UNIVERSIDADE FEDERAL DE SERGIPE

DEPARTAMENTO DE COMPUTAÇÃO

Geometria computacional

Projeto e Análise de Algoritmos

Bruno Prado

Departamento de Computação / UFS

Introdução

- O que é geometria computacional?
  - É um campo da computação dedicada a resolver problemas geométricos em diversas áreas

Introdução

- O que é geometria computacional?
  - É um campo da computação dedicada a resolver problemas geométricos em diversas áreas
    - Astronomia

Introdução

- O que é geometria computacional?
  - É um campo da computação dedicada a resolver problemas geométricos em diversas áreas
    - Astronomia
    - Processamento de imagens

Introdução

- O que é geometria computacional?
  - É um campo da computação dedicada a resolver problemas geométricos em diversas áreas
    - Astronomia
    - Processamento de imagens
    - Computação gráfica

Introdução

- O que é geometria computacional?
  - É um campo da computação dedicada a resolver problemas geométricos em diversas áreas
    - Astronomia
    - Processamento de imagens
    - Computação gráfica
    - Projeto de circuitos integrados

Introdução

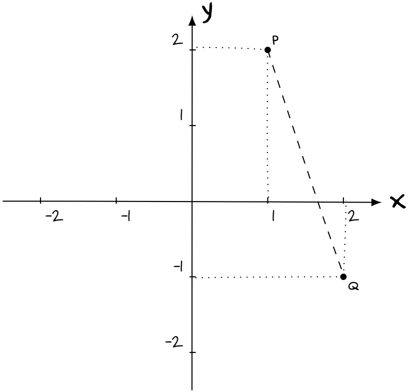
- O que é geometria computacional?
  - É um campo da computação dedicada a resolver problemas geométricos em diversas áreas
    - Astronomia
    - Processamento de imagens
    - Computação gráfica
    - Projeto de circuitos integrados
    - Modelagem molecular

Introdução

- O que é geometria computacional?
  - É um campo da computação dedicada a resolver problemas geométricos em diversas áreas
    - Astronomia
    - Processamento de imagens
    - Computação gráfica
    - Projeto de circuitos integrados
    - Modelagem molecular
    - Indústria têxtil
    - ...

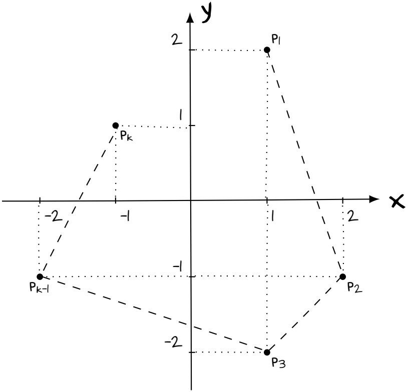
Introdução

- Definições básicas
  - Ponto: é uma representação por coordenadas do espaço geométrico (cartesiano, polar, esférico, ...)
  - Linha: definida por 2 pontos distintos p e q sendo denotado como segmento $p - q$



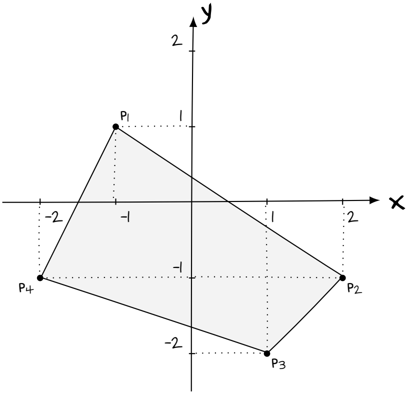
Introdução

- Definições básicas
  - Caminho: consistem em uma sequência de n pontos $p_1, p_2, \ldots, p_n$ e pelos segmentos de linha $p_1 - p_2$, $p_2 - p_3, \ldots, p_{k-1} - p_k$ que conectam estes pontos



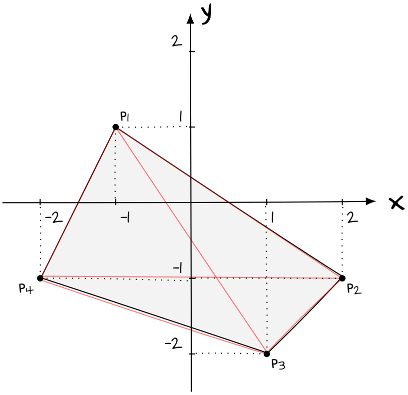
Introdução

- Definições básicas
  - Polígono simples: é um caminho fechado em que as linhas não se interceptam no plano formado



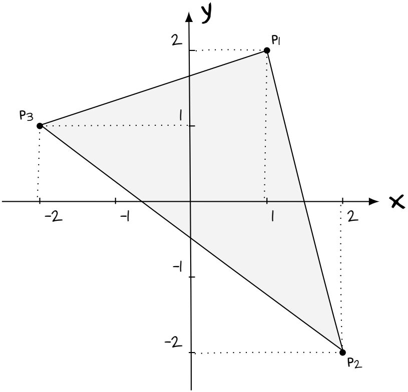
Introdução

- Definições básicas
  - Polígono convexo: qualquer linha que interconecta os pontos do polígono está contida em seu plano interno ou nos limites de suas arestas



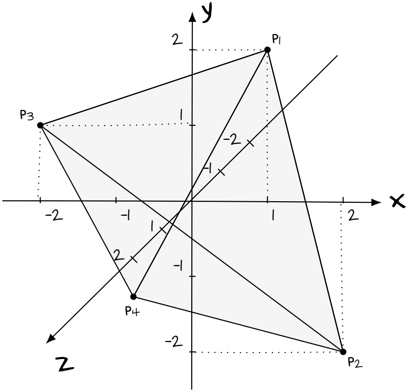
Introdução

- Representação dos objetos geométricos
  - Polígono convexo com 2 dimensões
  - Caminho fechado $p_1 - p_2, p_2 - p_3, p_3 - p_1$
    - $p_i = (x_i, y_i)$
    - $x_i, y_i \in \mathbb{R}$



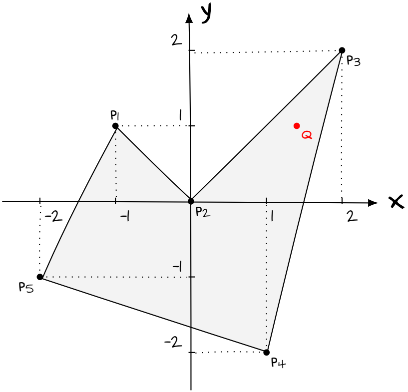
Introdução

- Representação dos objetos geométricos
  - Polígono convexo com 3 dimensões
  - Caminho fechado $p_1 - p_2, \ldots, p_3 - p_4$
    - $p_i = (x_i, y_i, z_i)$
    - $x_i, y_i, z_i \in \mathbb{R}$



