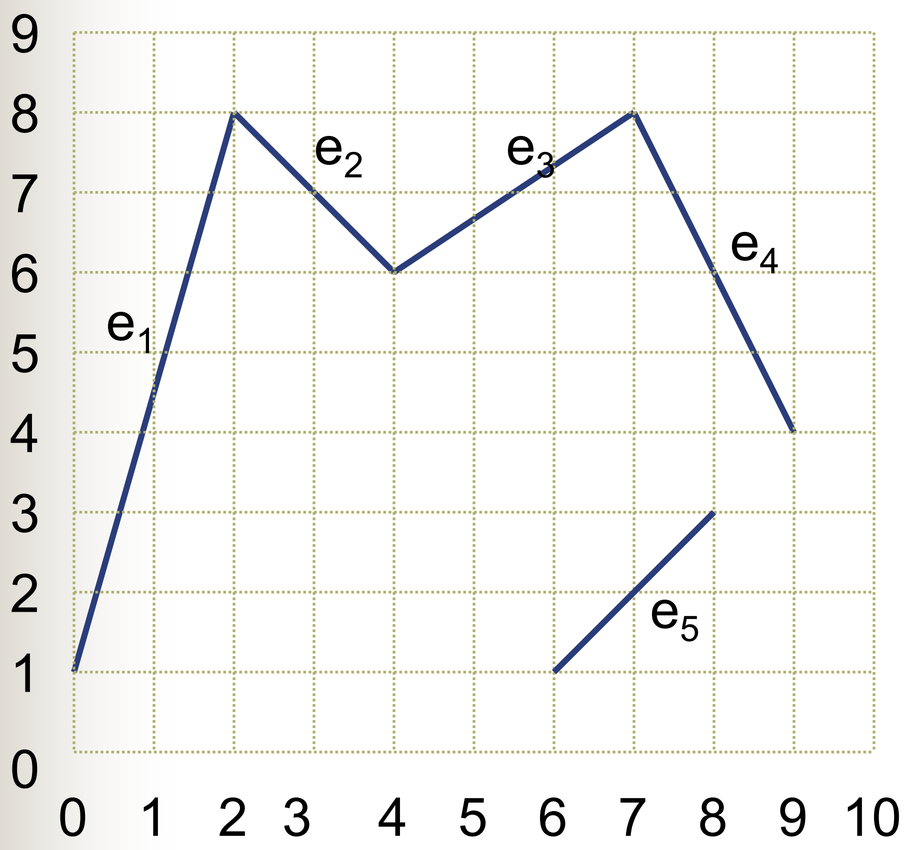
Otimização



Acelerar: saber aonde começam e terminam as arestas: e.g., e_1 vai da *scanline* 1 até a *scanline* 8; e_2 da *scanline* 8 até a 6

