Geometria Computacional

Sweep line: definição

Prof. Edson Alves

Faculdade UnB Gama

Sumário

Sweep line algorithms

Definição

- Também denominados *sweep plane algorithms*, estes algoritmos formam um paradigma de solução de problemas geométricos

Definição

- Também denominados *sweep plane algorithms*, estes algoritmos formam um paradigma de solução de problemas geométricos
- A ideia principal é mover uma reta (geralmente em sentido vertical) pelo espaço e parar em pontos chaves

Definição

- Também denominados *sweep plane algorithms*, estes algoritmos formam um paradigma de solução de problemas geométricos
- A ideia principal é mover uma reta (geralmente em sentido vertical) pelo espaço e parar em pontos chaves
- A cada parada, são processados os pontos ou que interceptam a reta ou que estejam em sua vizinhança

Definição

- Também denominados *sweep plane algorithms*, estes algoritmos formam um paradigma de solução de problemas geométricos
- A ideia principal é mover uma reta (geralmente em sentido vertical) pelo espaço e parar em pontos chaves
- A cada parada, são processados os pontos ou que interceptam a reta ou que estejam em sua vizinhança
- Outra interpretação possível é interpretar um conjunto de pontos como eventos, e estes eventos devem ser processados em ordem crescente de coordenada x ou y

Definição

- Também denominados *sweep plane algorithms*, estes algoritmos formam um paradigma de solução de problemas geométricos
- A ideia principal é mover uma reta (geralmente em sentido vertical) pelo espaço e parar em pontos chaves
- A cada parada, são processados os pontos ou que interceptam a reta ou que estejam em sua vizinhança
- Outra interpretação possível é interpretar um conjunto de pontos como eventos, e estes eventos devem ser processados em ordem crescente de coordenada x ou y
- Exemplos de algoritmos desta categoria são o algoritmo de Graham e a cadeia monótona de Andrew

Interseção de intervalos

Interseção de intervalos

- Um problema que pode ser resolvido usando o paradigma *sweep line* é o de contabilizar, dentre um conjunto de intervalos $I_i = [a_i, b_i)$, o tamanho do maior subconjunto S destes intervalos tal que $I_j \cap I_k \neq \emptyset$ para todo par $I_j, I_k \in S$

Interseção de intervalos

- Um problema que pode ser resolvido usando o paradigma *sweep line* é o de contabilizar, dentre um conjunto de intervalos $I_i = [a_i, b_i)$, o tamanho do maior subconjunto S destes intervalos tal que $I_j \cap I_k \neq \emptyset$ para todo par $I_j, I_k \in S$
- Uma aplicação prática deste problema seria: cada intervalo representa o início e o fim de um espetáculo que acontecerá em determinado dia. Qual é o número máximo de espetáculos que acontecerão simultaneamete?

Interseção de intervalos

- Um problema que pode ser resolvido usando o paradigma *sweep line* é o de contabilizar, dentre um conjunto de intervalos $I_i = [a_i, b_i)$, o tamanho do maior subconjunto S destes intervalos tal que $I_j \cap I_k \neq \emptyset$ para todo par $I_j, I_k \in S$
- Uma aplicação prática deste problema seria: cada intervalo representa o início e o fim de um espetáculo que acontecerá em determinado dia. Qual é o número máximo de espetáculos que acontecerão simultaneamete?
- A solução é criar, para cada intervalo, dois eventos: um evento de ínicio do espetáculo $(a_i, 1)$ e um evento de final $(b_i, 0)$

Interseção de intervalos

- Um problema que pode ser resolvido usando o paradigma *sweep line* é o de contabilizar, dentre um conjunto de intervalos $I_i = [a_i, b_i)$, o tamanho do maior subconjunto S destes intervalos tal que $I_j \cap I_k \neq \emptyset$ para todo par $I_j, I_k \in S$
- Uma aplicação prática deste problema seria: cada intervalo representa o início e o fim de um espetáculo que acontecerá em determinado dia. Qual é o número máximo de espetáculos que acontecerão simultaneamete?
- A solução é criar, para cada intervalo, dois eventos: um evento de ínicio do espetáculo $(a_i, 1)$ e um evento de final $(b_i, 0)$
- Uma vez ordenados estes eventos em ordem lexicográfica (primeiro por coordenada x, depois por coordenada y), basta processar todos eles, um por vez

Interseção de intervalos

- Um evento de início incrementa o número de eventos em andamento, o evento de fim decrementa

Interseção de intervalos

- Um evento de início incrementa o número de eventos em andamento, o evento de fim decrementa
- Com a representação de eventos escolhida, os intervalos $[a, b)$ e $[b, c)$ não tem interseção

Interseção de intervalos

- Um evento de início incrementa o número de eventos em andamento, o evento de fim decrementa
- Com a representação de eventos escolhida, os intervalos $[a, b)$ e $[b, c)$ não tem interseção
- A complexidade deste algoritmo é $O(N \log N)$, por conta da ordenação, pois cada ponto será processado uma única vez