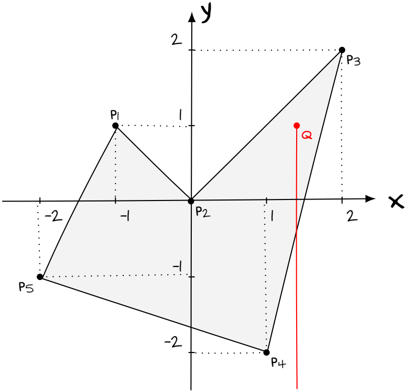
Geometria computacional

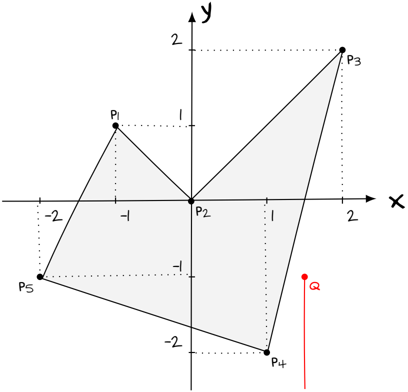
- Considerando um polígono simples P e um ponto q, determine se q está dentro ou fora do polígono P



Geometria computacional

- Considerando um polígono simples P e um ponto q, determine se q está dentro ou fora do polígono P

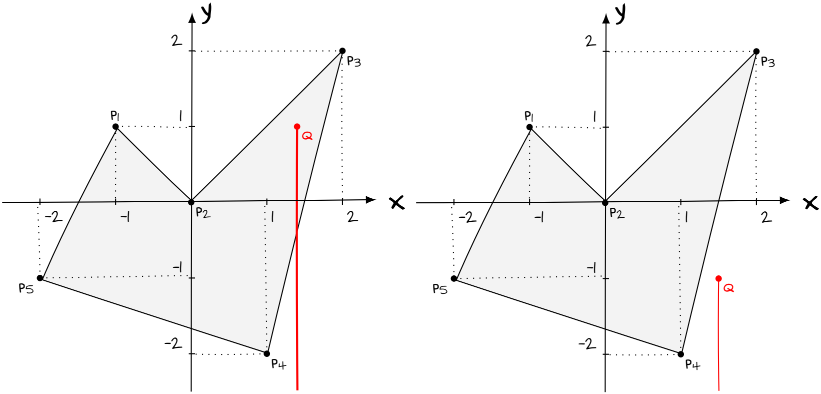




Ideia do algoritmo: traçar retas do ponto q
e contar as interseções com as arestas do polígono

Geometria computacional

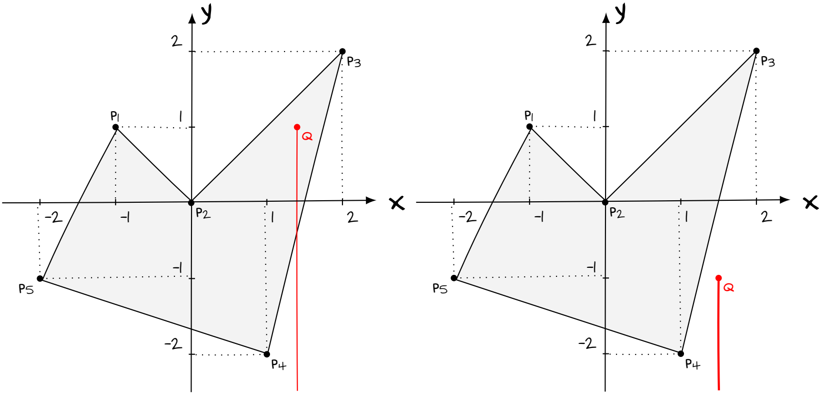
- Considerando um polígono simples P e um ponto q, determine se q está dentro ou fora do polígono P



Dentro do polígono (ímpar)

Geometria computacional

- Considerando um polígono simples P e um ponto q, determine se q está dentro ou fora do polígono P

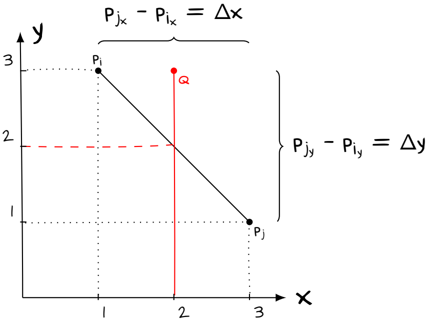


Fora do polígono (par)

Geometria computacional

- Considerando um polígono simples P e um ponto q, determine se q está dentro ou fora do polígono P

```c
// Função de interseção
float intersecao(float qx, poligono* P, int32_t i,
    int32_t j) {
     return (qx - P->x[i]) * (P->y[j] - P->y[i]) /
         (P->x[j] - P->x[i]) + P->y[i];
}
```

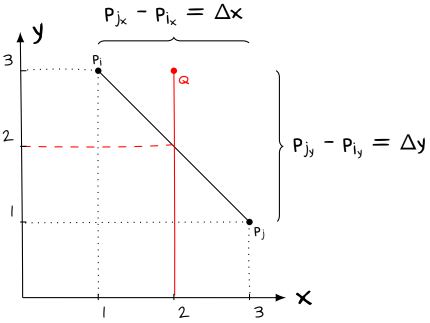


$$y = ax + B$$

Geometria computacional

- Considerando um polígono simples P e um ponto q, determine se q está dentro ou fora do polígono P

```c
// Função de interseção
float intersecao(float qx, poligono* P, int32_t i,
    int32_t j) {
     return (qx - P->x[i]) * (P->y[j] - P->y[i]) /
         (P->x[j] - P->x[i]) + P->y[i];
}
```



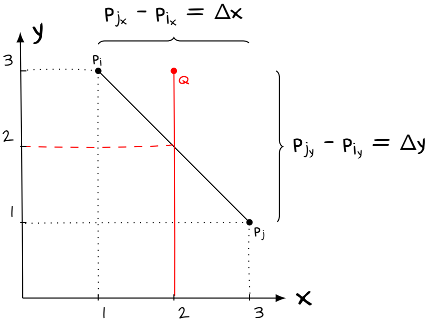
$$y = ax + B$$

$$a = \frac{\Delta y}{\Delta x}, \; B = P_{i_y}$$

Geometria computacional

- Considerando um polígono simples P e um ponto q, determine se q está dentro ou fora do polígono P

```c
// Função de interseção
float intersecao(float qx, poligono* P, int32_t i,
    int32_t j) {
     return (qx - P->x[i]) * (P->y[j] - P->y[i]) /
         (P->x[j] - P->x[i]) + P->y[i];
}
```



