Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Geometria Computacional

André Tavares da Silva

andre.silva@udesc.br

baseado no material de João Comba

Roteiro

- Fundamentação
- Interseções de Segmentos de Linha
- Envoltória Convexa
- Particionamento de Polígonos
- Triangulações de Delaunay
- Diagramas de Voronoy
- Nível de Detalhe
- Grafos de Visibilidade
- Planejamento de Movimento
- BSP-trees p/ visibilidade

Geometria Computacional

- Início na década de 70
- Problemas geométricos
- **Meta:** Buscar algoritmos e estruturas de dados com bom desempenho quando os problemas escalam de tamanho
- **Aplicações:** Computação Gráfica, Processamento de Imagens, Robótica, Sistemas Geográficos de Informação, VLSI, CAD, etc.

Geometria Computacional

- Problemas Típicos:
  - **Entrada:** Objetos Geométricos (pontos, linhas, planos, etc)
  - **Saída:** Resultados de consulta sobre os objetos, ou um novo objeto geométrico
- **Ingredientes:** Geometria, Topologia, Algoritmos, Estruturas de dados, Análise de algoritmos, Problemas numéricos

Aplicações

Computação Gráfica:

- Interseções entre primitivas
- Primitiva apontada pelo mouse
- Subconjunto de primitivas dentro de uma região
- Remoção de superfícies ocultas em 3D
- Cálculos de sombras em ray-tracing e radiosidade
- Detectar colisões entre objetos

Aplicações

Robótica:

- Planejamento de movimento de robô
- Planejamento de articulações de robô
  - Alcance do braço
  - Orientações das diferentes juntas

Aplicações

Sistemas Geográficos de Informação:

- Representações de dados geográficos:
  - Topográfica, Hidrográfica, Separações físicas e políticas, estradas (mapas diversos)
  - Dados são muito numerosos
- Extrair informações eficientemente:
  - Qual a média de chuva em uma dada região

Aplicações

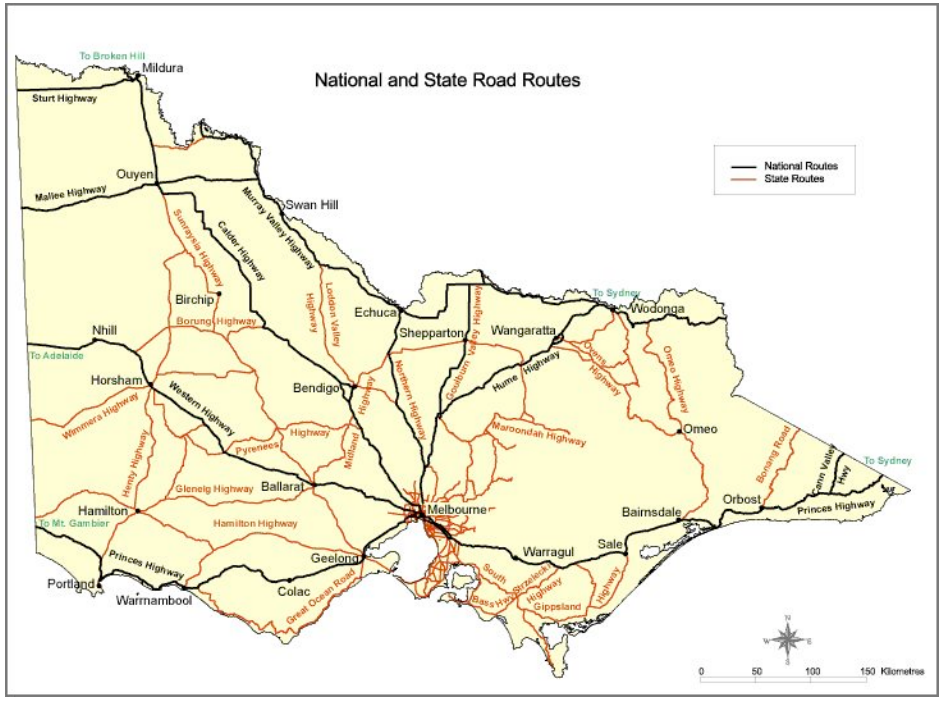
CAD/CAM:

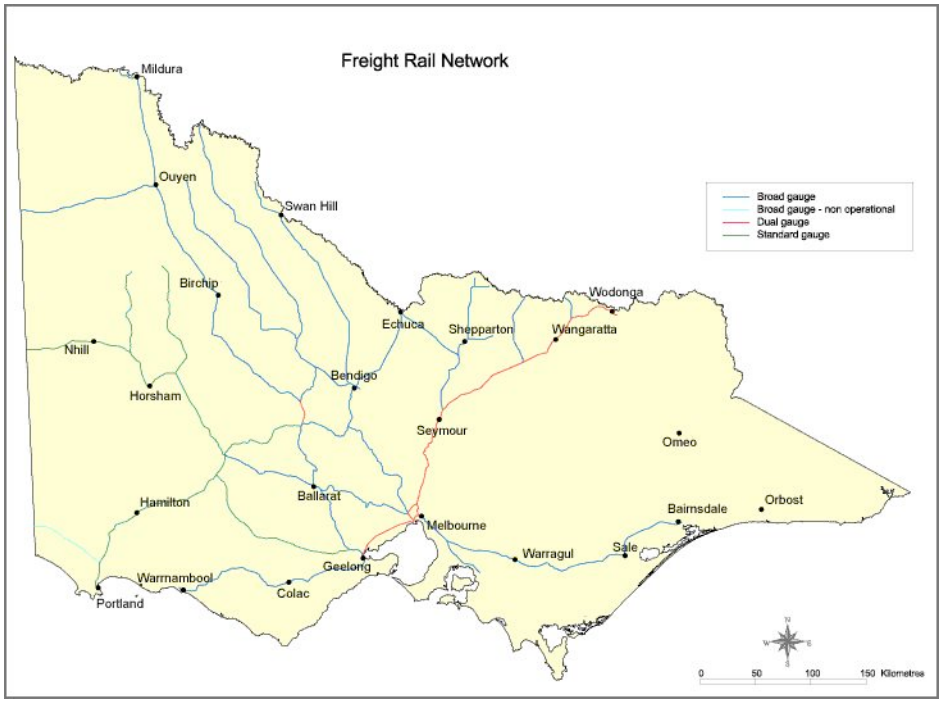
- Placas de Circuitos, Projetos arquitetônicos
- Entidades geométricas (problemas geométricos aparecem)
- Interseções e uniões de objetos
- Decomposição de objetos em formas mais simples
- Visualização dos objetos projetados

Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Sobreposição de Mapas Temáticos



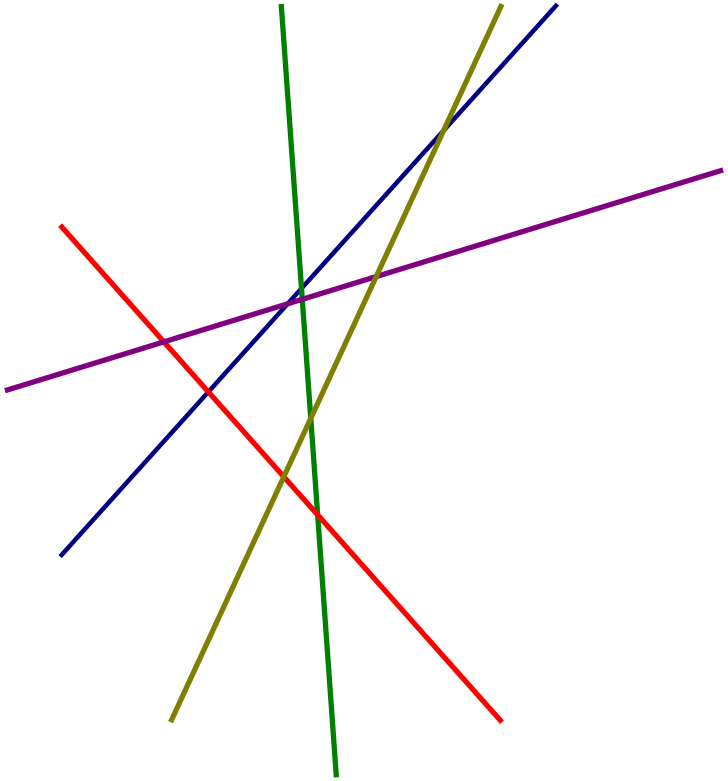


Sobreposição de Mapas Temáticos

- Segmentos curvos aproximados por vários segmentos de linha
- Problema de Sobreposição (Overlay):
  - Dados dois conjuntos de segmentos de linha, calcular todas interseções entre o segmento de um conjunto e um segmento do outro conjunto
- $O(n^2)$?
- Colocar todos segmentos em um mesmo conjunto (simplificar a descrição da solução)

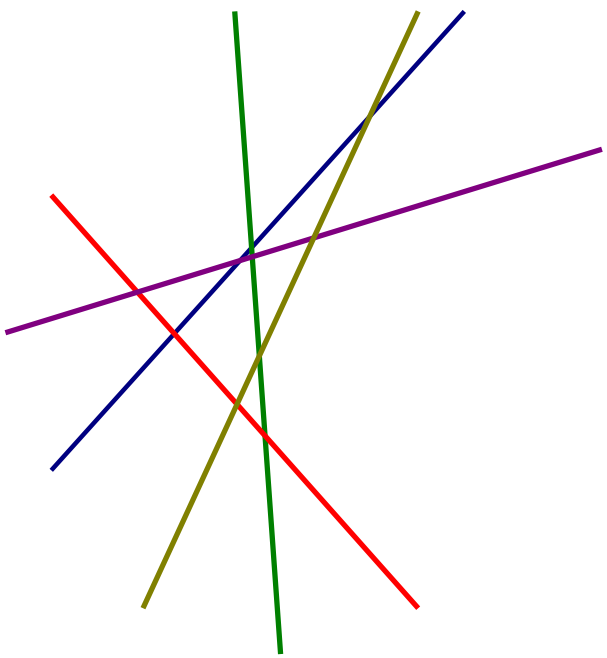
Sobreposição de Mapas Temáticos

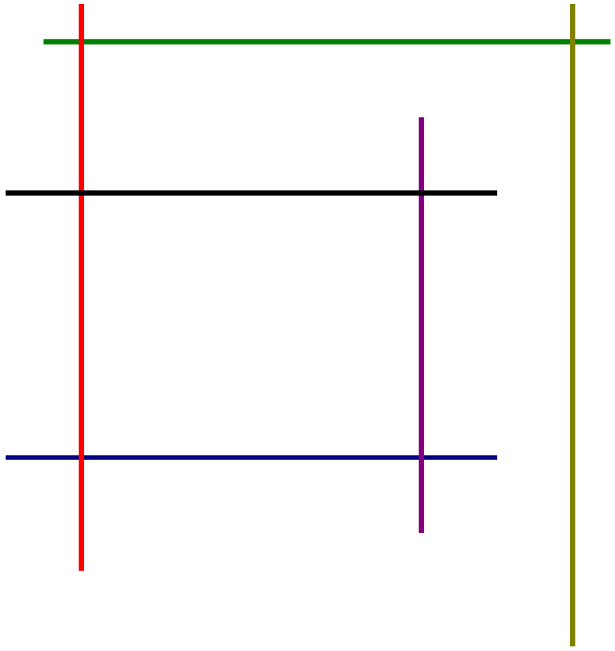
Casos Extremos



Sobreposição de Mapas Temáticos

Casos Extremos





Sobreposição de Mapas Temáticos

Como evitar testar todos os pares?