Interseção de intervalos

- Um evento de início incrementa o número de eventos em andamento, o evento de fim decrementa
- Com a representação de eventos escolhida, os intervalos $[a, b)$ e $[b, c)$ não tem interseção
- A complexidade deste algoritmo é $O(N \log N)$, por conta da ordenação, pois cada ponto será processado uma única vez
- A representação dos eventos pode ser modificada, de tal modo que é possível identificar quais são os intervalos simultâneos

Interseção de intervalos

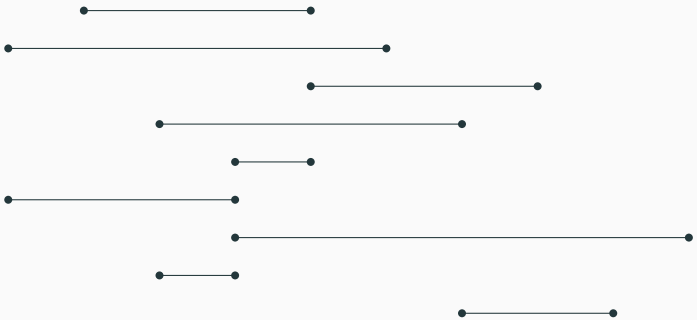
- Um evento de início incrementa o número de eventos em andamento, o evento de fim decrementa
- Com a representação de eventos escolhida, os intervalos $[a, b)$ e $[b, c)$ não tem interseção
- A complexidade deste algoritmo é $O(N \log N)$, por conta da ordenação, pois cada ponto será processado uma única vez
- A representação dos eventos pode ser modificada, de tal modo que é possível identificar quais são os intervalos simultâneos
- Basta fazer (a_i, i) e $(b_i, -i)$ e manter os índices dos intervalos em exibição em um conjunto, removendo-os a cada evento de encerramento

Interseção de intervalos

- Um evento de início incrementa o número de eventos em andamento, o evento de fim decrementa
- Com a representação de eventos escolhida, os intervalos $[a, b)$ e $[b, c)$ não tem interseção
- A complexidade deste algoritmo é $O(N \log N)$, por conta da ordenação, pois cada ponto será processado uma única vez
- A representação dos eventos pode ser modificada, de tal modo que é possível identificar quais são os intervalos simultâneos
- Basta fazer (a_i, i) e $(b_i, -i)$ e manter os índices dos intervalos em exibição em um conjunto, removendo-os a cada evento de encerramento
- Veja que, nesta represenção, os intervalos devem ser numerados a partir de 1, pois o zero geraria ambiguidade

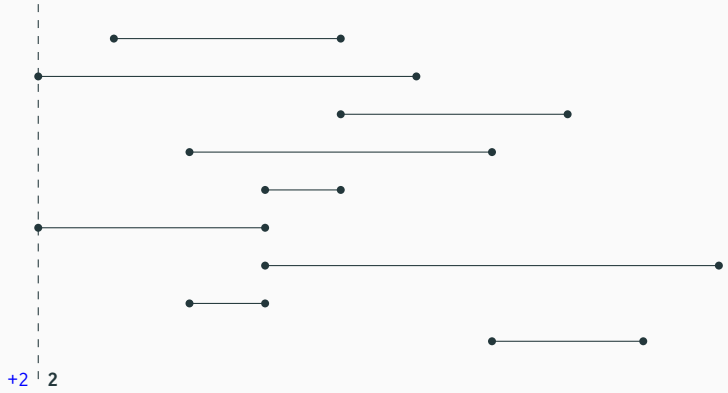
Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



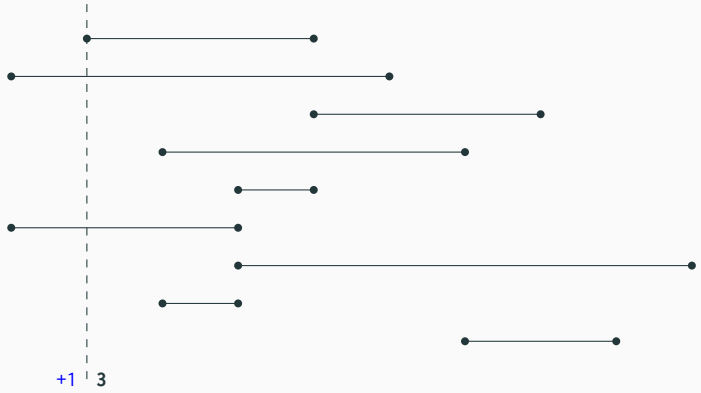
Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



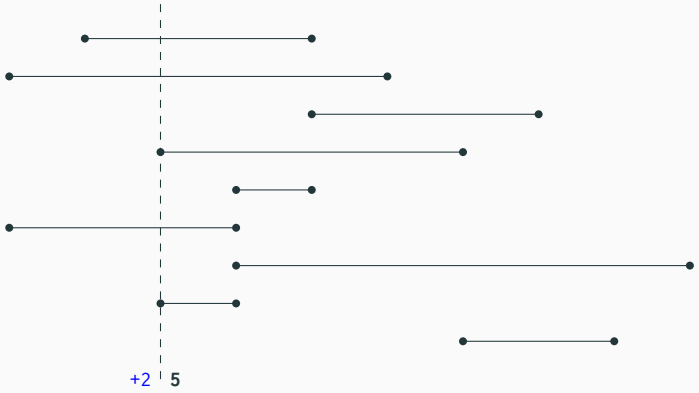
Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