$$y = ax + B$$

$$a = \frac{\Delta y}{\Delta x},\ B = P_{i_y}$$

$$Q_y = \frac{\Delta y}{\Delta x}(Q_x - P_{i_x}) + P_{i_y}$$

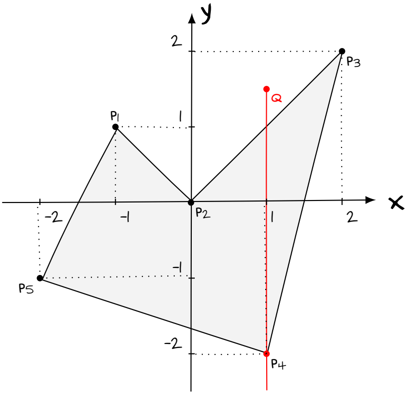
Geometria computacional

- Considerando um polígono simples *P* e um ponto *q*, determine se *q* está dentro ou fora do polígono *P*

```c
// Função de pertinência do ponto q no polígono P
uint8_t q_contido_P(ponto* q, poligono* P) {
    // Resultado boleano
    uint8_t r = 0;
    // Iterando nos segmentos do polígono
    for(int32_t i = 0, j = 1; i < P->n - 1; i++, j++)
        {
        // Checando se qx está entre o segmento i-j
        if((q->x > P->x[i] && q->x <= P->x[j]) ||
            (q->x > P->x[j] && q->x <= P->x[i]))
            // Verificando a interseção do ponto
            if(q->y > intersecao(q->x, P, i, j))
                // Complementando o resultado
                r = !r;
    }
    // Retornando resultado
    return r;
}
```

Geometria computacional

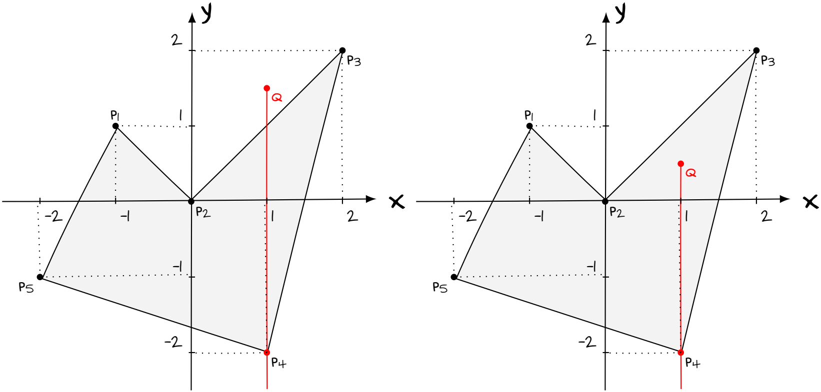
- Considerando um polígono simples P e um ponto q, determine se q está dentro ou fora do polígono P



Casos especiais

Geometria computacional

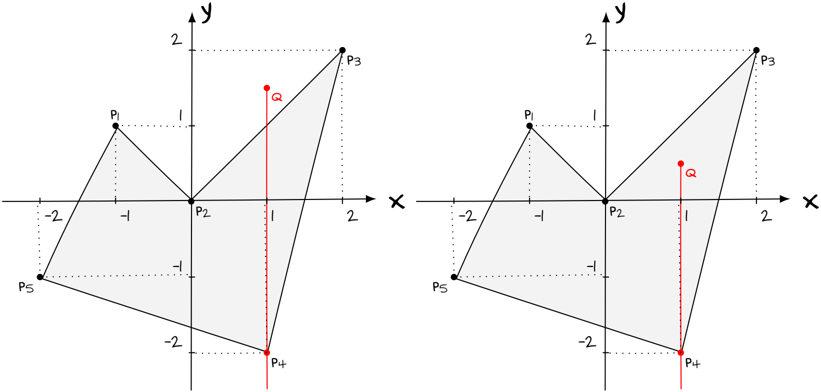
- Considerando um polígono simples P e um ponto q, determine se q está dentro ou fora do polígono P



Espaço e tempo $O(n)$

Geometria computacional

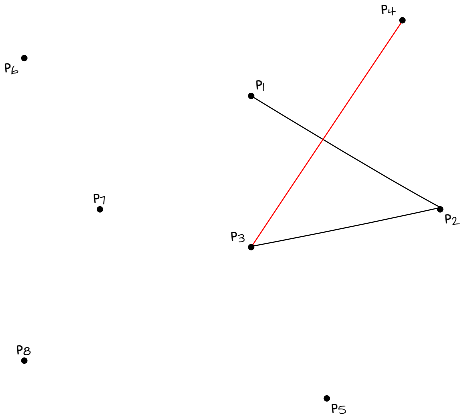
- Considerando um polígono simples P e um ponto q, determine se q está dentro ou fora do polígono P



Aplicações: astronomia, visão computacional, projeto de circuitos integrados, ...

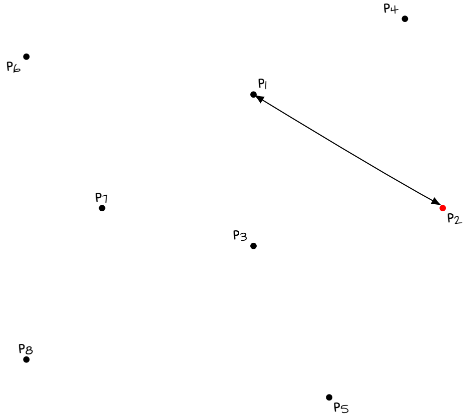
Geometria computacional

- Realizar a construção de um polígono simples P a partir de um conjunto com n pontos no plano



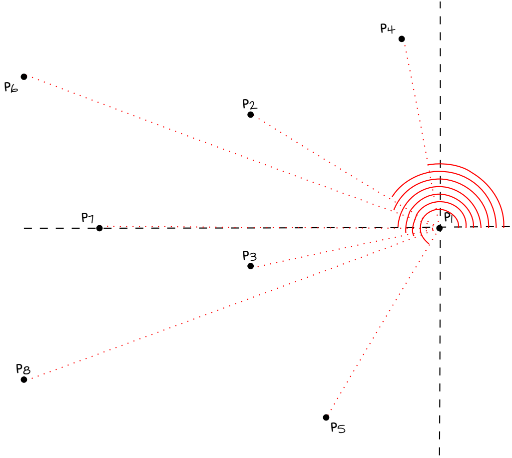
Geometria computacional

- Realizar a construção de um polígono simples P a partir de um conjunto com n pontos no plano
  - O ponto mais extremo do plano, com a maior valor de x e menor valor de y, é definido como referência



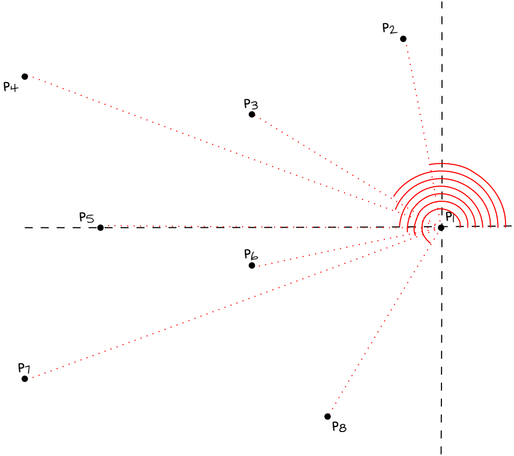
Geometria computacional

- Realizar a construção de um polígono simples P a partir de um conjunto com n pontos no plano
  - Para a ordenação dos pontos são calculados os ângulos de todos os pontos a partir de p_1