- Segmentos próximos são candidatos mais fortes a interseção
- Algoritmo de *Plane Sweep*
- Ordernar segmentos na *sweep line*

Estruturas de Dados

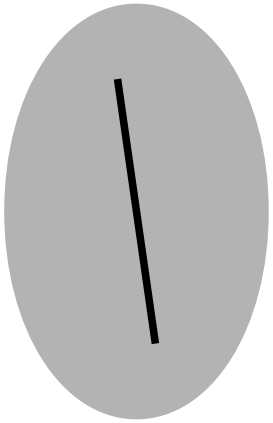
- Fila de eventos
- Representação por grafos planares
- Lista de arestas
- Lista de arestas duplamente conectadas

Departamento de Ciência da COMPUTAÇÃO

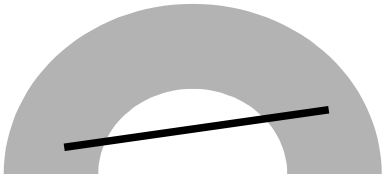
UDESC Joinville

Convex Hull

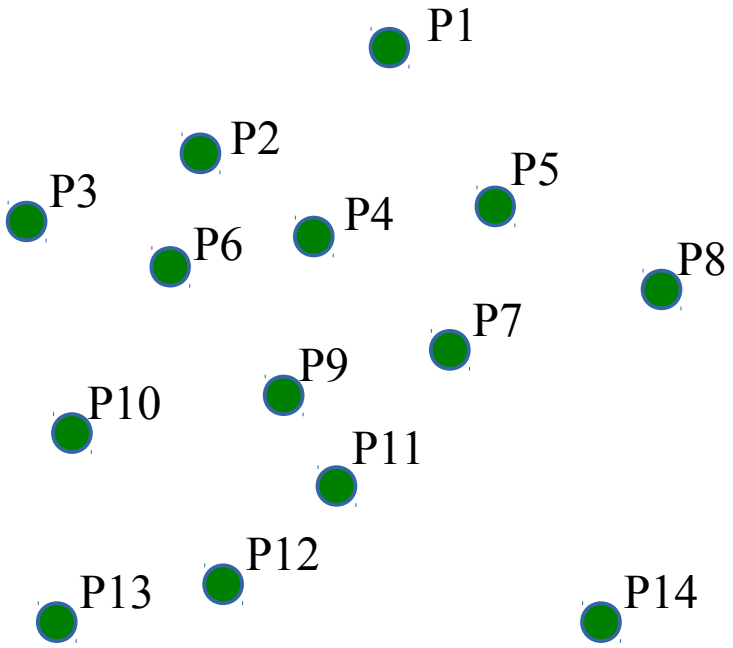
(envoltória convexa / fecho convexo)



Convexo



Não convexo

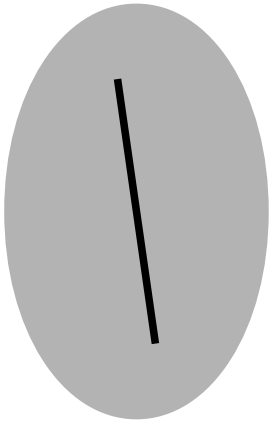


Departamento de Ciência da COMPUTAÇÃO

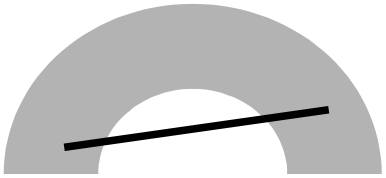
UDESC Joinville

Convex Hull

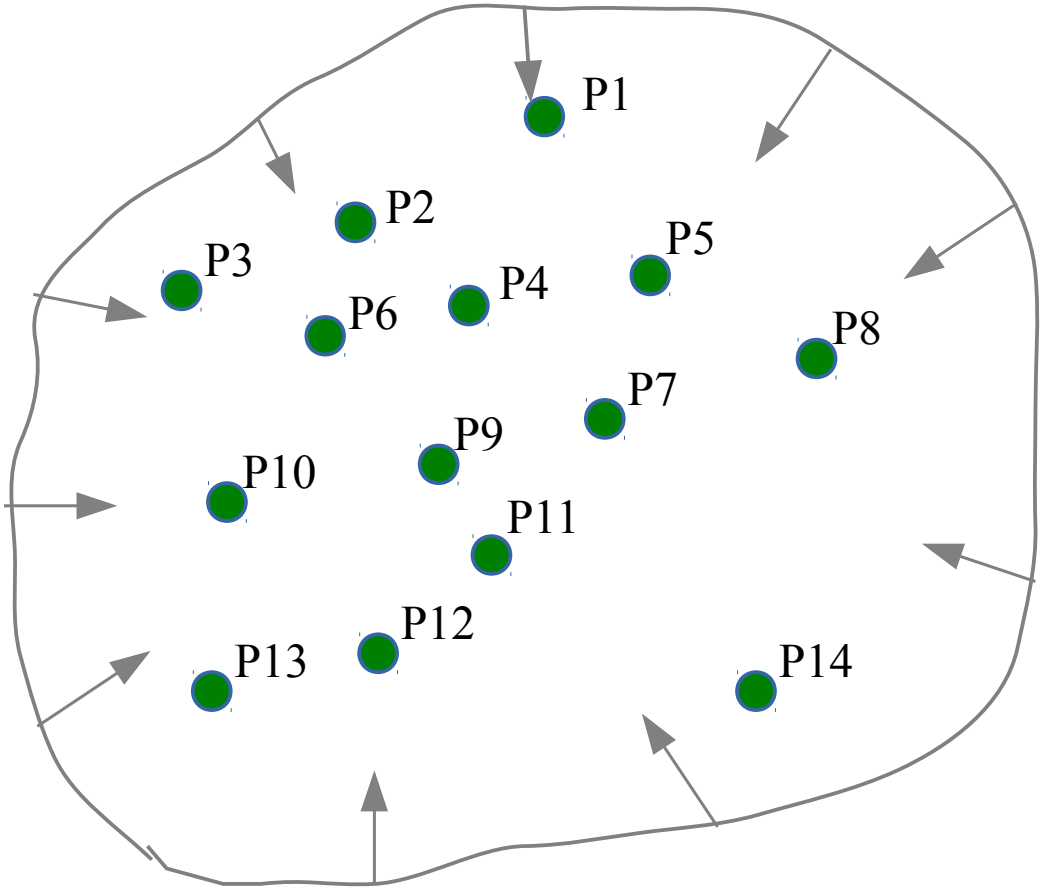
(envoltória convexa / fecho convexo)



Convexo



Não convexo

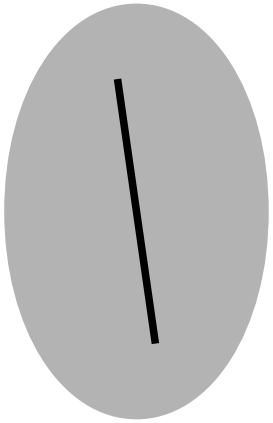


Departamento de Ciência da COMPUTAÇÃO

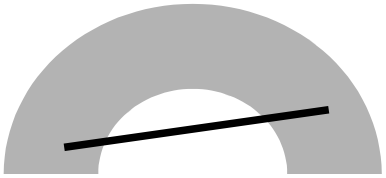
UDESC Joinville

Convex Hull

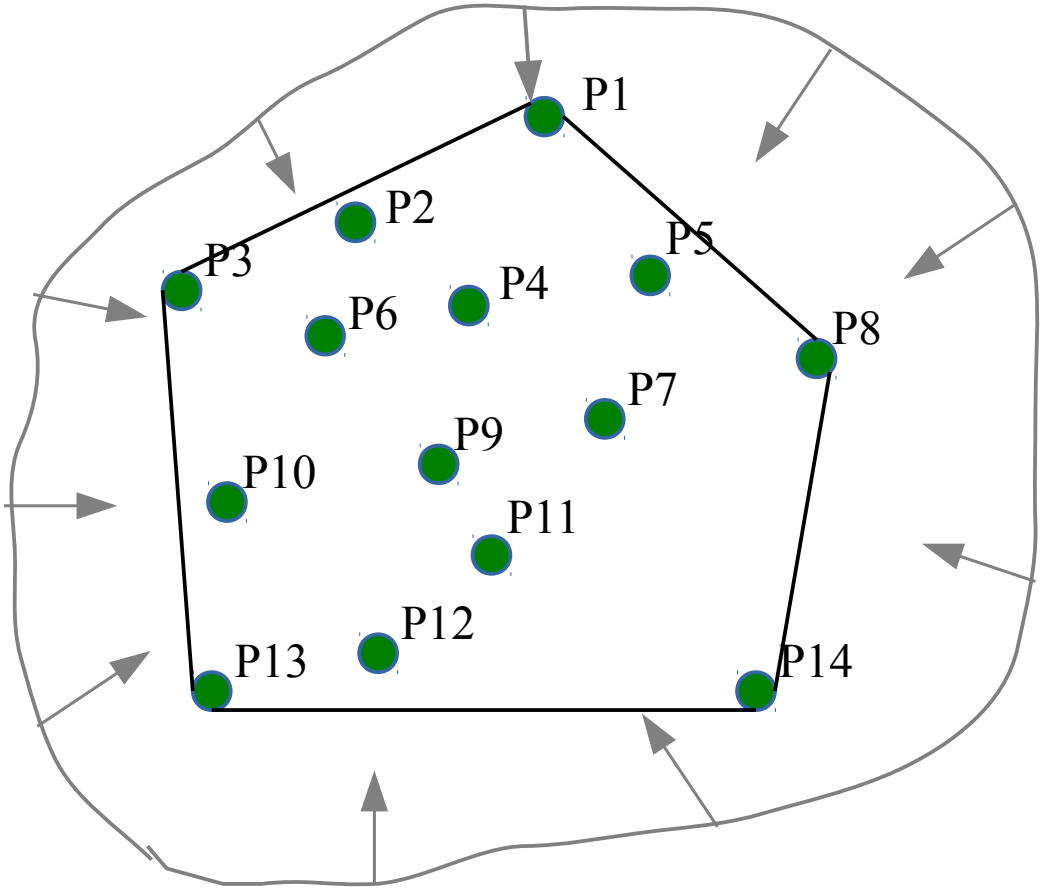
(envoltória convexa / fecho convexo)



Convexo



Não convexo



Departamento de Ciência da COMPUTAÇÃO

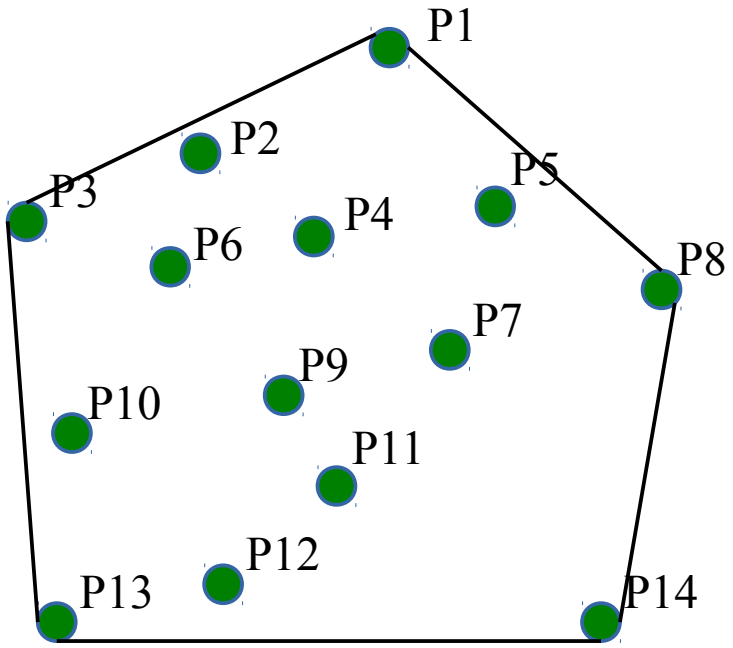
UDESC Joinville

Convex Hull

(envoltória convexa / fecho convexo)

Entrada: P1, P2, P3,...,P14

Saída: P3, P1,P8,P14,P13



Convex Hull

(envoltória convexa / fecho convexo)

Como saber se um ponto está a direita de um segmento de reta?

- Ao ter uma linha formada por P0(x0,y0) até P1(x1,y1), um ponto P(x,y) e a expressão

  `(y-y0)(x1-x0)-(x-x0)(y1-y0)`,

  podemos dizer que se o valor da expressão for menor que 0 então P está à direita do segmento de linha, se maior que 0 à esquerda e se **igual** a 0 então o ponto está **sobre o segmento** de linha.