Assim, sabemos que arestas testar para cada *scanline*

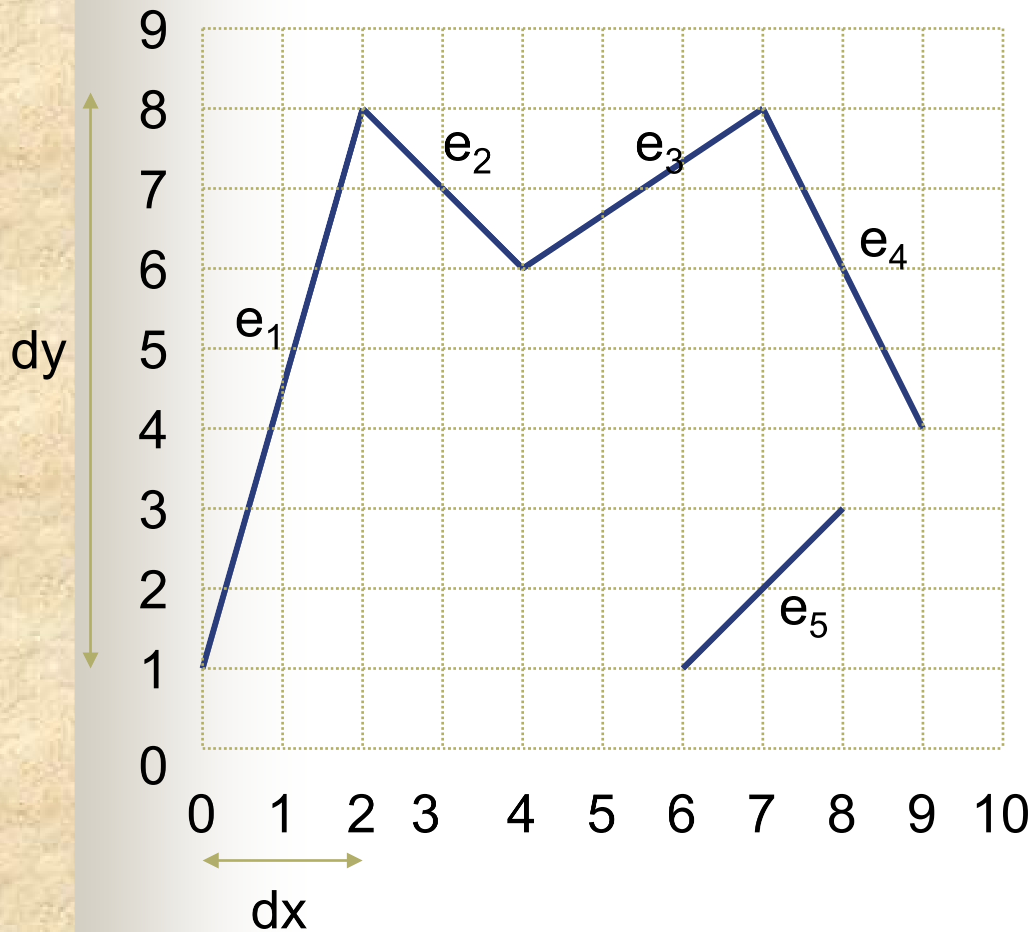
Conjuntos Ordenados de Arestas



Ordenar as arestas pelo seu ponto de mínimo; cada *scanline* é associada a um conjunto de arestas, ordenado segundo a coord. x da intersecção

0:
1: e_1,e_5
2:
3:
4: e_4
5:
6: e_2,e_3
7:
8:
9:

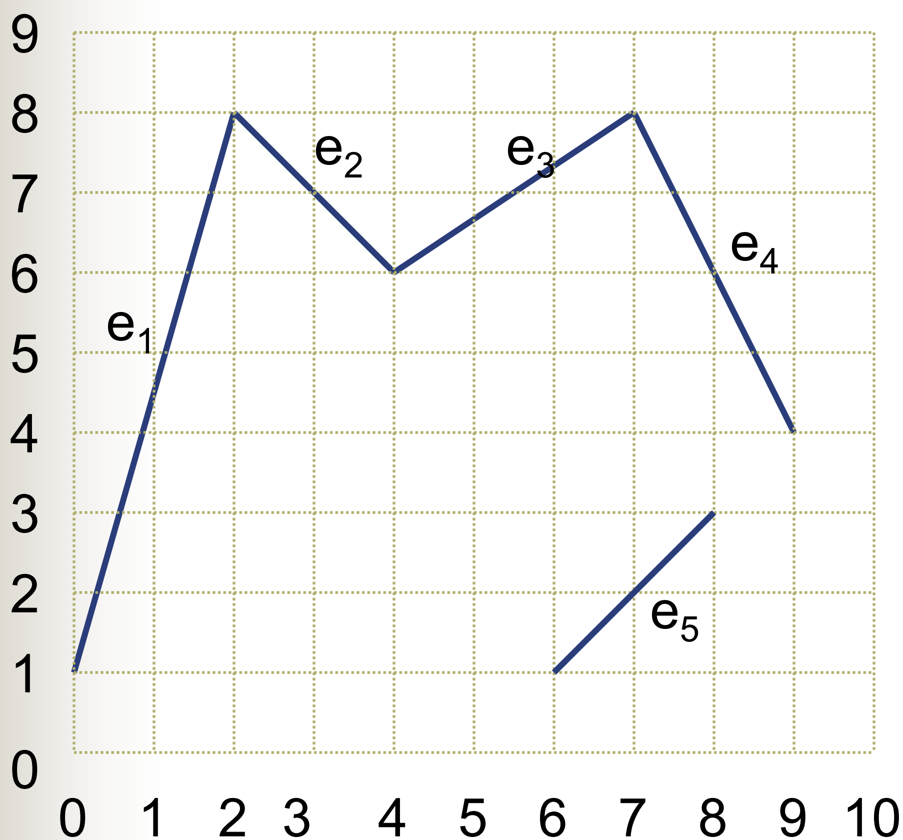
Otimização - Coerência



Ex.: observe e_1
Assuma intersecção com *scanline 1* conhecida, então, a intersecção com *scanline* 2 é:

$x_2 = x_1 + 1/m$
i.e.: $x_2 = x_1 + dx / dy$
i.e.: $x_2 = 0 + 2 / 7 = 2/7$

Tabela de Arestas



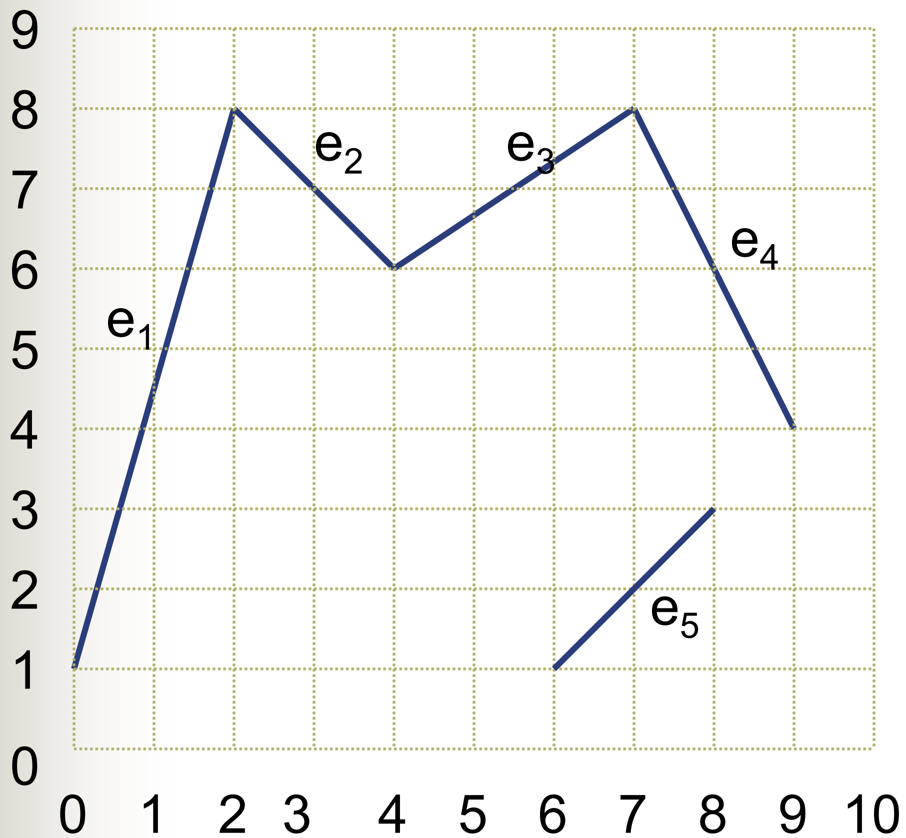
Útil armazenar informação sobre arestas em uma tabela:

e	prim x	max y	dx	dy
1	0	8	2	7
2	4	8	-2	2
3				
4				
5				

Estrutura de dados

- Tabela de arestas global: descreve as arestas do polígono a serem processadas explicitamente
  - Um conjunto para cada linha de varredura
  - Mantém-se ao longo de todo o processamento
- Tabela de arestas ativas (as que estão sendo interceptadas pela linha de varredura corrente)
  - É atualizada a medida que a varredura da cena prossegue