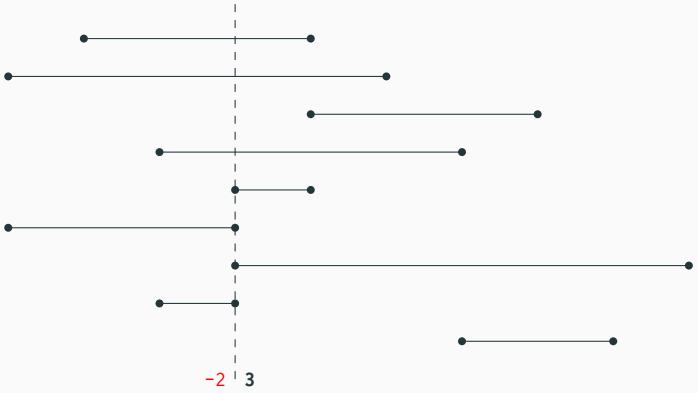
Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



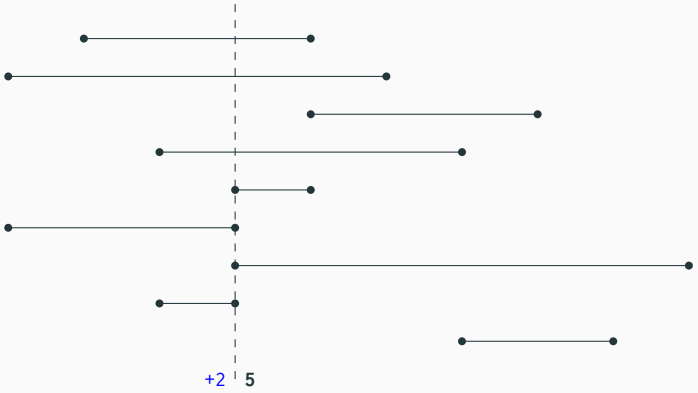
Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



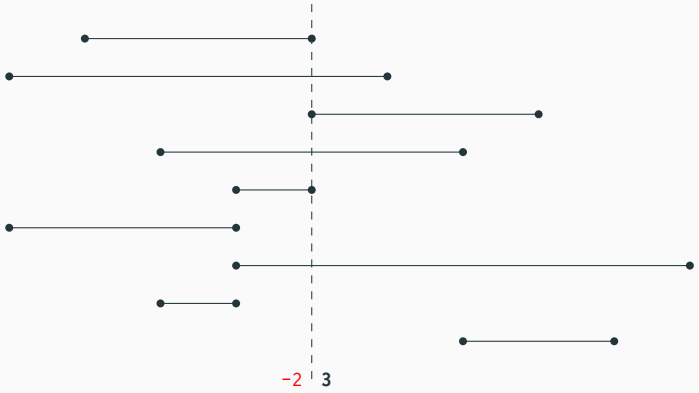
Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



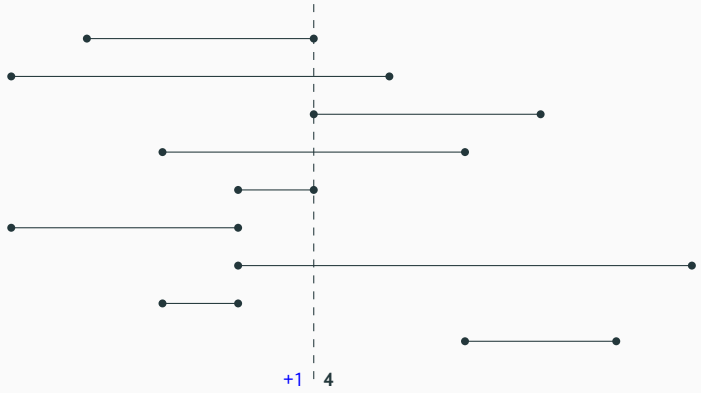
Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



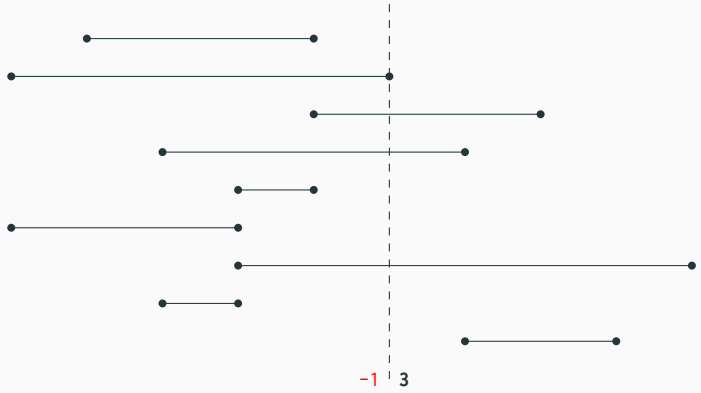
Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