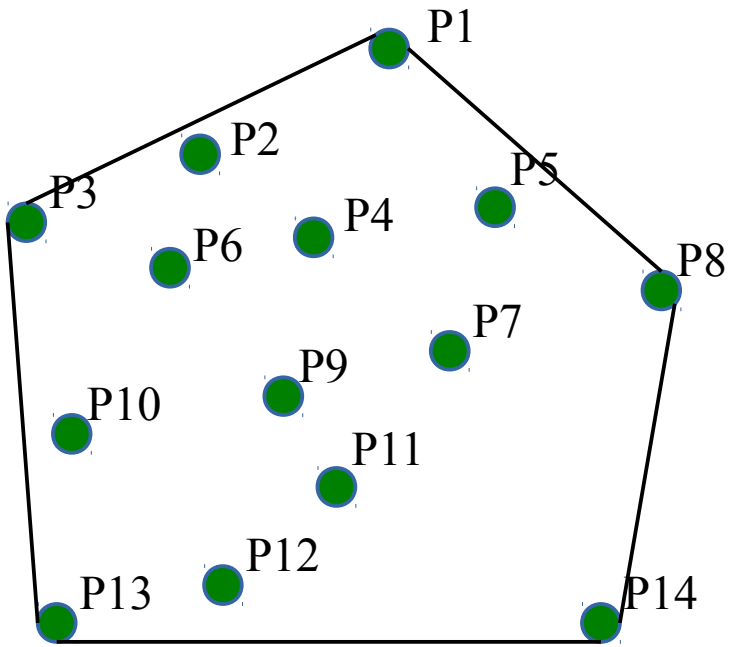
Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Convex Hull

(envoltória convexa / fecho convexo)

Entrada: Conjunto de pontos P no plano
Saída: Uma lista de vértices em ordem
1. E = ø
2. FOR todos pares (p,q) PxP (p!=q)
3. DO valido = true
4. FOR todos pontos r P(r!=p and r!=q)
5. IF r está a direita da reta pq
6. THEN valido = false
7. IF valido THEN adicione pq para E
8. Para o conjunto E de arestas, construa uma lista de vértices classificados em ordem cronológica



Departamento de Ciência da COMPUTAÇÃO

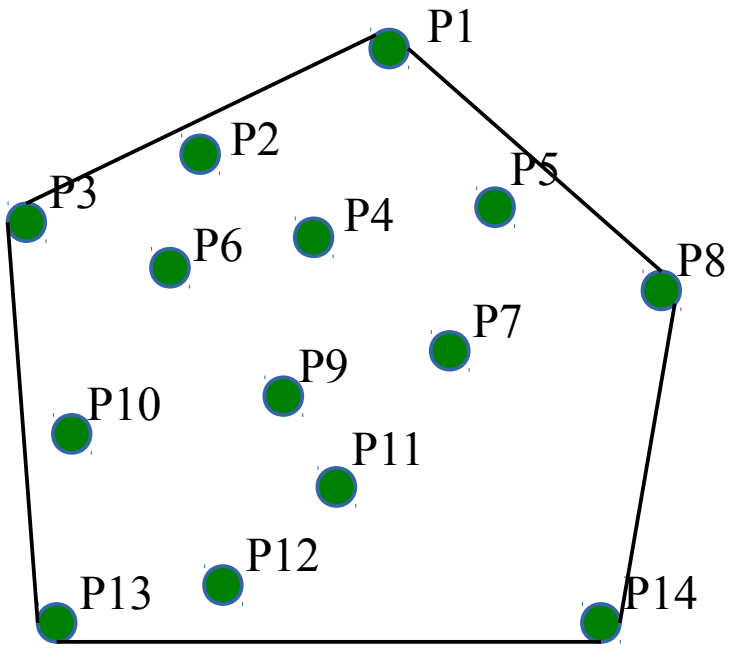
UDESC Joinville

Convex Hull

(envoltória convexa / fecho convexo)

Entrada: Conjunto de pontos P no plano
Saída: Uma lista de vértices em ordem

1. E = ø
2. FOR todos pares (p,q) PxP (p!=q)
3. DO valido = true
4. FOR todos pontos r P(r!=p and r!=q)
5. IF r está a direita da reta pq
6. THEN valido = false
7. IF valido THEN adicione pq para E
8. Para o conjunto E de arestas, construa uma lista de vértices classificados em ordem cronológica



Complexidade: $(n^2 - n) * (n-2) = n^3 - 3n^2 + 2n = \mathbf{O(n^3)}$

Departamento de Ciência da COMPUTAÇÃO

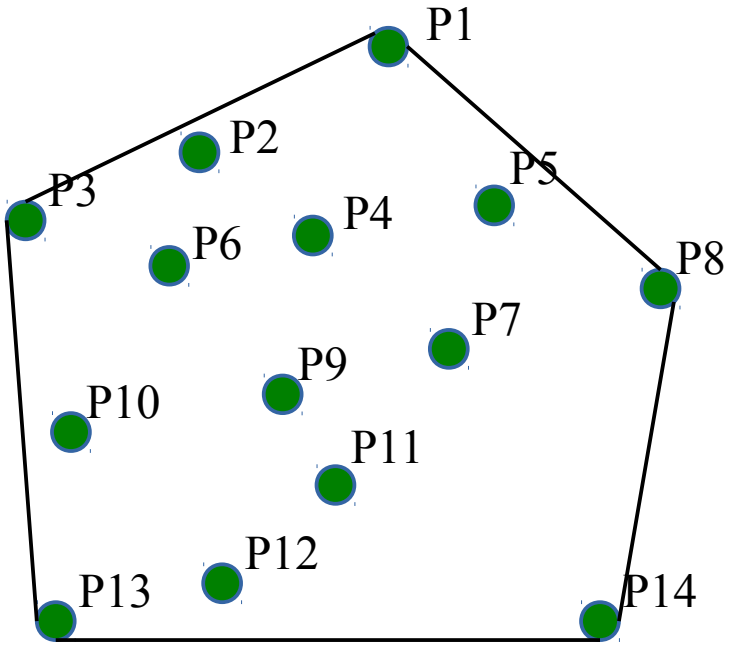
UDESC Joinville

Convex Hull

(envoltória convexa / fecho convexo)

E se os pontos estiverem ordenados?

E como ordenar?

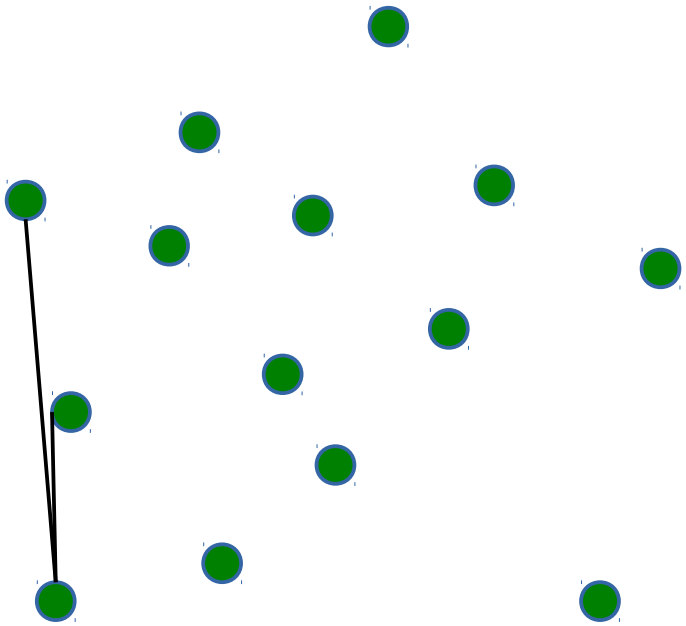


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Convex Hull

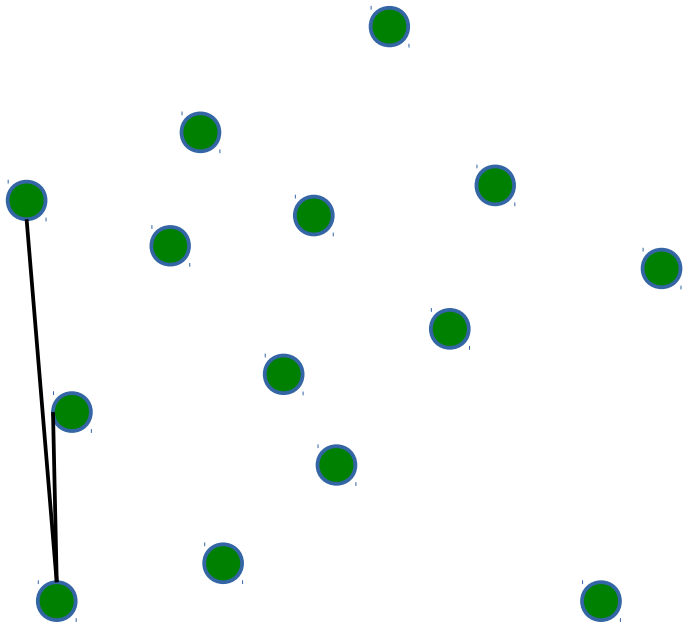
(envoltória convexa / fecho convexo)



Convex Hull

(envoltória convexa / fecho convexo)

Virada à esquerda!

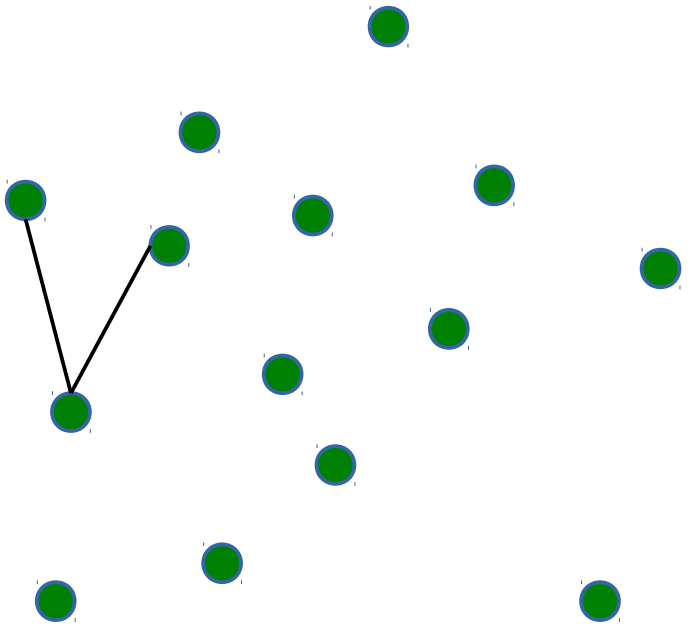


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Convex Hull

(envoltória convexa / fecho convexo)

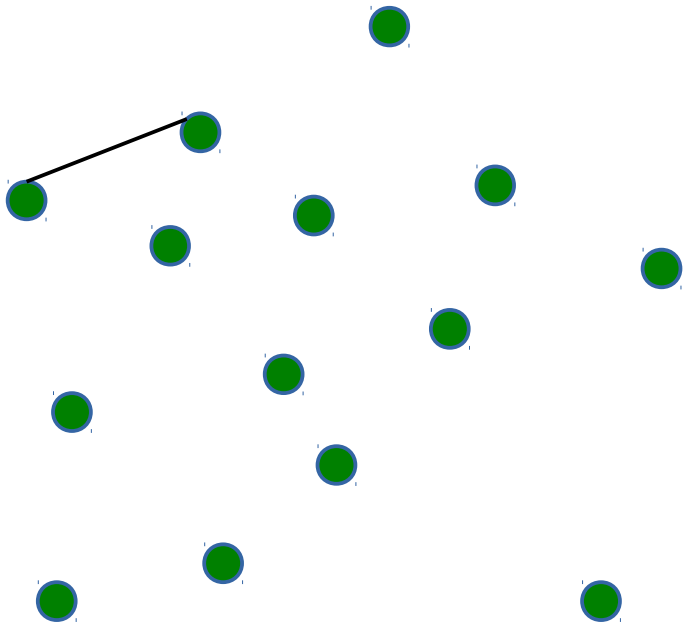


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

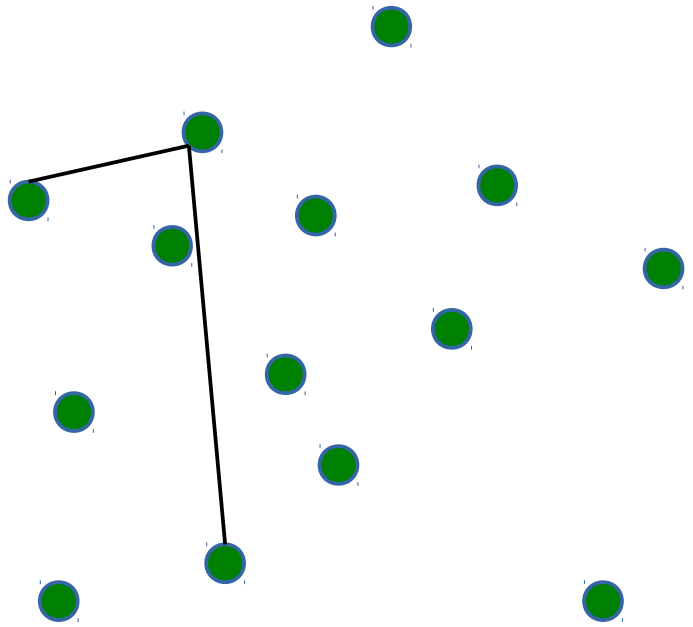
Convex Hull

(envoltória convexa / fecho convexo)