Algoritmo de Preenchimento



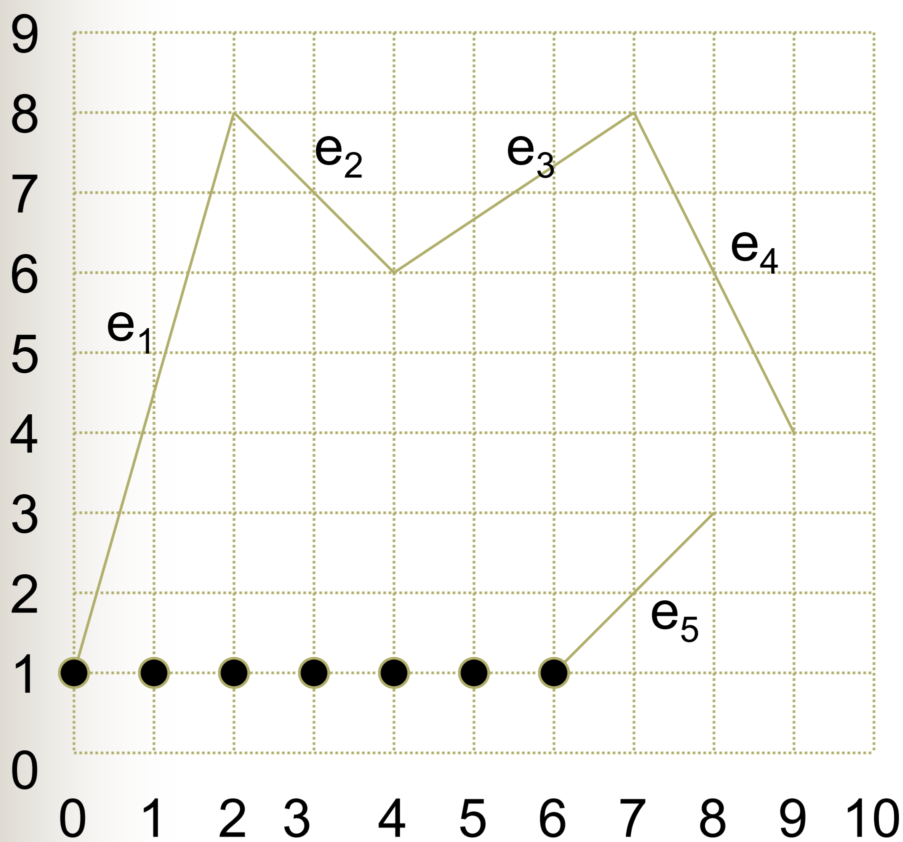
(1) Faça y = 0

(2) Verifique conjunto-y

(3) Insira as arestas necessárias na **active edge table (AET)**

(4) SE AET vazia ENTÃO
y = y + 1, GO TO (2)