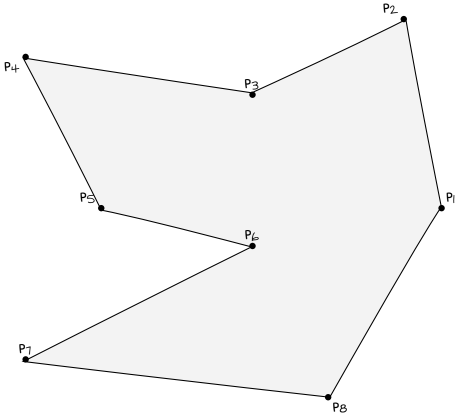
Geometria computacional

- Realizar a construção de um polígono simples *P* a partir de um conjunto com *n* pontos no plano
  - Para a ordenação dos pontos são calculados os ângulos de todos os pontos a partir de p_1



Geometria computacional

- Realizar a construção de um polígono simples P a partir de um conjunto com n pontos no plano
  - Para a ordenação dos pontos são calculados os ângulos de todos os pontos a partir de p_1



Geometria computacional

- Realizar a construção de um polígono simples P a partir de um conjunto com n pontos no plano

```
// Procedimento de construção do polígono simples
void poligono_simples(poligono* P, ponto* p[],
    uint32_t n) {
    // Definição do ponto extremo p1
    ponto_extremo(P, p, n);
    // Iterando nos pontos do polígono
    for(uint32_t i = 1; i < n; i++)
        // Calculando ângulo com relação a p1
        calcular_angulo(P, p, i);
    // Ordenação dos pontos pelo ângulo
    ordenar_por_angulo(P);
}
```

Geometria computacional

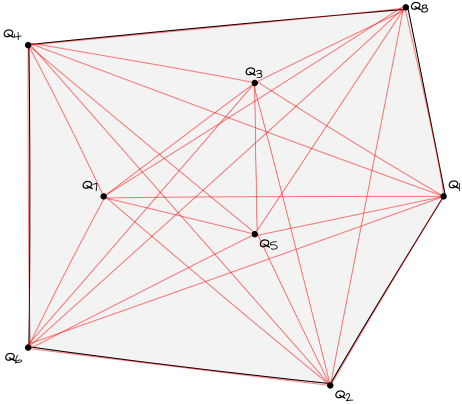
- Realizar a construção de um polígono simples P a partir de um conjunto com n pontos no plano

```
// Procedimento de construção do polígono simples
void poligono_simples(poligono* P, ponto* p[],
    uint32_t n) {
    // Definição do ponto extremo p1
    ponto_extremo(P, p, n);
    // Iterando nos pontos do polígono
    for(uint32_t i = 1; i < n; i++)
        // Calculando ângulo com relação a p1
        calcular_angulo(P, p, i);
    // Ordenação dos pontos pelo ângulo
    ordenar_por_angulo(P);
}
```

Espaço $O(n)$ e tempo $O(n + n + n \log n) = O(n \log n)$

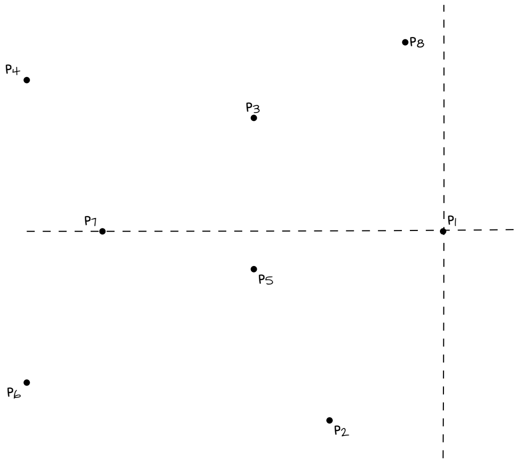
Geometria computacional

- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos



Geometria computacional

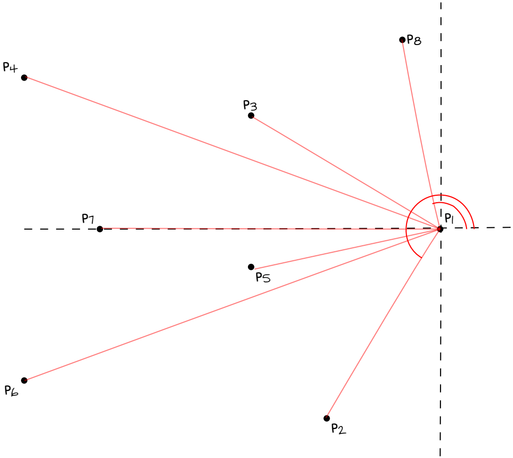
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Abordagem incremental



Definição do ponto extremo

Geometria computacional

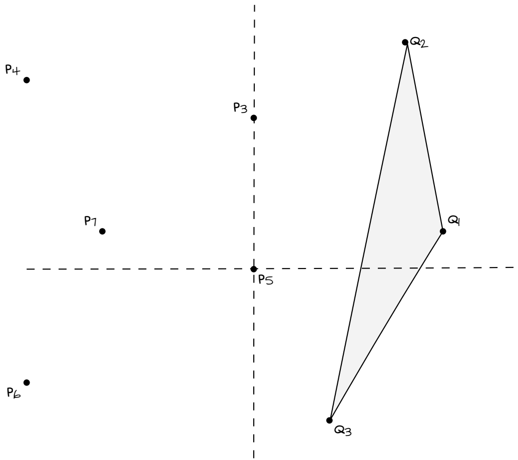
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Abordagem incremental



Seleção dos pontos com maior e menor grau

Geometria computacional

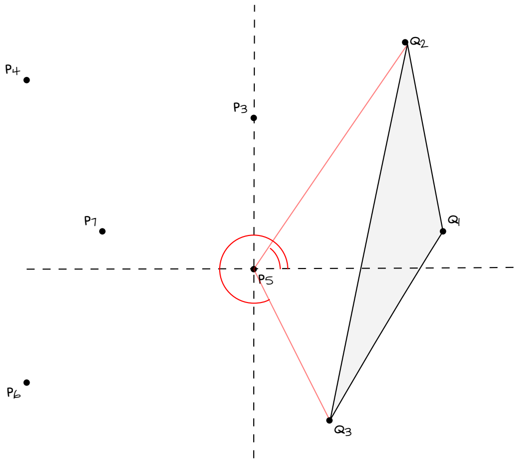
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Abordagem incremental