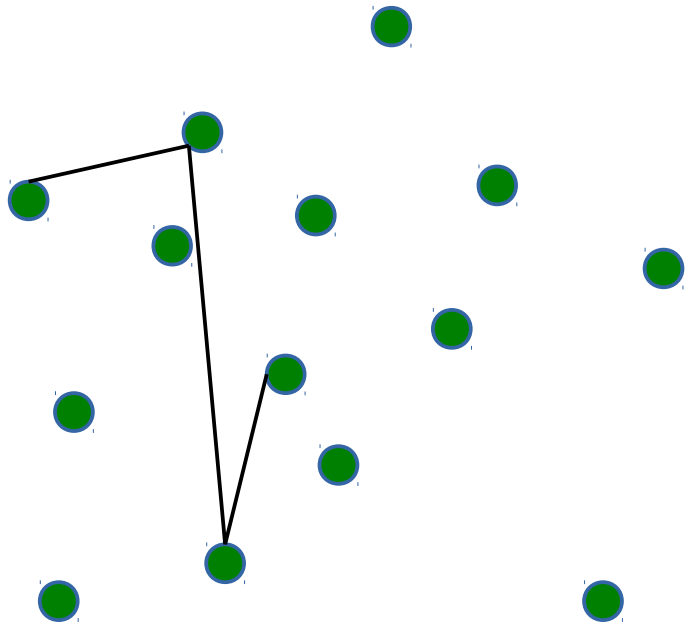
Convex Hull

(envoltória convexa / fecho convexo)



Convex Hull

(envoltória convexa / fecho convexo)

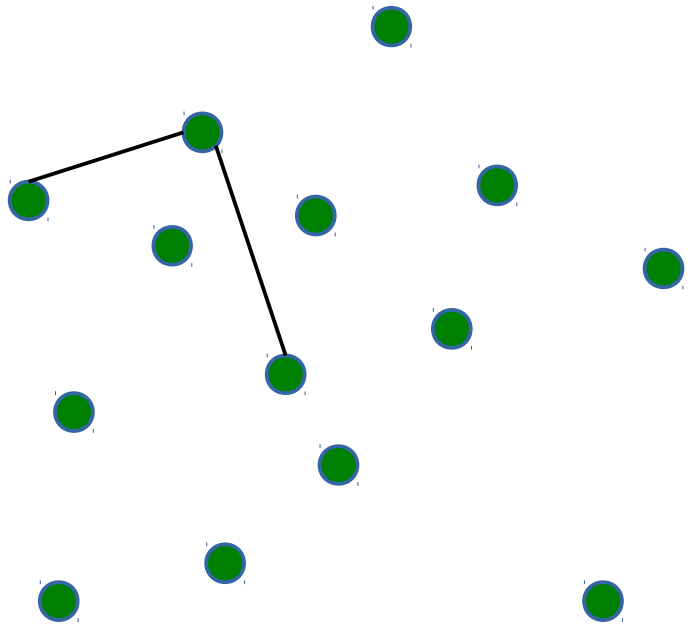


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

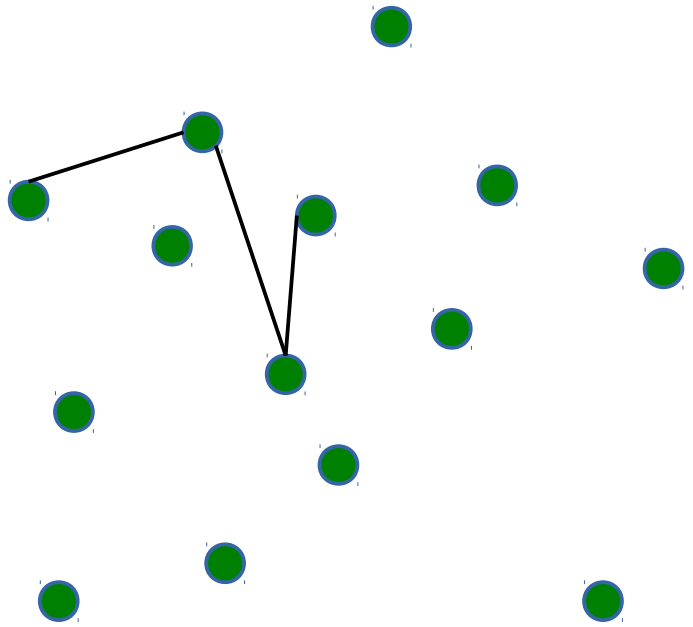
Convex Hull

(envoltória convexa / fecho convexo)



Convex Hull

(envoltória convexa / fecho convexo)

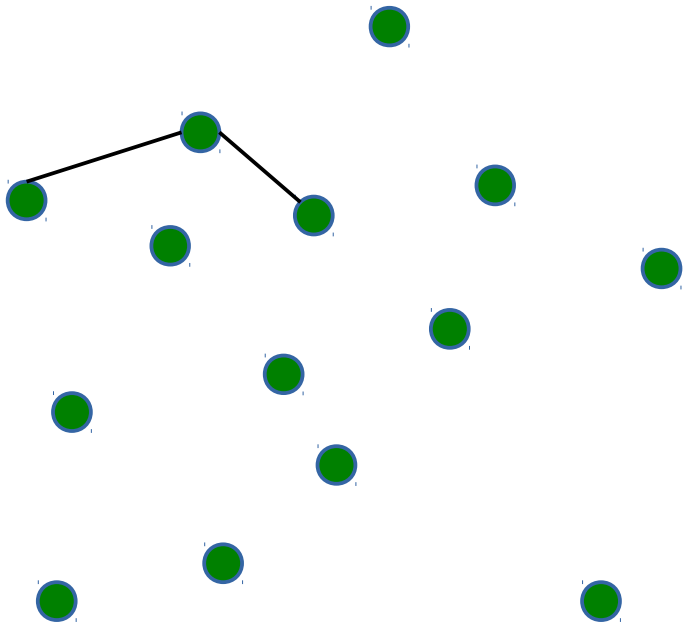


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

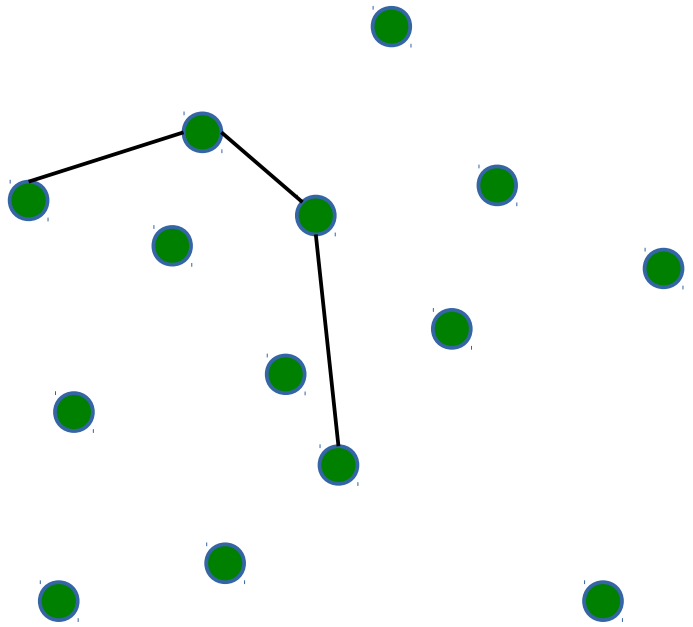
Convex Hull

(envoltória convexa / fecho convexo)



Convex Hull

(envoltória convexa / fecho convexo)

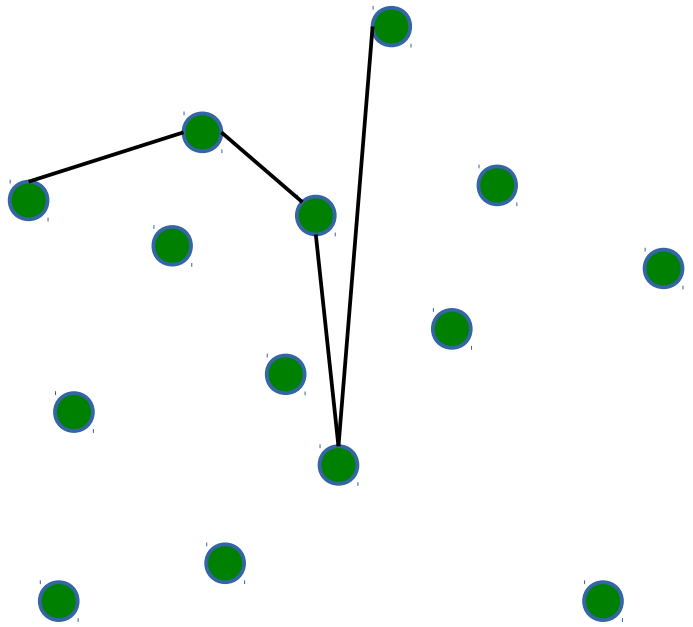


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Convex Hull

(envoltória convexa / fecho convexo)

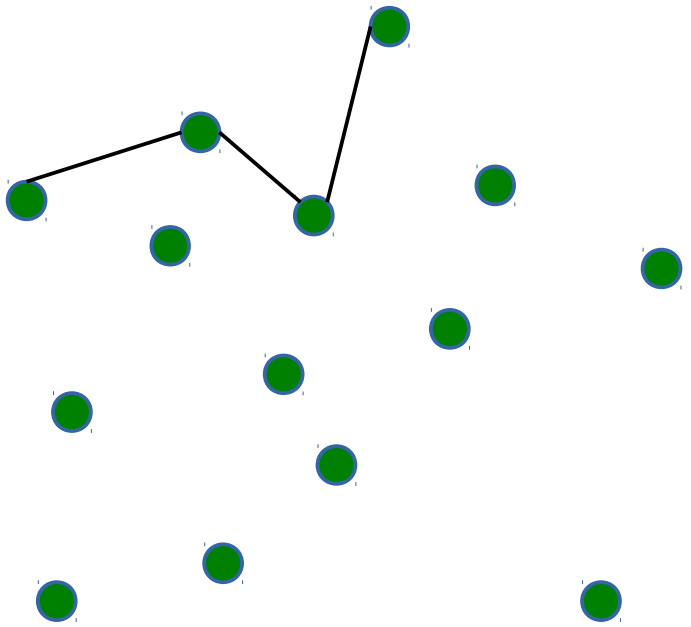


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Convex Hull

(envoltória convexa / fecho convexo)

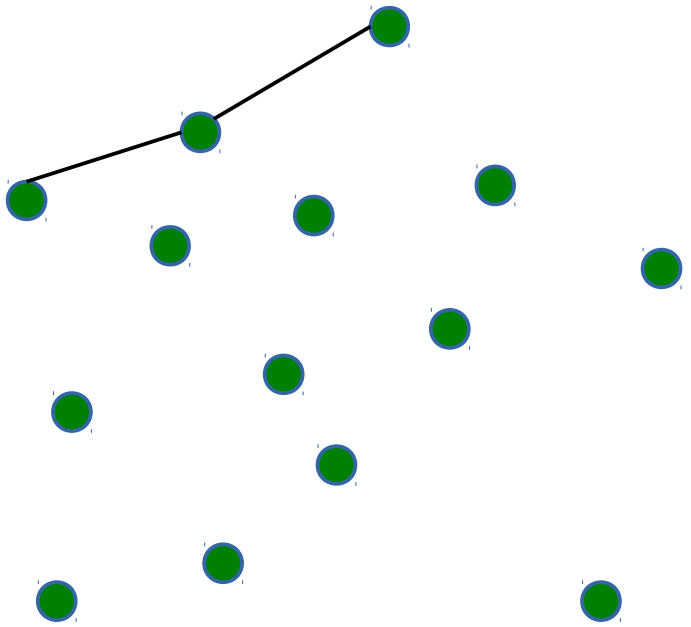


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Convex Hull

(envoltória convexa / fecho convexo)