Tabela de Arestas Ativas

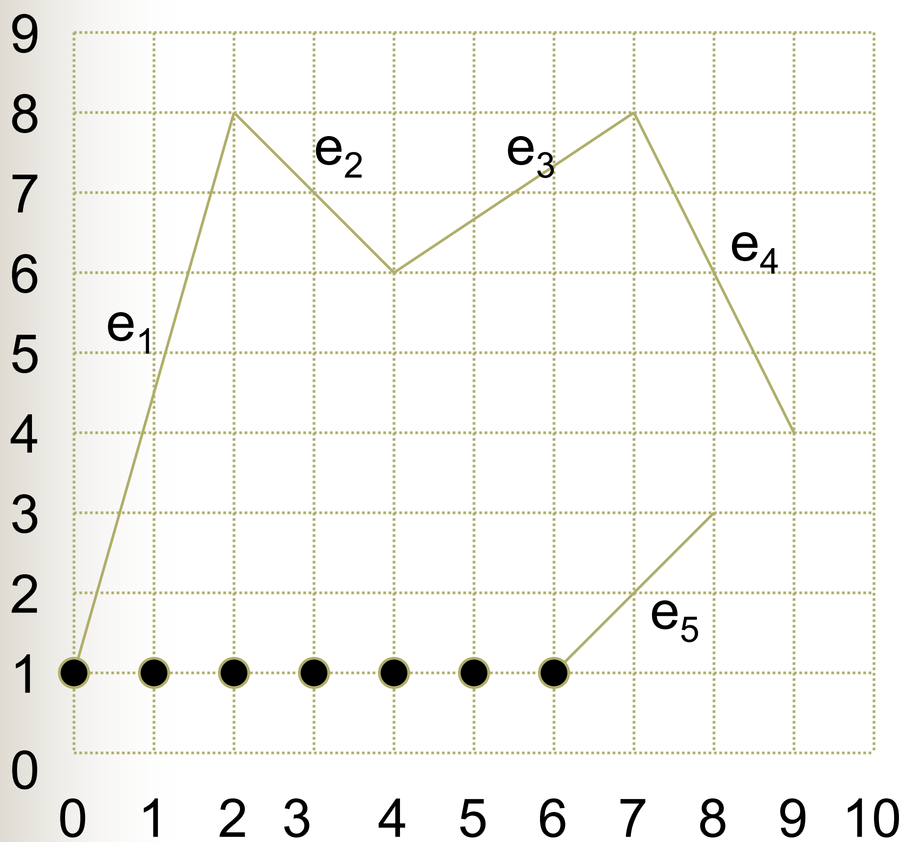


y = 1

e	x	max y	dx	dy
1	0	8	2	7
5	6	3	2	2

(5) Ordene por valores-x e preencha entre pares sucessivos

Tabela de Arestas Ativas



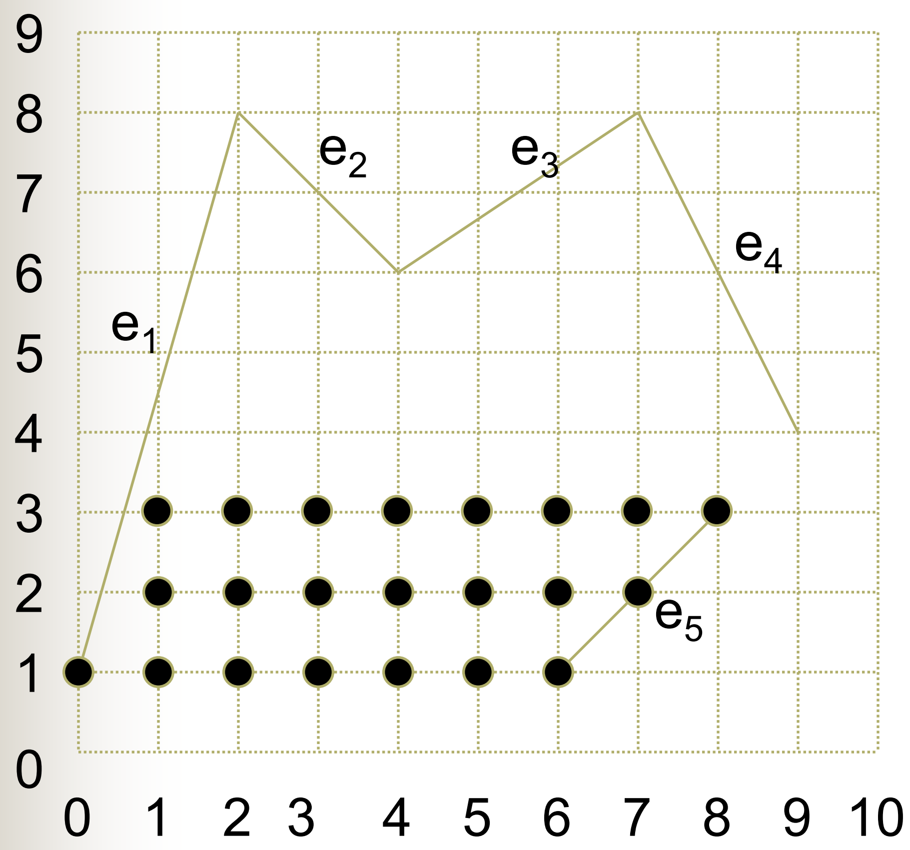
(6) FAÇA y = y+1; remova da AET arestas com max y < y