Definição do ponto extremo fora do polígono

Geometria computacional

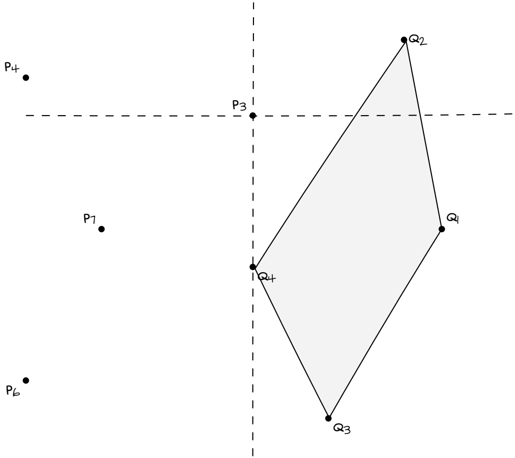
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Abordagem incremental



Seleção dos pontos com maior e menor grau

Geometria computacional

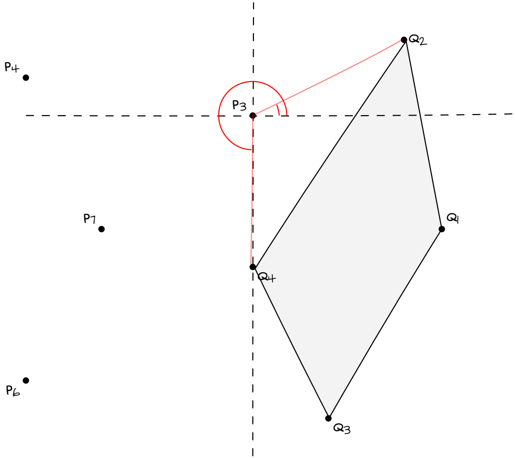
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Abordagem incremental



Definição do ponto extremo fora do polígono

Geometria computacional

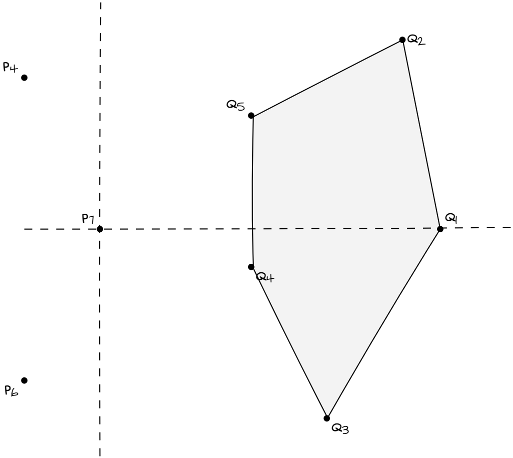
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Abordagem incremental



Seleção dos pontos com maior e menor grau

Geometria computacional

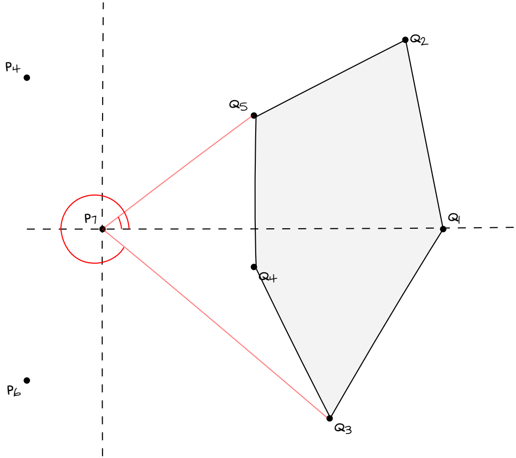
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Abordagem incremental



Definição do ponto extremo fora do polígono

Geometria computacional

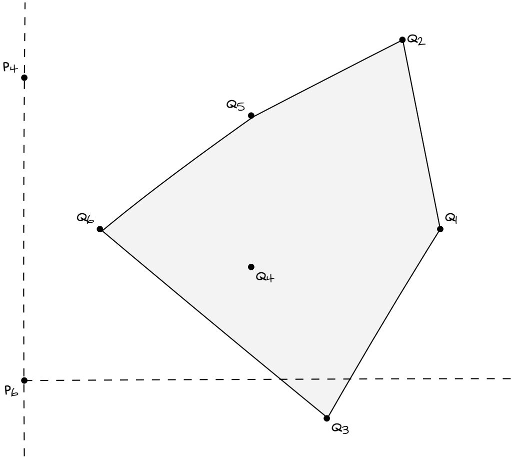
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Abordagem incremental



Seleção dos pontos com maior e menor grau

Geometria computacional

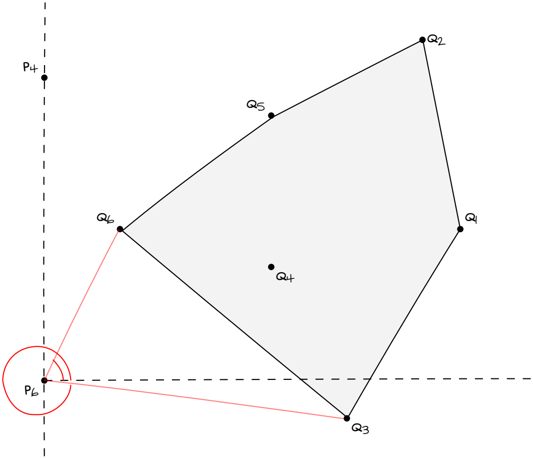
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Abordagem incremental



Definição do ponto extremo fora do polígono

Geometria computacional

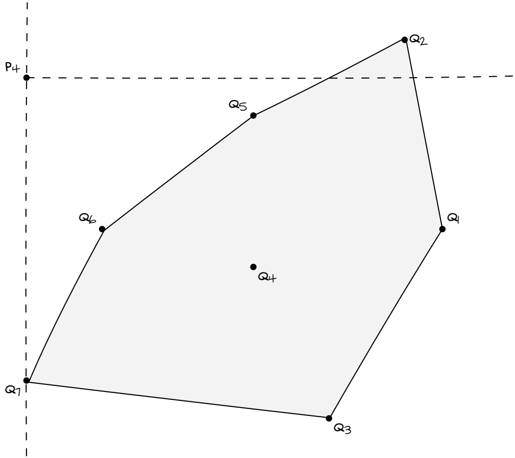
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Abordagem incremental



Seleção dos pontos com maior e menor grau

Geometria computacional

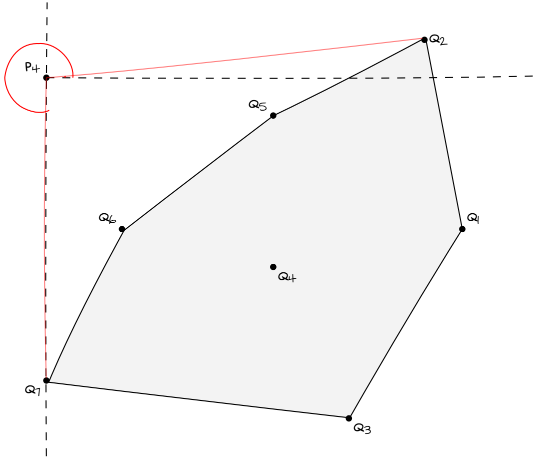
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Abordagem incremental