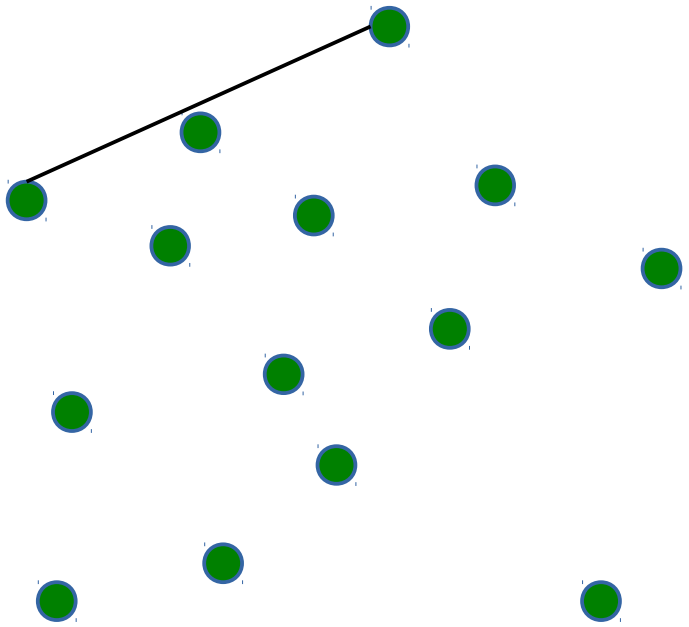


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Convex Hull

(envoltória convexa / fecho convexo)

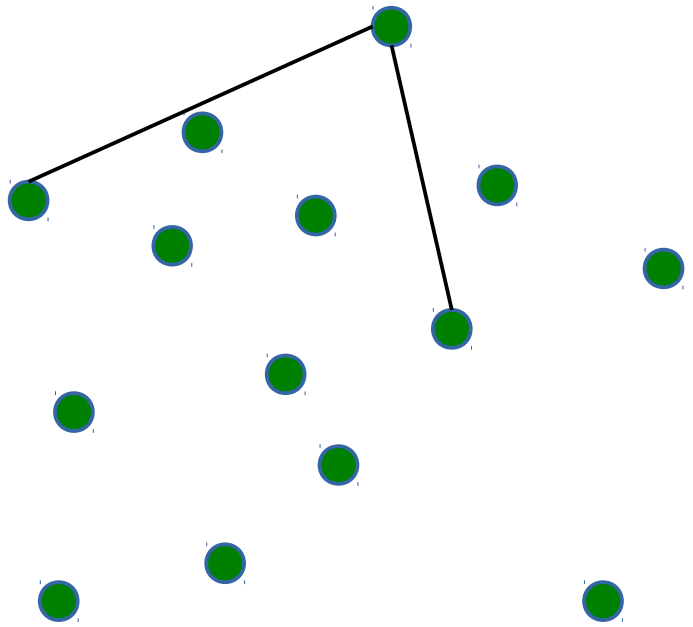


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

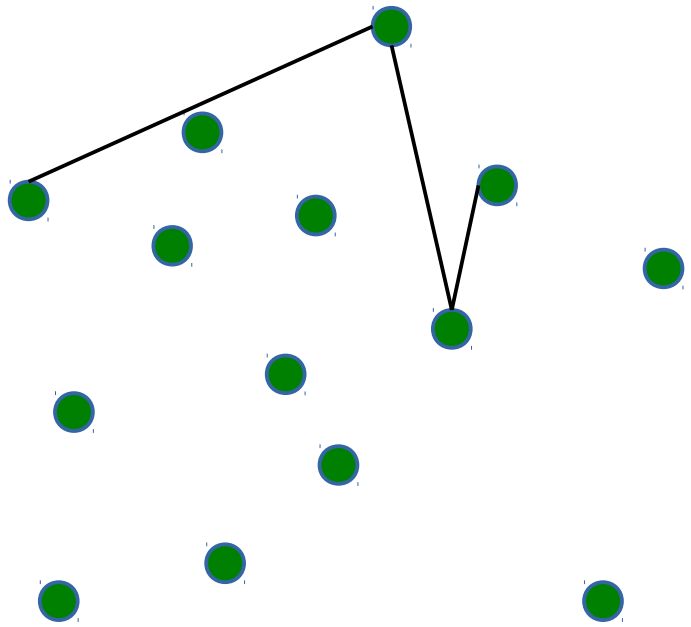
Convex Hull

(envoltória convexa / fecho convexo)



Convex Hull

(envoltória convexa / fecho convexo)

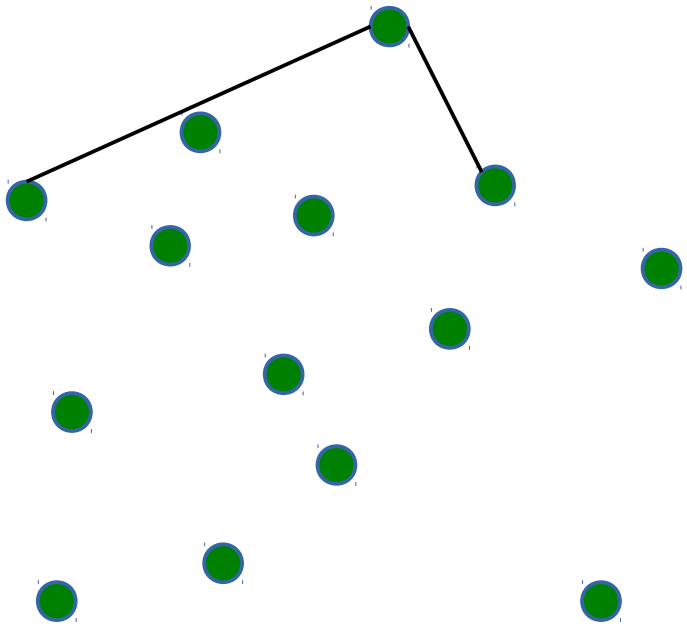


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Convex Hull

(envoltória convexa / fecho convexo)

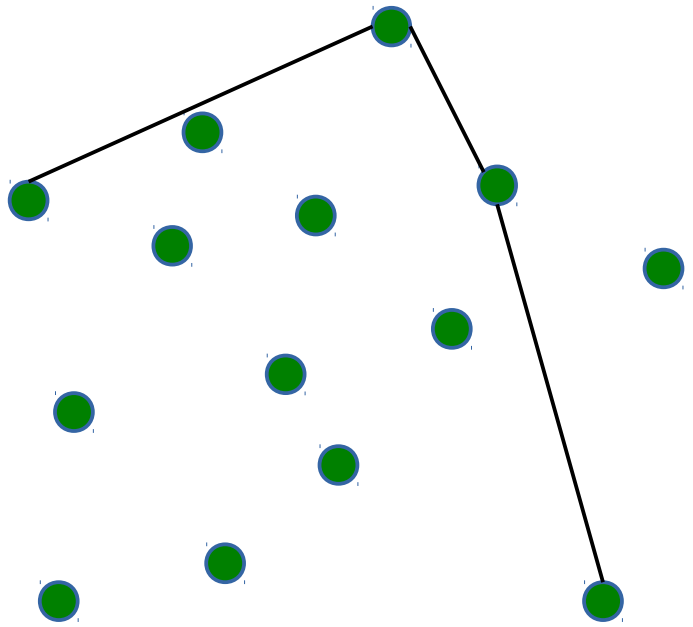


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Convex Hull

(envoltória convexa / fecho convexo)

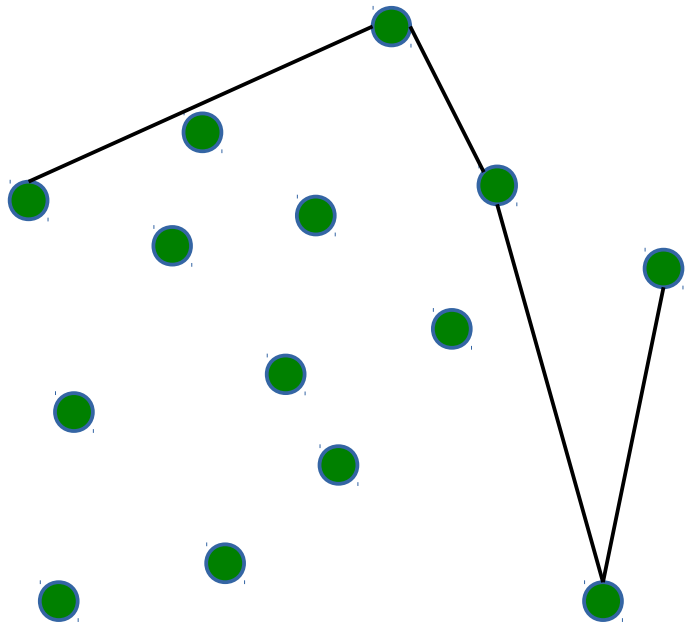


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Convex Hull

(envoltória convexa / fecho convexo)

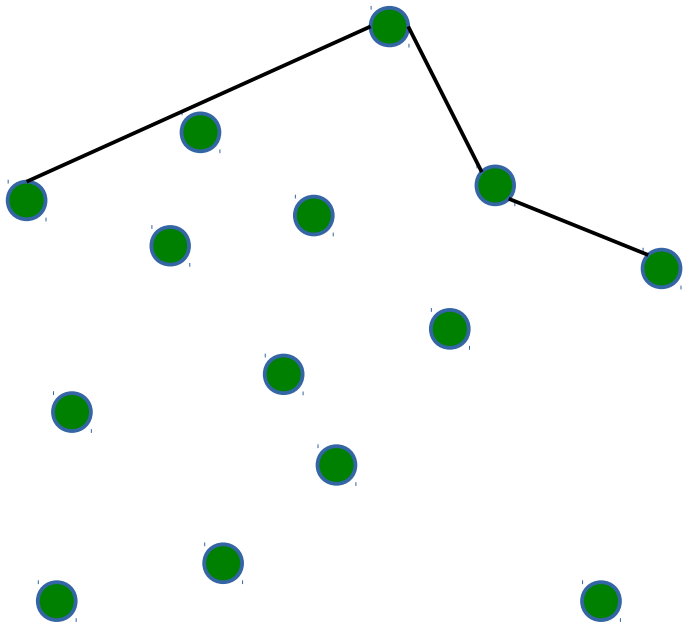


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

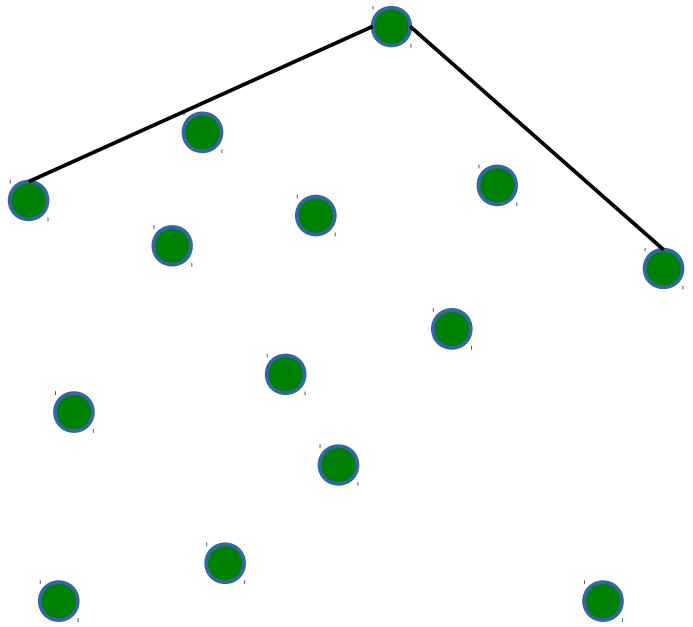
Convex Hull

(envoltória convexa / fecho convexo)



Convex Hull

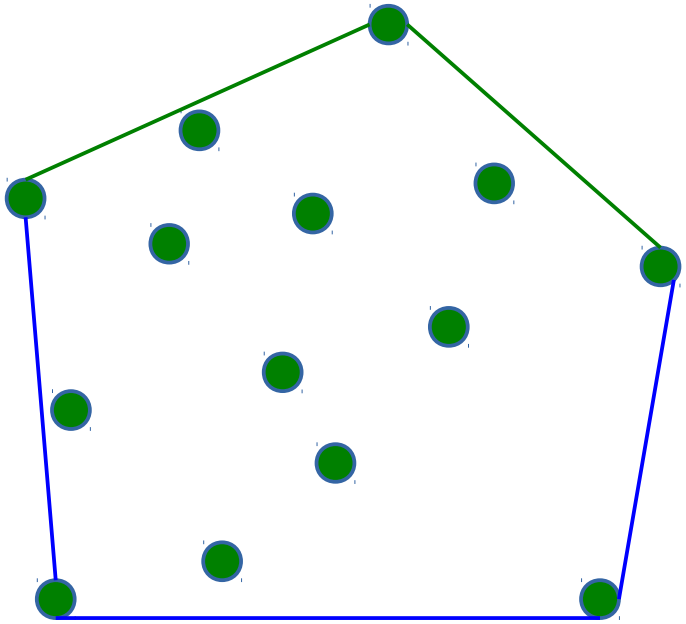
(envoltória convexa / fecho convexo)



Convex Hull

(envoltória convexa / fecho convexo)

Idem para a parte inferior!