(7) Incrementa x de 1/m (dx/dy)

e	x	max y	dx	dy
1	2/7	8	2	7
5	7	3	2	2

(8) Retorna para (2)

Próxima Scanline



A cada estágio do algoritmo, um (ou vários) *span*(s) de pixels são traçados

Eficiência – aritmética inteira

- É interessante usar apenas aritmética inteira –
  - Requer uma coluna extra na AET para acumularmos separadamente a parte inteira da parte fracionária de 1/m para atualizar x

Eficiência – aritmética inteira

Primeira intersecção é em x=0, e a inclinação da aresta é 7/2 – i.e. dy=7, dx=2

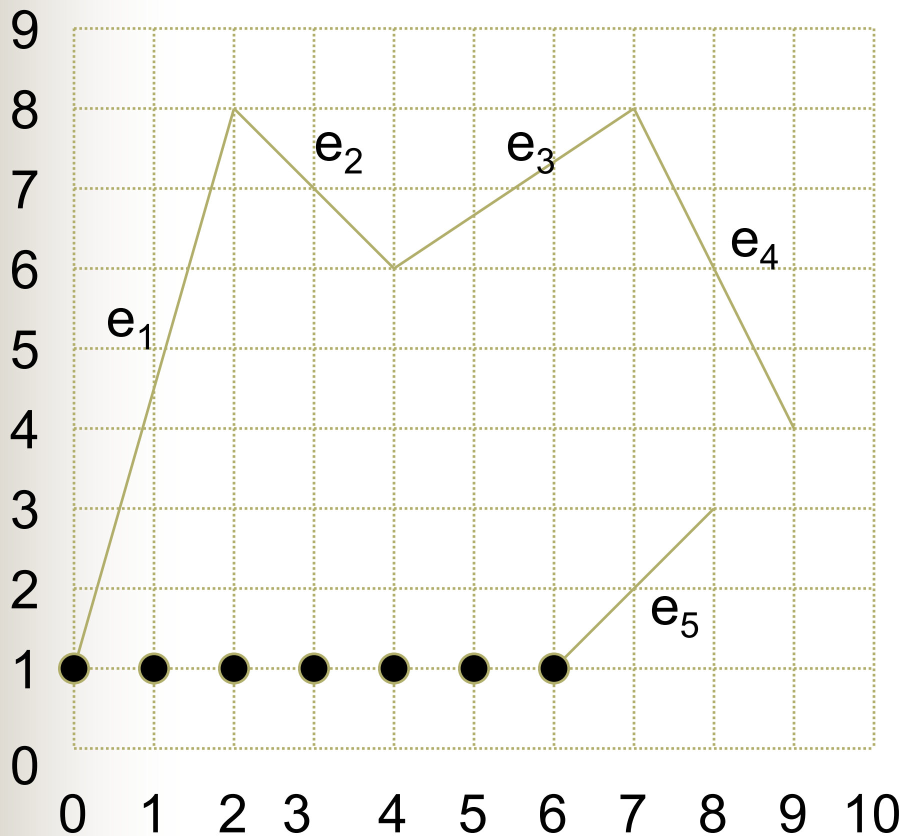
Os próximos pontos de intersecção são:

0 2/7 4/7 6/7 8/7 etc.