Definição do ponto extremo fora do polígono

Geometria computacional

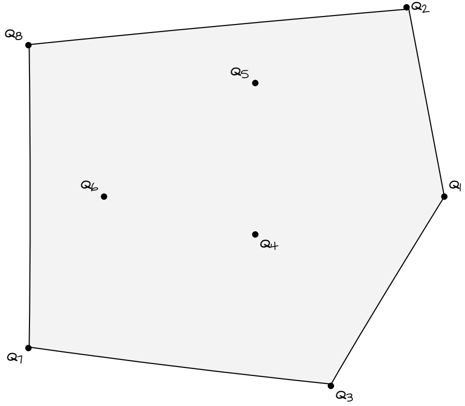
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Abordagem incremental



Seleção dos pontos com maior e menor grau

Geometria computacional

- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Abordagem incremental



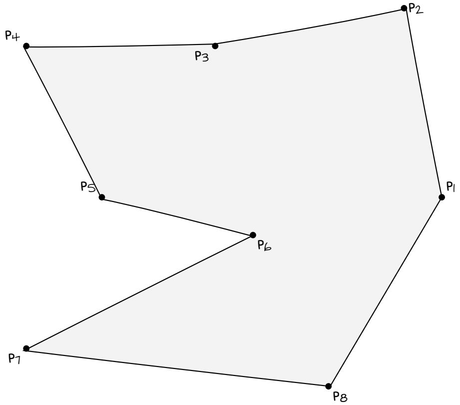
Espaço $O(n)$ e tempo $O(n^2)$

Geometria computacional

- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham

Geometria computacional

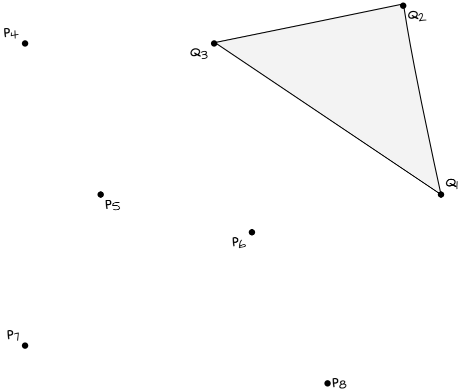
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham



Os pontos são ordenados para obter um polígono simples

Geometria computacional

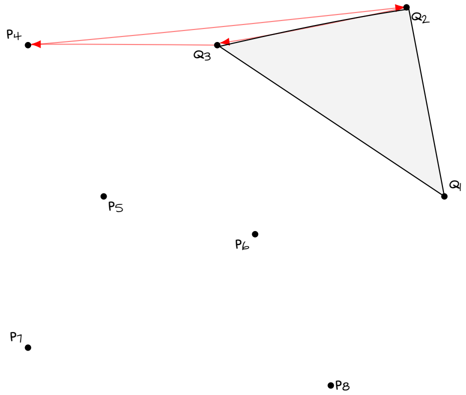
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham



Envoltória convexa com os 3 primeiros pontos

Geometria computacional

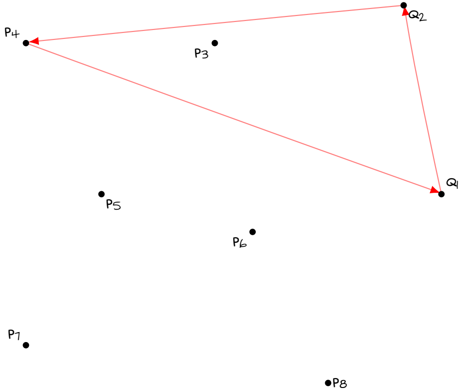
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham



Sentido horário de $q_2 - q_3 - p_4 - q_2$

Geometria computacional

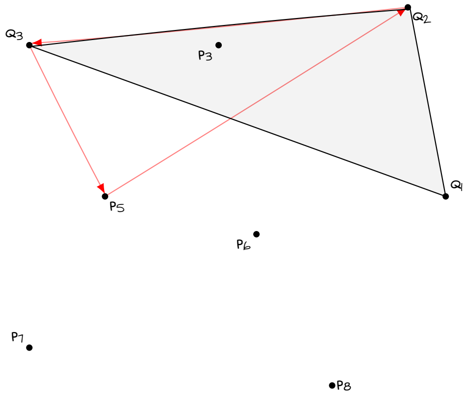
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham



Sentido anti-horário de $q_1 - q_2 - p_4 - q_1$

Geometria computacional

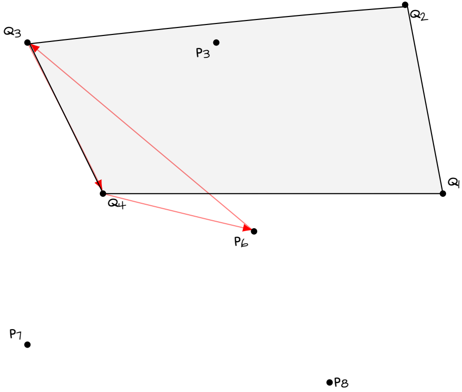
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham



Sentido anti-horário de $q_2 - q_3 - p_5 - q_2$

Geometria computacional

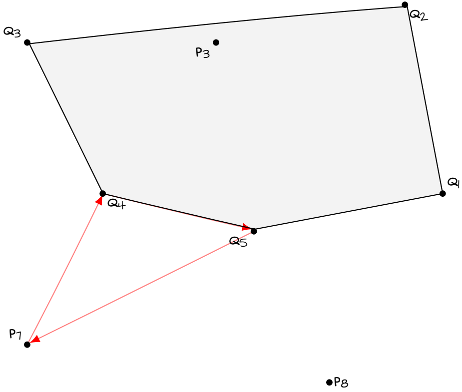
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham



Sentido anti-horário de $q_3 - q_4 - p_6 - q_3$

Geometria computacional

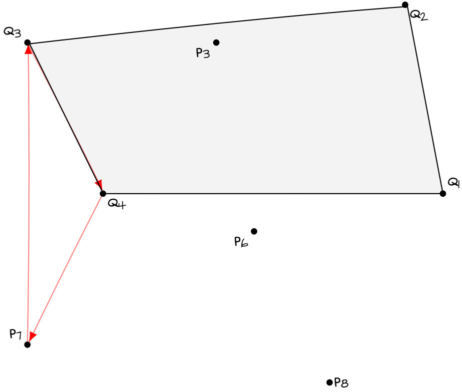
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham



Sentido horário de $q_4 - q_5 - p_7 - q_4$

Geometria computacional

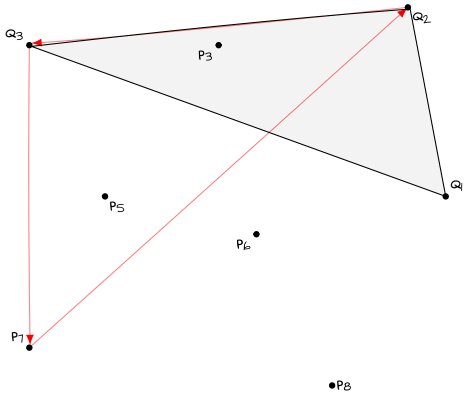
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham



Sentido horário de $q_3 - q_4 - p_7 - q_3$

Geometria computacional

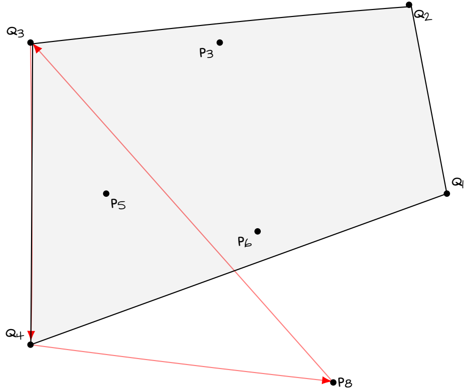
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham



Sentido anti-horário de $q_2 - q_3 - p_7 - q_2$

Geometria computacional

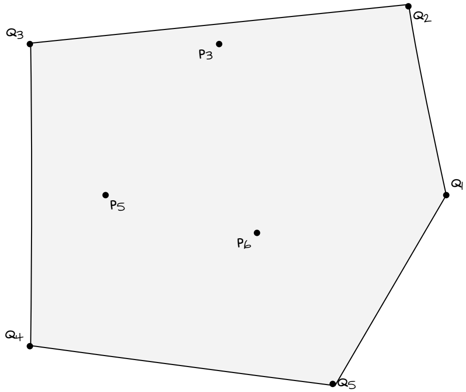
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham



Sentido anti-horário de $q_3 - q_4 - p_8 - q_3$

Geometria computacional

- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham



Geometria computacional

- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham

```
// Procedimento de varredura de Graham
void graham(pilha* P, poligono* Q, ponto* p[],
    uint32_t n) {
    // Criando um polígono simples
    poligono_simples(Q, p, n);
    // Inicializando a pilha
    iniciar(P, Q->p[0], Q->p[1], Q->p[2]);
    // Calculando o sentido dos pontos
    for(uint32_t i = 3; i < n; i++)
        while(!antihorario(topo(P), subtopo(P),
            Q->p[i])) desempilhar(P);
        empilhar(P, Q->P[i]);
}
```

Espaço $O(n)$ e tempo $O(n \log n + n) = O(n \log n)$

Geometria computacional

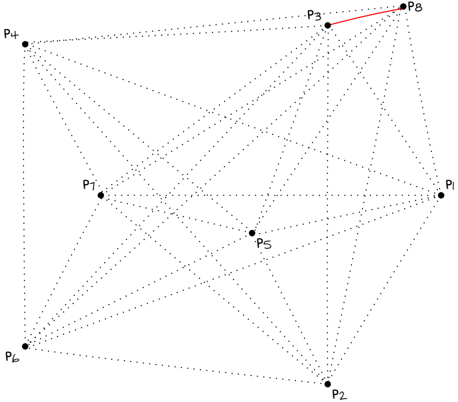
- Envoltória convexa: encontrar o menor polígono convexo Q que envolva um conjunto de n pontos
  - Varredura de Graham

```
// Procedimento de varredura de Graham
void graham(pilha* P, poligono* Q, ponto* p[],
    uint32_t n) {
    // Criando um polígono simples
    poligono_simples(Q, p, n);
    // Inicializando a pilha
    iniciar(P, Q->p[0], Q->p[1], Q->p[2]);
    // Calculando o sentido dos pontos
    for(uint32_t i = 3; i < n; i++)
        while(!antihorario(topo(P), subtopo(P),
            Q->p[i])) desempilhar(P);
        empilhar(P, Q->P[i]);
}
```

Aplicações: reconhecimento de padrões, segmentação de imagens, prevenção de colisões, ...

Geometria computacional

- Encontrar o par mais próximo em *n* pontos



Espaço $O(n)$ e tempo $O(n^2)$

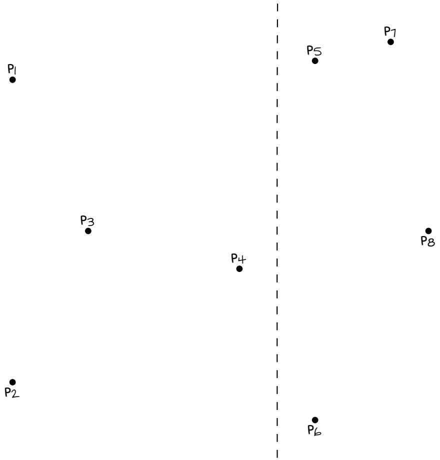
Geometria computacional

- Encontrar o par mais próximo em *n* pontos
  - Estratégia de divisão e conquista

Os pontos são ordenados pelos valores de *x* e *y*

Geometria computacional

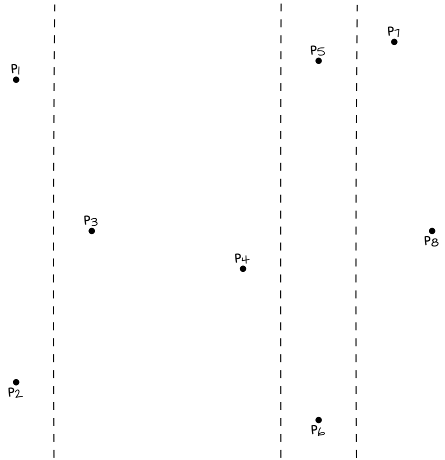
- Encontrar o par mais próximo em *n* pontos
  - Estratégia de divisão e conquista



Divisão do conjunto de pontos

Geometria computacional

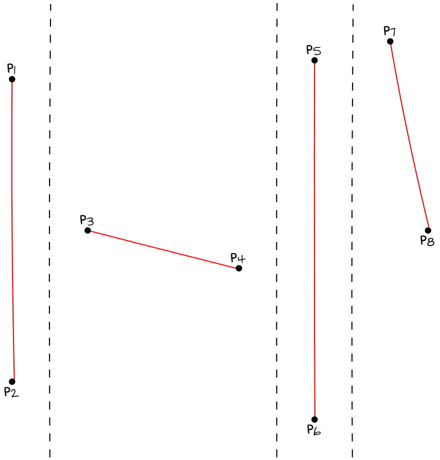
- Encontrar o par mais próximo em *n* pontos
  - Estratégia de divisão e conquista