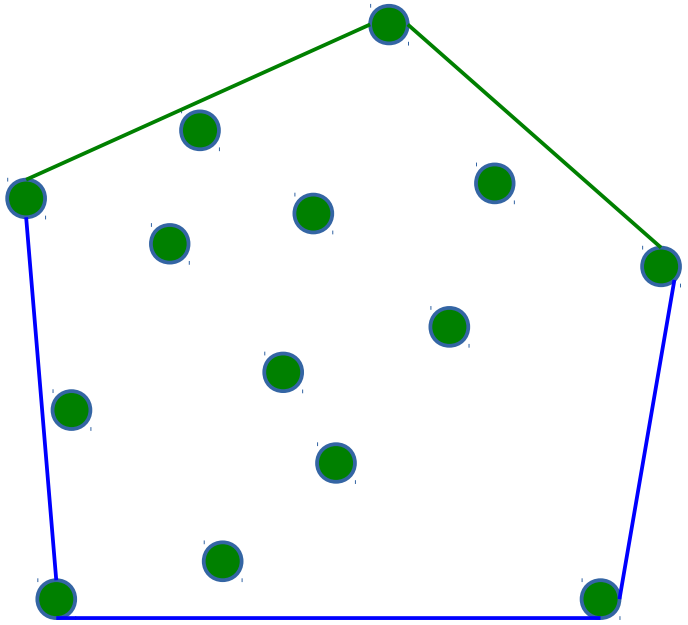


Complexidade: n log(n) + n log(n) = **n log(n)**

Convex Hull

(envoltória convexa / fecho convexo)

Ex.: computador com capacidade de realizar 1.000.000 de comparações se *P* está esq/dir de segmento de reta em um segundo.

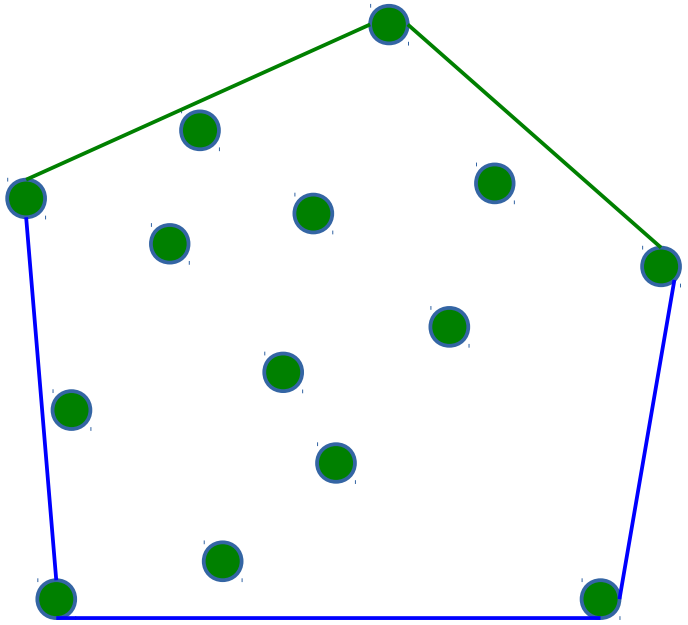


Se tivermos uma nuvem com 100.000 de pontos, quanto tempo precisaríamos para Calcular o *convex hull*?

Convex Hull

(envoltória convexa / fecho convexo)

Ex.: computador com capacidade de realizar 1.000.000 de comparações se *P* está esq/dir de segmento de reta em um segundo.



Se tivermos uma nuvem com 100.000 de pontos, quanto tempo precisaríamos para Calcular o *convex hull*?

Resposta: 0,5 segundos no n log(n) e ...

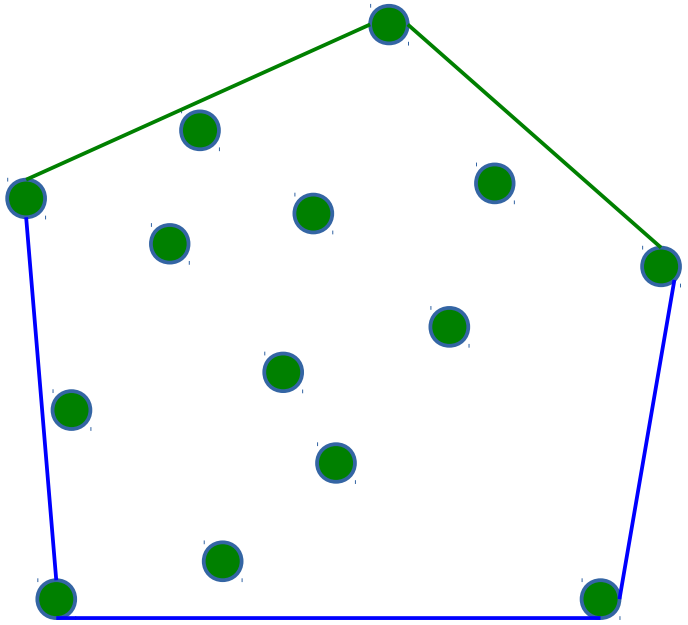
Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Convex Hull

(envoltória convexa / fecho convexo)

Ex.: computador com capacidade de realizar 1.000.000 de comparações se *P* está esq/dir de segmento de reta em um segundo.



Se tivermos uma nuvem com 100.000 de pontos, quanto tempo precisaríamos para Calcular o *convex hull*?

Resposta: 0,5 segundos no n log(n) e 1.000.000 s = 1,9 anos no n^3

Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Problema do Museu (ou da galeria)

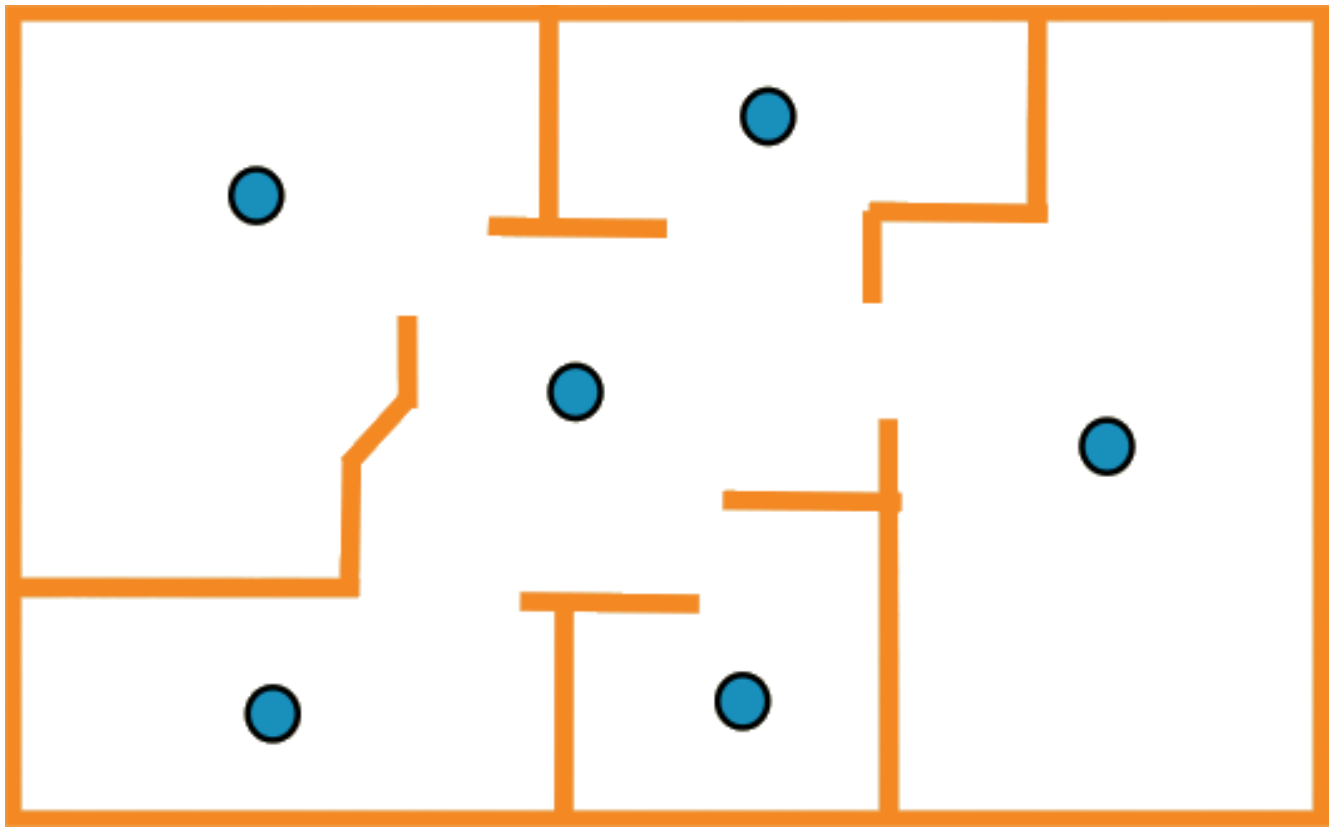
http://pt.wikipedia.org/wiki/Problema_da_galeria_de_arte

Museu da Imigração

MARGS

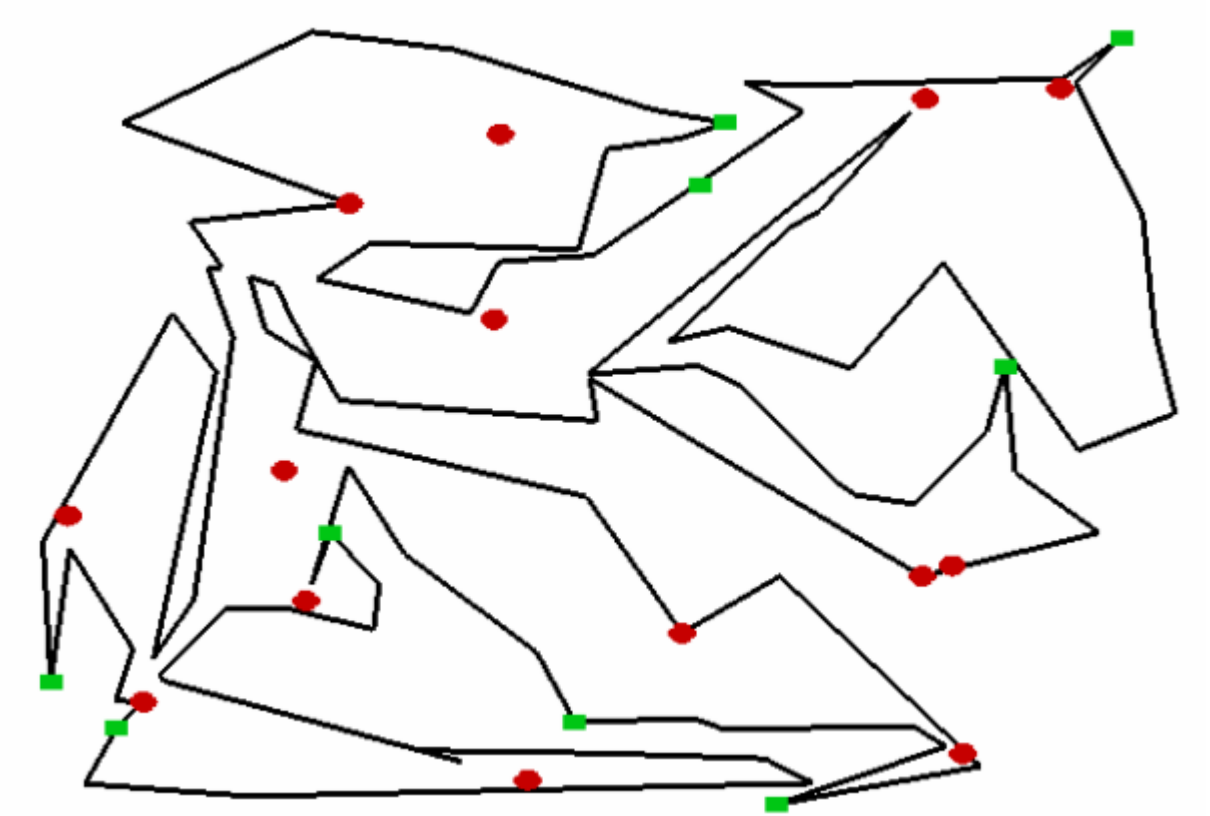
Problema do Museu

- Qual é o menor número *g* de guardas estáticos que guardam uma arte de galeria modelada por um polígono simples com *n* vértices?