Obtemos esses valores simplesmente somando dx a cada estágio, até que dy é atingido, então dx é reduzido de dy:

0 2 4 6 1 (8-7) etc.

Tabela de Arestas Ativas Modificada



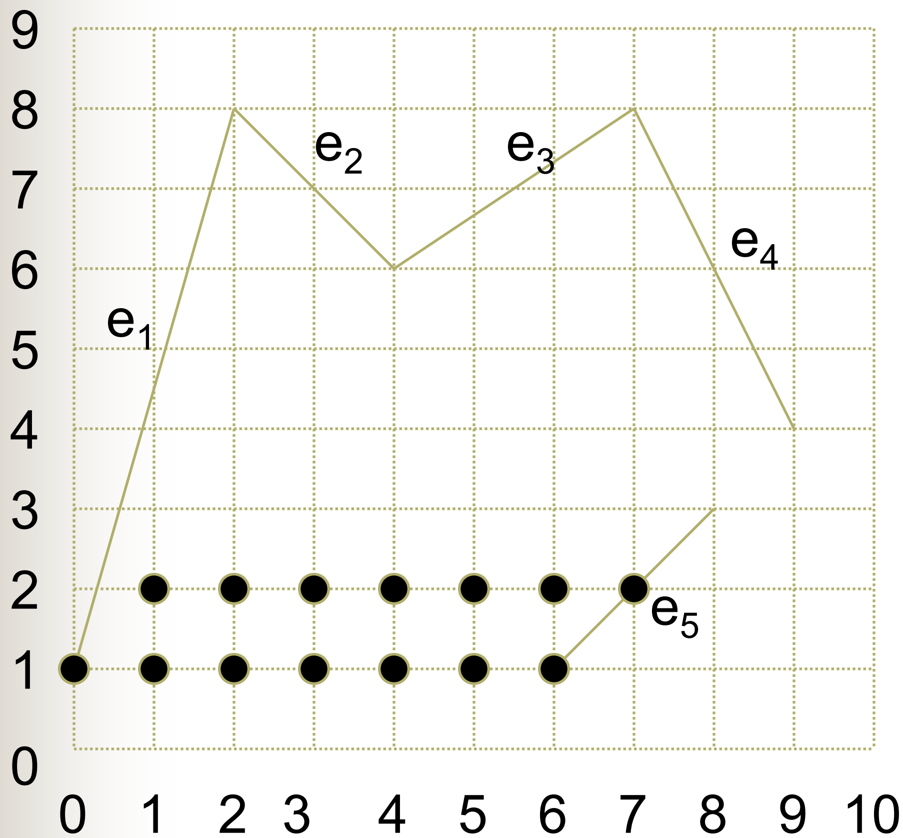
y = 1

e	x	max y	dx	dy
1	0	8	2	7
5	6	3	2	2

torna-se

e	x int	x frac	max y	dx	dy
1	0	0	8	2	7
5	6	0	3	2	2

Eficiência – aritmética inteira



No passo (7), ao invés de incrementar x por dx/dy, incrementamos x-frac de dx; sempre que x-frac excede dy somamos 1 a x-int & reduzimos x-frac de dy

y = 2

e	x int	x frac	max y	dx	dy
1	0	2	8	2	7
5	7	0	3	2	2

Bibliografia

- Hearn & Baker, 4.10
- http://www.cs.rit.edu/~icss571/filling/example.html