Divisão do conjunto de pontos

Geometria computacional

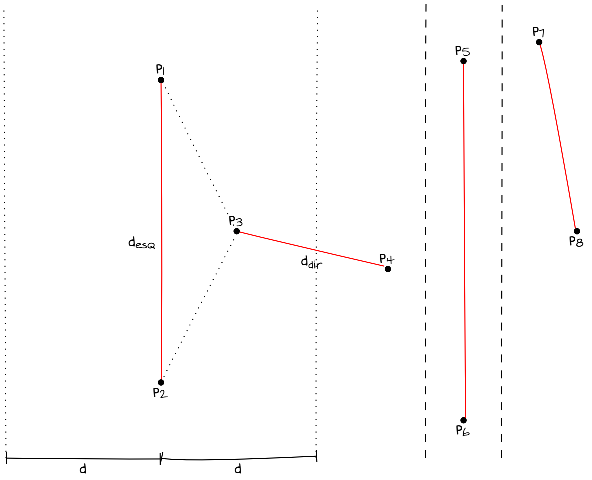
- Encontrar o par mais próximo em *n* pontos
  - Estratégia de divisão e conquista



Calcular a menor distância de cada partição

Geometria computacional

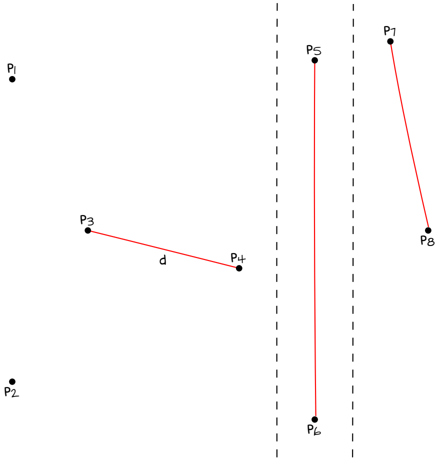
- Encontrar o par mais próximo em *n* pontos
  - Estratégia de divisão e conquista



Conquista das soluções parciais com $d = min(d_{esq}, d_{dir})$

Geometria computacional

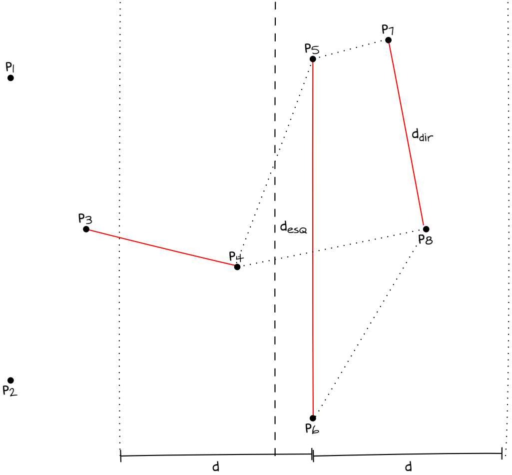
- Encontrar o par mais próximo em *n* pontos
  - Estratégia de divisão e conquista



É obtida uma solução parcial *d*

Geometria computacional

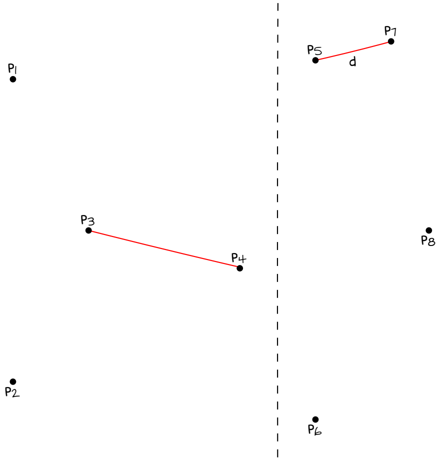
- Encontrar o par mais próximo em *n* pontos
  - Estratégia de divisão e conquista



Conquista das soluções parciais com $d = min(d_{esq}, d_{dir})$

Geometria computacional

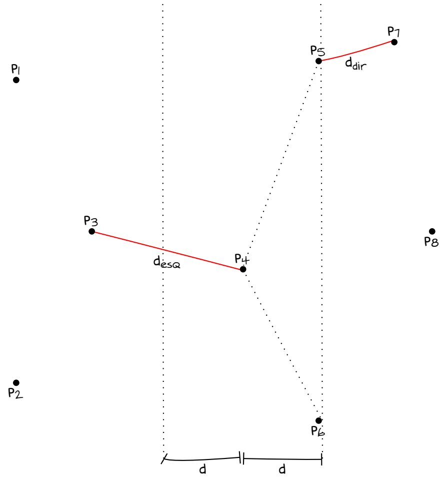
- Encontrar o par mais próximo em *n* pontos
  - Estratégia de divisão e conquista



É obtida uma solução parcial *d*

Geometria computacional

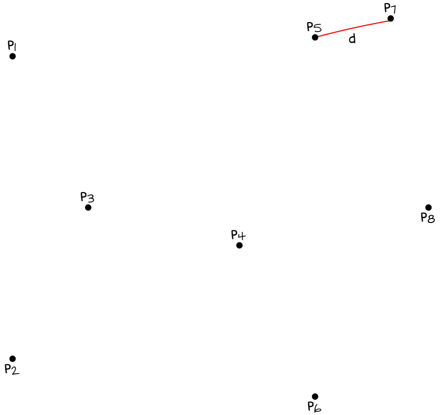
- Encontrar o par mais próximo em *n* pontos
  - Estratégia de divisão e conquista



Conquista das soluções parciais com $d = min(d_{esq}, d_{dir})$

Geometria computacional

- Encontrar o par mais próximo em *n* pontos
  - Estratégia de divisão e conquista



Par de pontos com menor distância *d*

- Encontrar o par mais próximo em n pontos
  - Estratégia de divisão e conquista

```
// Função de distância do par mais próximo
float par_mais_proximo(ponto* P[], uint32_t n) {
    // Caso com até 3 pontos
    if(n <= 3) return forca_bruta(P, n);
    else {
        // Mínima distância das partições
        float esq = par_mais_proximo(P, n / 2);
        float dir = par_mais_proximo(P + (n / 2), n - (n / 2));
        float d = min(esq, dir);
        // Busca nas partições até distância d
        return min(d, busca_particoes(P, d));
    }
}
```

Espaço $O(n)$ e tempo $O(n \log n)$

Geometria computacional

- Encontrar o par mais próximo em *n* pontos
  - Estratégia de divisão e conquista

```
// Função de distância do par mais próximo
float par_mais_proximo(ponto* P[], uint32_t n) {
    // Caso com até 3 pontos
    if(n <= 3) return forca_bruta(P, n);
    else {
        // Mínima distância das partições
        float esq = par_mais_proximo(P, n / 2);
        float dir = par_mais_proximo(P + (n / 2), n -
            (n / 2));
        float d = min(esq, dir);
        // Busca nas partições até distância d
        return min(d, busca_particoes(P, d));
    }
}
```

Aplicações: astronomia,
sistemas de irrigação, heurísticas de otimização, ...