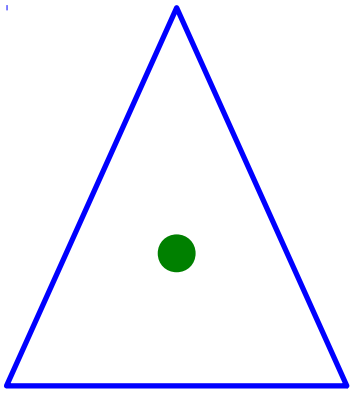


Problema do Museu

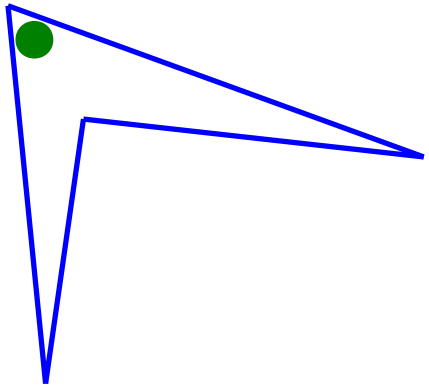
- Qual é o menor número *g* de guardas estáticos que guardam uma arte de galeria modelada por um polígono simples com *n* vértices?



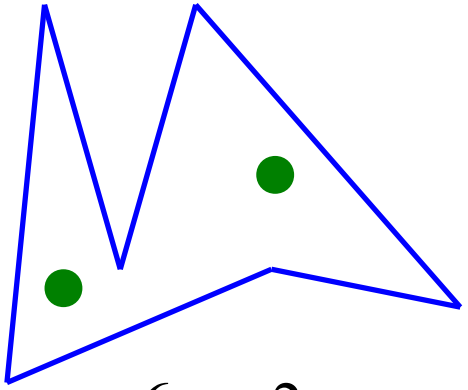
Problema do Museu



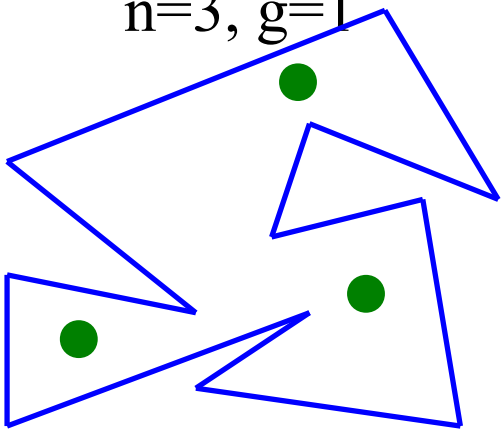
n=3, g=1



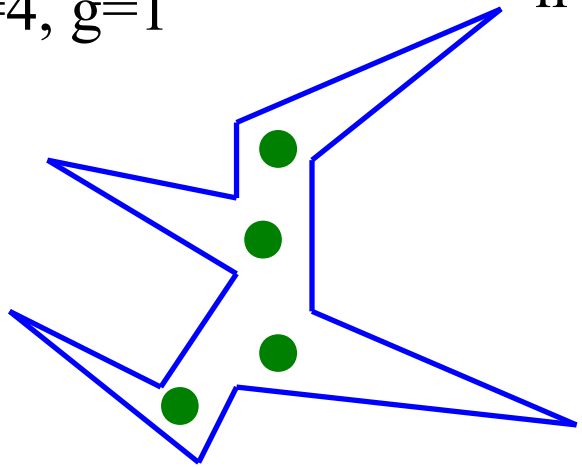
n=4, g=1



n=6, g=2

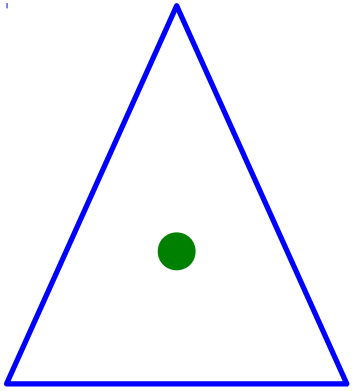


n=12, g=3

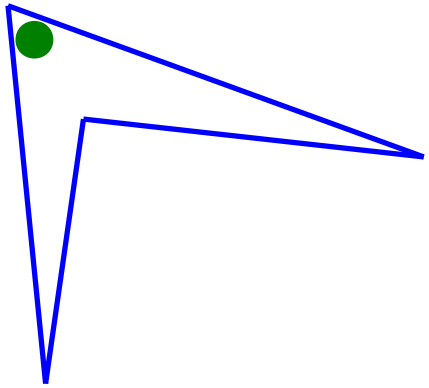


n=12, g=4

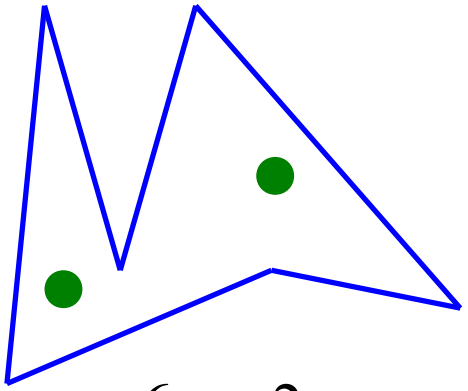
Problema do Museu



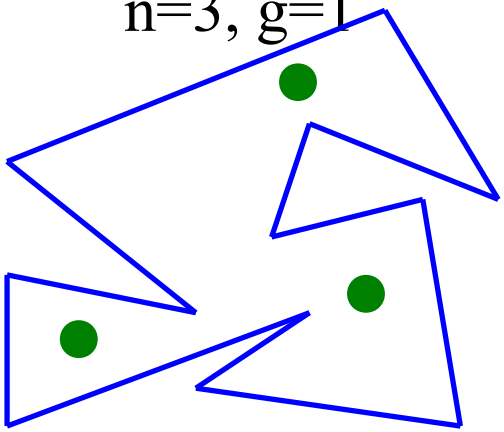
n=3, g=1



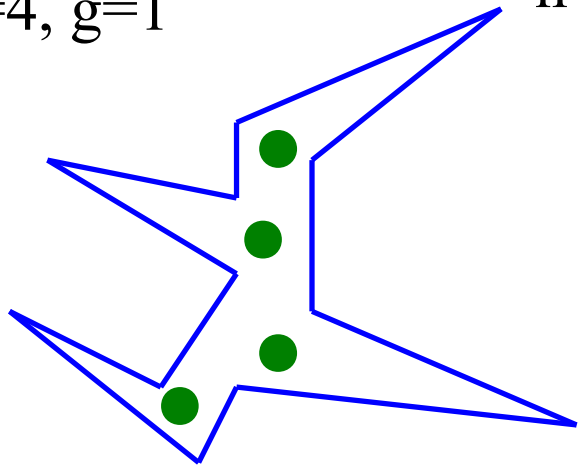
n=4, g=1



n=6, g=2



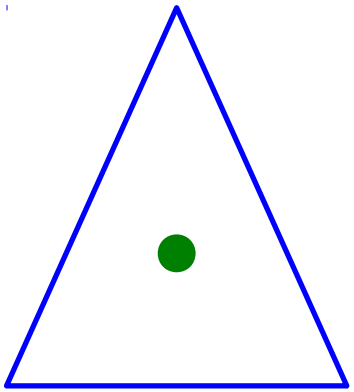
n=12, g=3



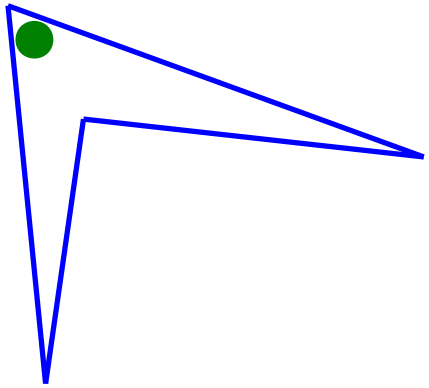
n=12, g=4

G(n) = floor(n/3)?