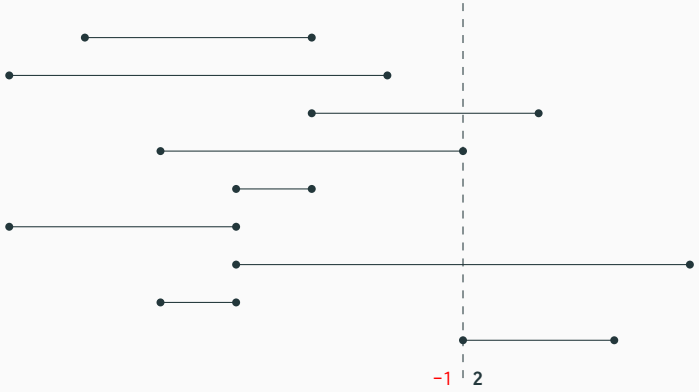
Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



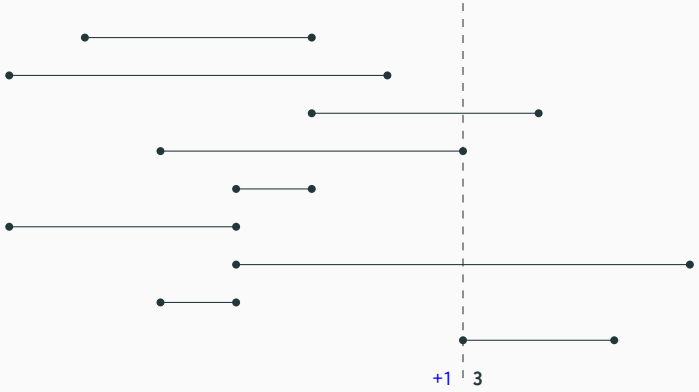
Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



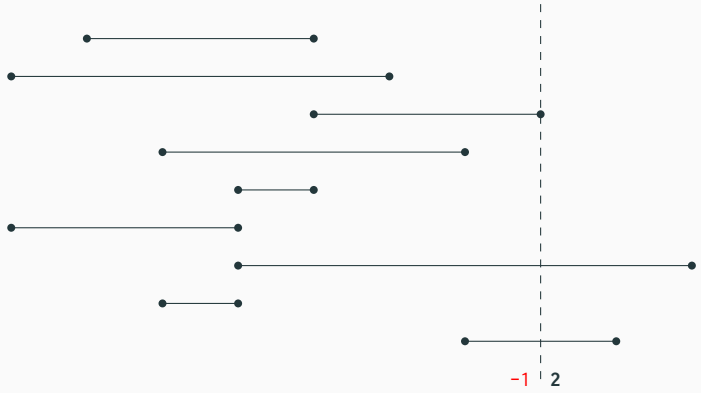
Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



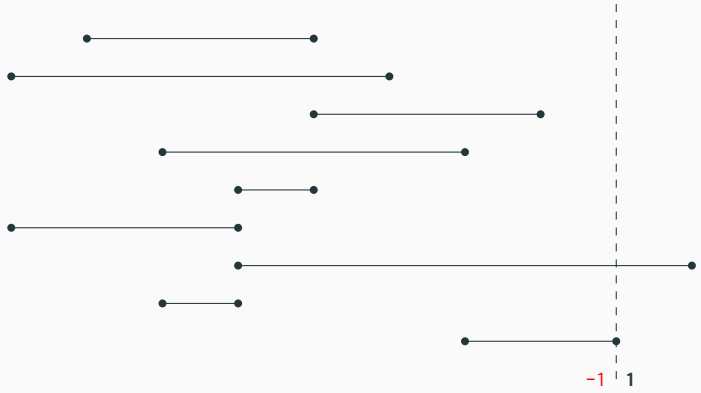
Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



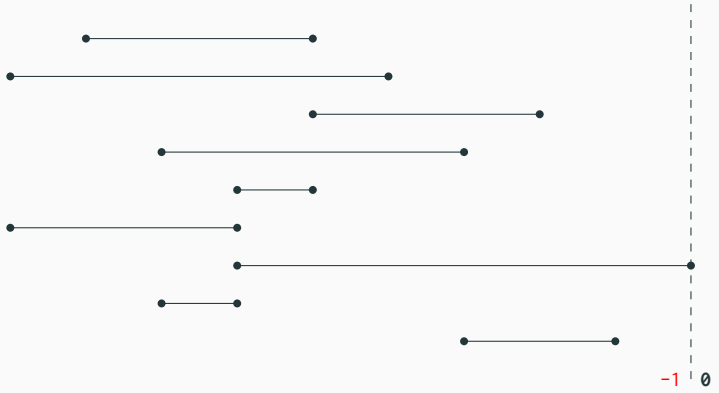
Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



Visualização do algoritmo de interseção de intervalos

Intervalos: $(2,5), (1,6), (5,8), (3,7), (4,5), (1,4), (4,10), (3,4), (7,9)$



Implementação da interseção de intervalos

```cpp
#include <bits/stdc++.h>

using namespace std;
using interval = pair<int, int>;

vector<int> max_intersection(const vector<interval>& is)
{
    using event = pair<int, int>;

    vector<event> es;

    for (size_t i = 0; i < is.size(); ++i)
    {
        auto [a, b] = is[i];

        es.emplace_back(a, i + 1);      // Evento de início
        es.emplace_back(b, -(i + 1));   // Evento de fim
    }
```

Implementação da interseção de intervalos

```
20      sort(es.begin(), es.end());
21
22      set<int> active, max_set;
23
24      for (const auto& [_, i] : es)
25      {
26          if (i > 0)
27              active.emplace(i);
28          else
29              active.erase(-i);
30
31          if (active.size() >= max_set.size())
32              max_set = active;
33      }
34
35      return { max_set.begin(), max_set.end() };
36  }
```

Pontos de interseção

Pontos de interseção entre segmentos verticais e horizontais

- Considere o seguinte problema: seja S o conjunto de N segmentos de reta, horizontais ou verticais. Determine o conjuntos dos pontos de interseção entre estes segmentos

Pontos de interseção entre segmentos verticais e horizontais

- Considere o seguinte problema: seja S o conjunto de N segmentos de reta, horizontais ou verticais. Determine o conjuntos dos pontos de interseção entre estes segmentos
- Um ponto estará no conjunto das interseções se ele pertence, simultaneamente, a um segmento vertical e a um segmento horizontal de S

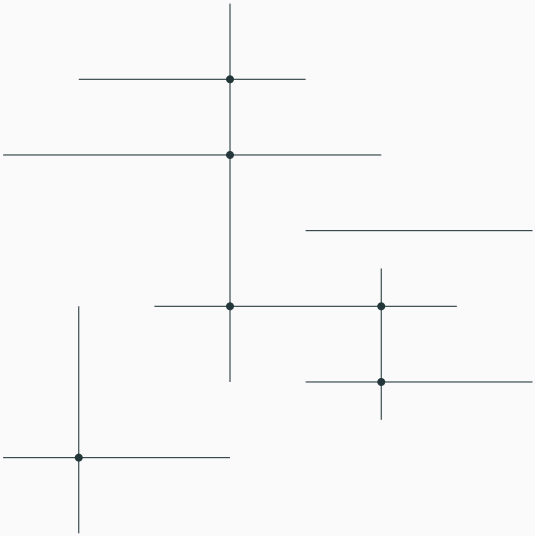
Pontos de interseção entre segmentos verticais e horizontais

- Considere o seguinte problema: seja S o conjunto de N segmentos de reta, horizontais ou verticais. Determine o conjuntos dos pontos de interseção entre estes segmentos
- Um ponto estará no conjunto das interseções se ele pertence, simultaneamente, a um segmento vertical e a um segmento horizontal de S
- Assuma que os segmentos horizontais não tenham interseção entre si; e o mesmo para os segmentos verticais

Pontos de interseção entre segmentos verticais e horizontais

- Considere o seguinte problema: seja S o conjunto de N segmentos de reta, horizontais ou verticais. Determine o conjuntos dos pontos de interseção entre estes segmentos
- Um ponto estará no conjunto das interseções se ele pertence, simultaneamente, a um segmento vertical e a um segmento horizontal de S
- Assuma que os segmentos horizontais não tenham interseção entre si; e o mesmo para os segmentos verticais
- O algoritmo de força bruta compara um segmento com todos os demais, tendo complexidade $O(N^2)$

Visualização do problema de pontos de interseção



Solução utilizado sweep line

- O *sweep line* pode ser utilizado em uma solução $O(N \log N)$

Solução utilizado sweep line

- O *sweep line* pode ser utilizado em uma solução $O(N \log N)$
- Os intervalos podem gerar três tipos de eventos:

Solução utilizado sweep line

- O *sweep line* pode ser utilizado em uma solução $O(N \log N)$
- Os intervalos podem gerar três tipos de eventos:
  1. Início de um intervalo horizontal com altura y

Solução utilizado sweep line

- O *sweep line* pode ser utilizado em uma solução $O(N \log N)$
- Os intervalos podem gerar três tipos de eventos:
  1. Início de um intervalo horizontal com altura y
  2. Intervalo vertical $[a, b]$

Solução utilizado sweep line

- O *sweep line* pode ser utilizado em uma solução $O(N \log N)$
- Os intervalos podem gerar três tipos de eventos:
  1. Início de um intervalo horizontal com altura y
  2. Intervalo vertical $[a, b]$
  3. Fim de um intervalo horizontal com altura y

Solução utilizado sweep line

- O *sweep line* pode ser utilizado em uma solução $O(N \log N)$
- Os intervalos podem gerar três tipos de eventos:
  1. Início de um intervalo horizontal com altura y
  2. Intervalo vertical $[a, b]$
  3. Fim de um intervalo horizontal com altura y
- Cada evento do tipo 1 deve armazenar as coordenadas y dos intervalos abertos

Solução utilizado sweep line

- O *sweep line* pode ser utilizado em uma solução $O(N \log N)$
- Os intervalos podem gerar três tipos de eventos:
  1. Início de um intervalo horizontal com altura y
  2. Intervalo vertical $[a, b]$
  3. Fim de um intervalo horizontal com altura y
- Cada evento do tipo 1 deve armazenar as coordenadas y dos intervalos abertos
- Em cada evento do tipo 2 deve ser contabilizados quantos valores estão no conjunto e que pertencem ao intervalo $[a, b]$

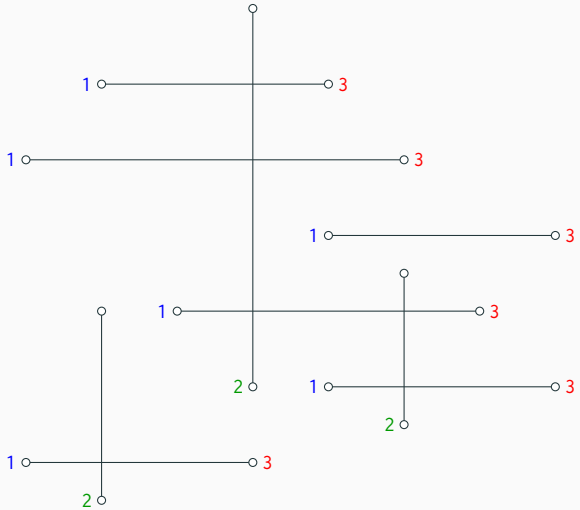
Solução utilizado sweep line

- O *sweep line* pode ser utilizado em uma solução $O(N \log N)$
- Os intervalos podem gerar três tipos de eventos:
  1. Início de um intervalo horizontal com altura y
  2. Intervalo vertical $[a, b]$
  3. Fim de um intervalo horizontal com altura y
- Cada evento do tipo 1 deve armazenar as coordenadas y dos intervalos abertos
- Em cada evento do tipo 2 deve ser contabilizados quantos valores estão no conjunto e que pertencem ao intervalo $[a, b]$
- Para determinar esta quantia os valores y devem ser armazenados em uma árvore de Fenwick

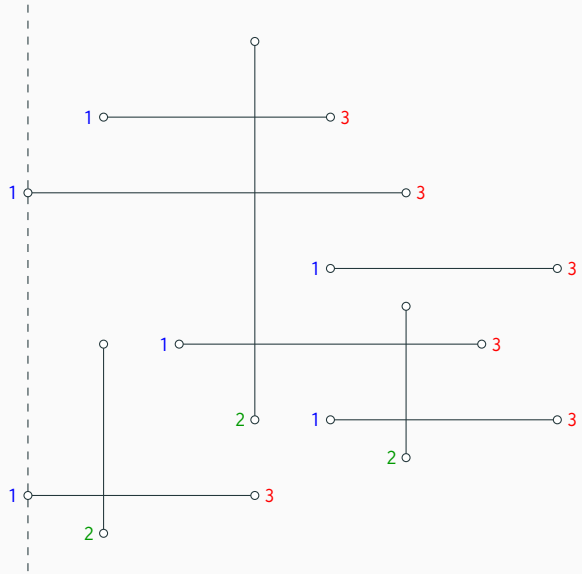
Solução utilizado sweep line

- O *sweep line* pode ser utilizado em uma solução $O(N \log N)$
- Os intervalos podem gerar três tipos de eventos:
  1. Início de um intervalo horizontal com altura y
  2. Intervalo vertical $[a, b]$
  3. Fim de um intervalo horizontal com altura y
- Cada evento do tipo 1 deve armazenar as coordenadas y dos intervalos abertos
- Em cada evento do tipo 2 deve ser contabilizados quantos valores estão no conjunto e que pertencem ao intervalo $[a, b]$
- Para determinar esta quantia os valores y devem ser armazenados em uma árvore de Fenwick
- Cada evento do tipo 3 remove a coordenada y do intervalo a ser fechado

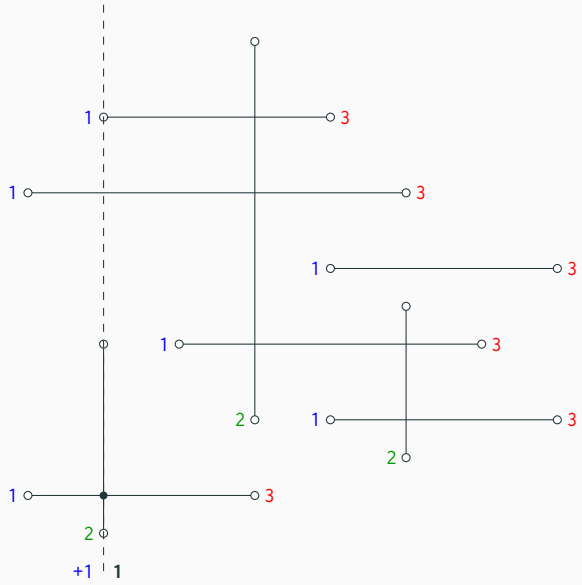
Visualização do algoritmo de pontos de interseção



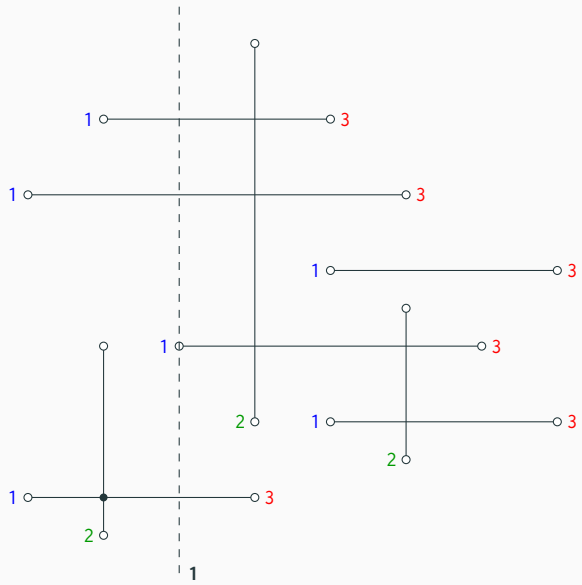
Visualização do algoritmo de pontos de interseção



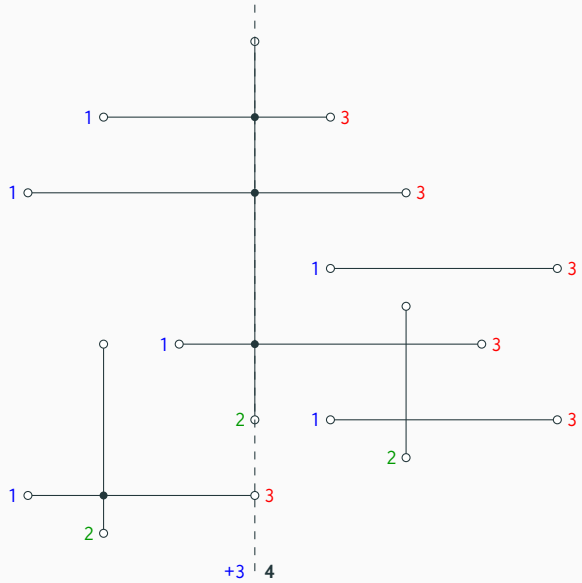
Visualização do algoritmo de pontos de interseção



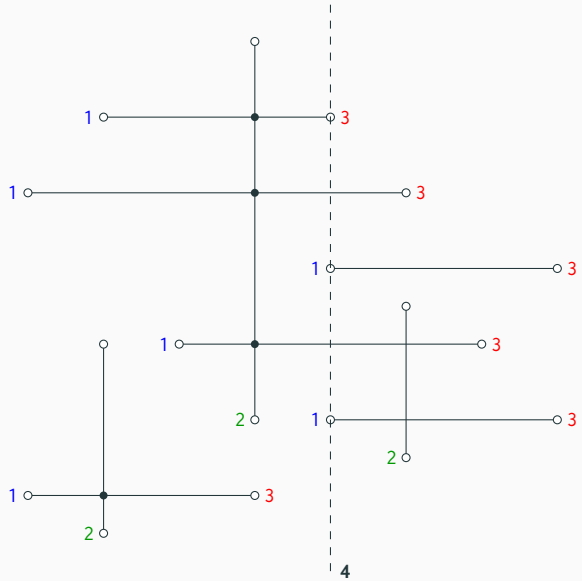
Visualização do algoritmo de pontos de interseção



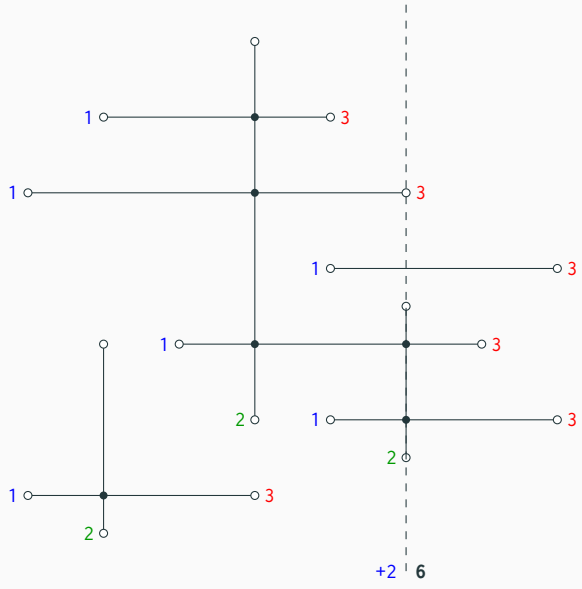
Visualização do algoritmo de pontos de interseção



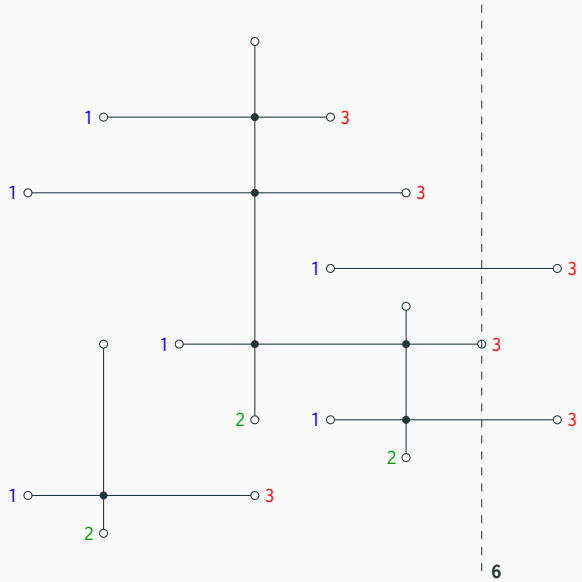
Visualização do algoritmo de pontos de interseção



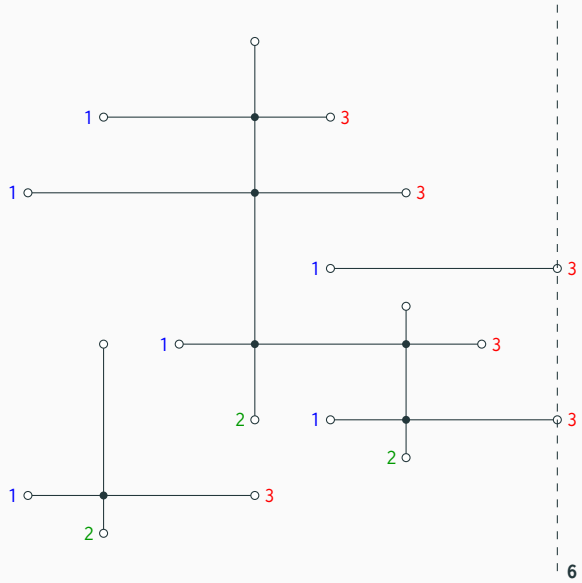
Visualização do algoritmo de pontos de interseção



Visualização do algoritmo de pontos de interseção



Visualização do algoritmo de pontos de interseção



Implementação da contagem dos pontos de interseção

```cpp
#include <bits/stdc++.h>

using namespace std;

class BITree {
private:
    vector<int> ts;
    size_t N;

    int LSB(int n) { return n & (-n); }

    int RSQ(int i)
    {
        int sum = 0;

        while (i >= 1)
        {
            sum += ts[i];
            i -= LSB(i);
        }
```

Implementação da contagem dos pontos de interseção

```
22          return sum;
23      }
24
25  public:
26      BITree(size_t n) : ts(n + 1, 0), N(n) {}
27
28      int RSQ(int i, int j)
29      {
30          return RSQ(j) - RSQ(i - 1);
31      }
32
33      void add(size_t i, int x)
34      {
35          if (i == 0)
36              return;
```

Implementação da contagem dos pontos de interseção

```
38          while (i <= N)
39          {
40              ts[i] += x;
41              i += LSB(i);
42          }
43      }
44  };
45
46  struct Point { int x, y; };
47
48  struct Interval { Point A, B; };
49
50  int index(const vector<int>& hs, int value)
51  {
52      auto it = lower_bound(hs.begin(), hs.end(), value);
53
54      return it - hs.begin() + 1;      // Contagem inicia em 1
55  }
```

Implementação da contagem dos pontos de interseção

```
57 int intersections(const vector<Interval>& is)
58 {
59     struct Event {
60         int type, x;
61         size_t idx;
62 
63         bool operator<(const Event& e) const
64         {
65             if (x != e.x)
66                 return x < e.x;
67 
68             if (type != e.type)
69                 return type < e.type;
70 
71             return idx < e.idx;
72         }
73     };
```

Implementação da contagem dos pontos de interseção

```
75      vector<Event> es;
76      set<int> ys;        // Conjunto para compressão das coordenadas
77
78      for (size_t i = 0; i < is.size(); ++i)
79      {
80          auto I = is[i];
81
82          ys.emplace(I.A.y);
83          ys.emplace(I.B.y);
84
85          auto xmin = min(I.A.x, I.B.x), xmax = max(I.A.x, I.B.x);
86
87          if (I.A.x == I.B.x)     // Vertical
88              es.emplace_back(2, xmin, i);
89          else                    // Horizontal
90          {
91              es.emplace_back(1, xmin, i);
92              es.emplace_back(3, xmax, i);
93          }
94      }
```

Implementação da contagem dos pontos de interseção

```
96      sort(es.begin(), es.end());
97
98      vector<int> hs(ys.begin(), ys.end());   // Mapa de compressão
99      BITree ft(hs.size());
100     auto total = 0;
101
102     for (const auto& event : es)
103     {
104         auto I = is[event.idx];
105
106         switch (event.type) {
107         case 1:
108             {
109                 auto y = index(hs, I.A.y);
110                 ft.add(y, 1);
111             }
112             break;
```

Implementação da contagem dos pontos de interseção

```
114         case 2:
115             {
116                 auto ymin = min(index(hs, I.A.y), index(hs, I.B.y));
117                 auto ymax = max(index(hs, I.A.y), index(hs, I.B.y));
118 
119                 total += ft.RSQ(ymin, ymax);
120             }
121             break;
122 
123         default:
124             {
125                 auto y = index(hs, I.B.y);
126                 ft.add(y, -1);
127             }
128         }
129     }
130 
131     return total;
132 }
```

Referências

1. **HALIM**, Felix; **HALIM**, Steve. *Competitive Programming 3*, 2010.
2. **LAAKSONEN**, Antti. *Competitive Programmer's Handbook*, 2018.
3. **De BERG**, Mark; **CHEONG**, Otfried. *Computational Geometry: Algorithms and Applications*, 2008.
4. Wikipedia. Sweep line algorithm, acesso em 22/05/2019.