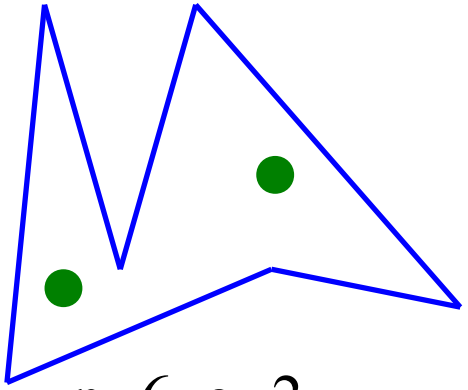
Problema do Museu



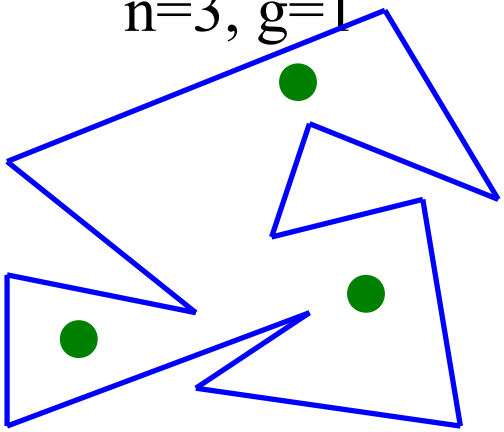
n=3, g=1



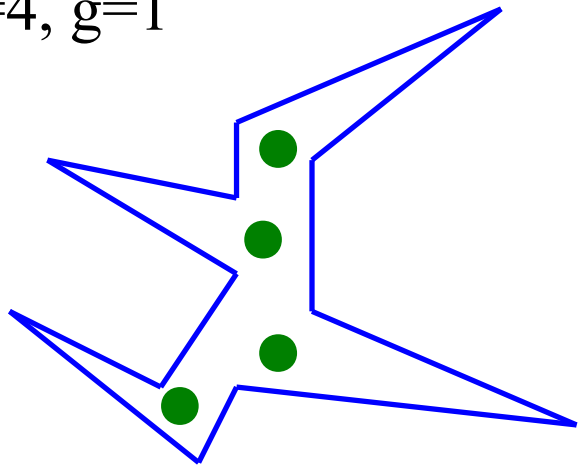
n=4, g=1



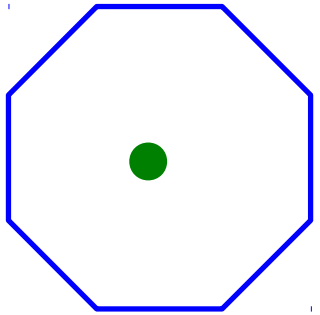
n=6, g=2



n=12, g=3



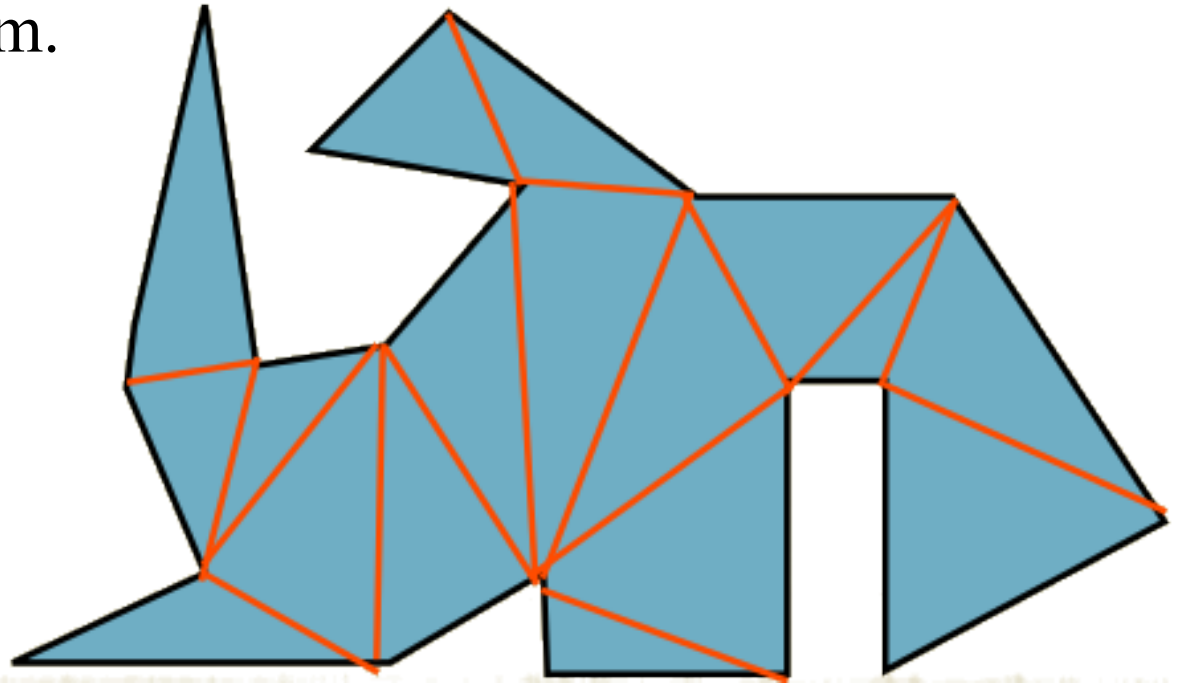
n=12, g=4



n=8, g=1

Problema do Museu

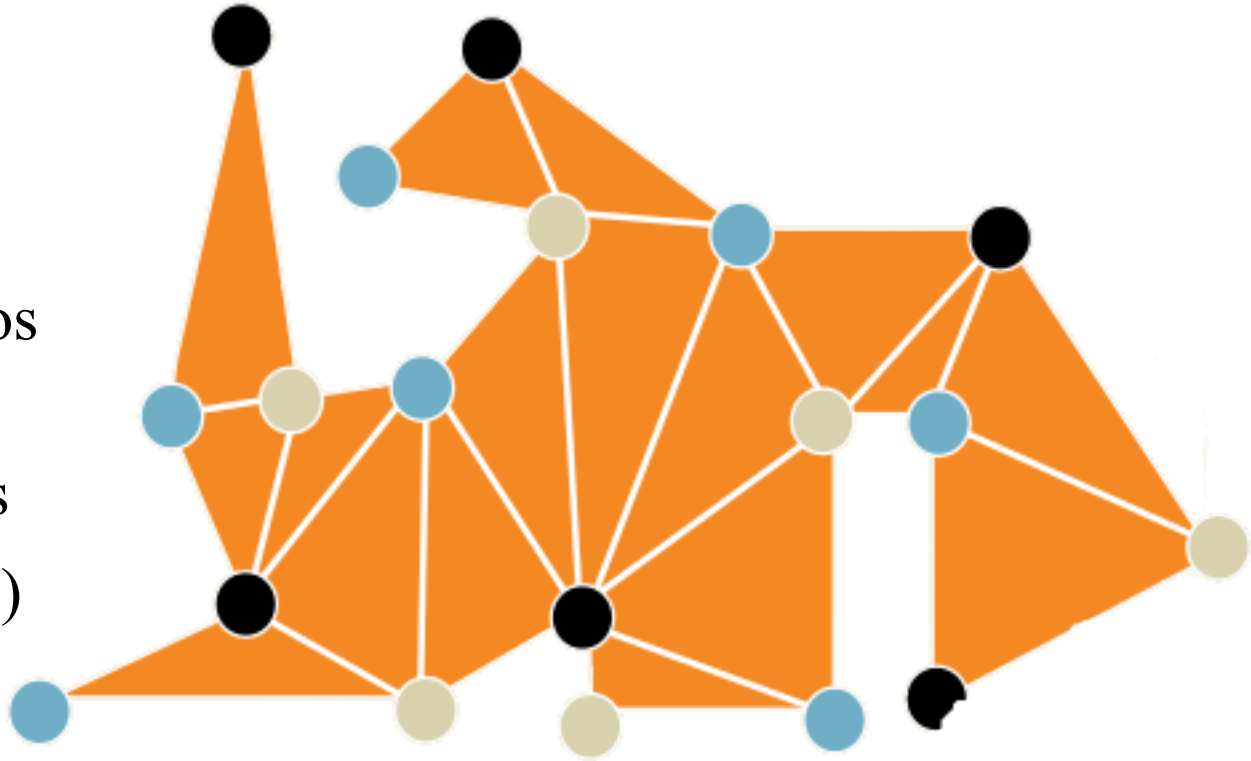
- **Triangulação:** Decomposição de um polígono P em triângulos por um conjunto maximal de diagonais que não se interceptam.



- Toda triangulação de um polígono simples com n vértices consiste de n-2 triângulos

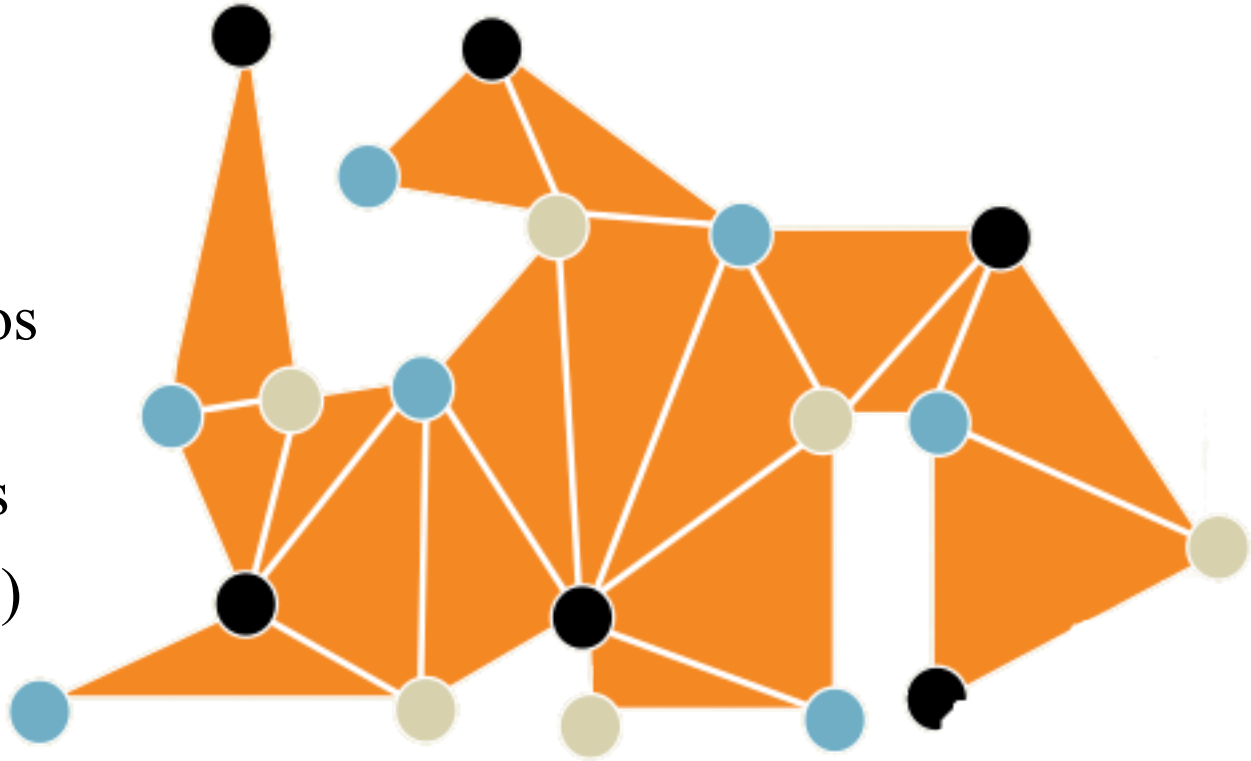
Problema do Museu

- Selecione um subconjunto de vértices como guardas
- Use algoritmo de coloração-3 de grafos
- Escolha a menor classe como guardas
- Resultado: floor(n/3) guardas
- Coloração sempre existe?



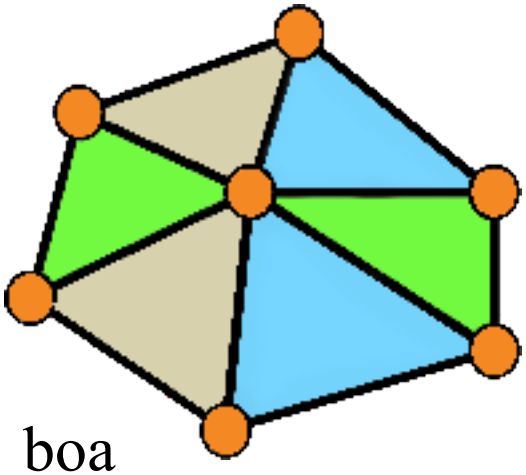
Problema do Museu

- Selecione um subconjunto de vértices como guardas
- Use algoritmo de coloração-3 de grafos
- Escolha a menor classe como guardas
- Resultado: floor(n/3) guardas
- Coloração sempre existe. Desafio: prove!

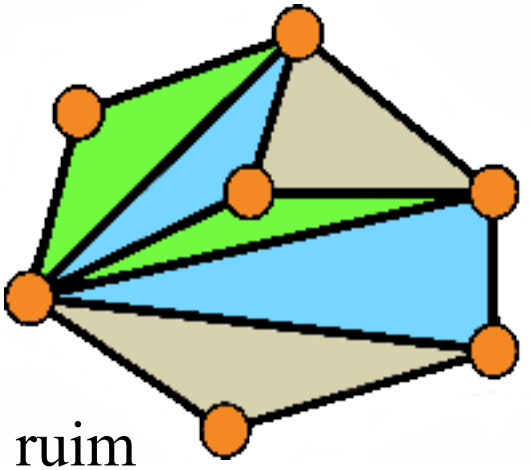


Triangulação de Delaunay

- Seja $\alpha(T)$ o vetor de ângulos de uma triangulação T em ordem crescente.
- Uma triangulação T_1 é "melhor" que T_2 se $\alpha(T_1) > \alpha(T_2)$ lexicograficamente.
- A Triangulação de Delaunay é a melhor de todas



boa



ruim

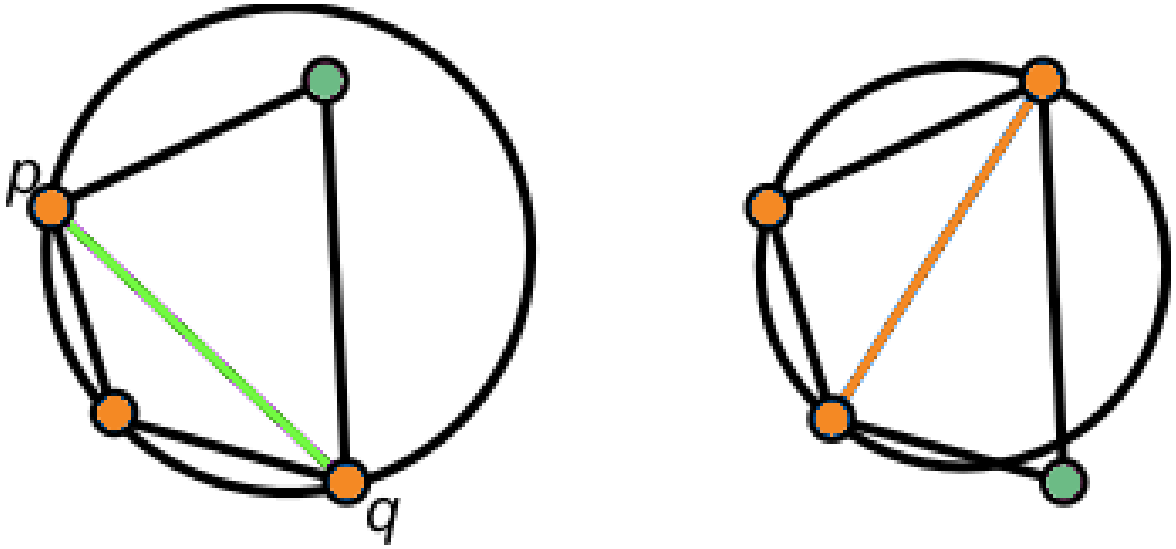
Triangulação de Delaunay

- TEOREMA 1: Seja P um conjunto de pontos, e *T* uma triangulação de P. *T* é uma triangulação de Delaunay de P se e somente se o "circuncirculo" de cada triângulo de *T* não possui nenhum ponto de P no seu interior.

- TEOREMA 2: Seja P um conjunto de pontos no plano. Uma triangulação *T* de P é legal se e somente se *T* é a triangulação de Delaunay de P.

Triangulação de Delaunay

- Lema: Uma aresta pq é ilegal se e somente se um dos seus vértices opostos está **contido** no círculo definido pelos outros dois.



Triangulação de Delaunay

Algoritmo de Delaunay Simples

- Comece com uma triangulação arbitrária. Troque todas as arestas ilegais até que nenhuma mais exista.
- Requer prova que não existe mínimos locais.
- Pode levar um grande tempo para terminar: **$O(n^4)$**

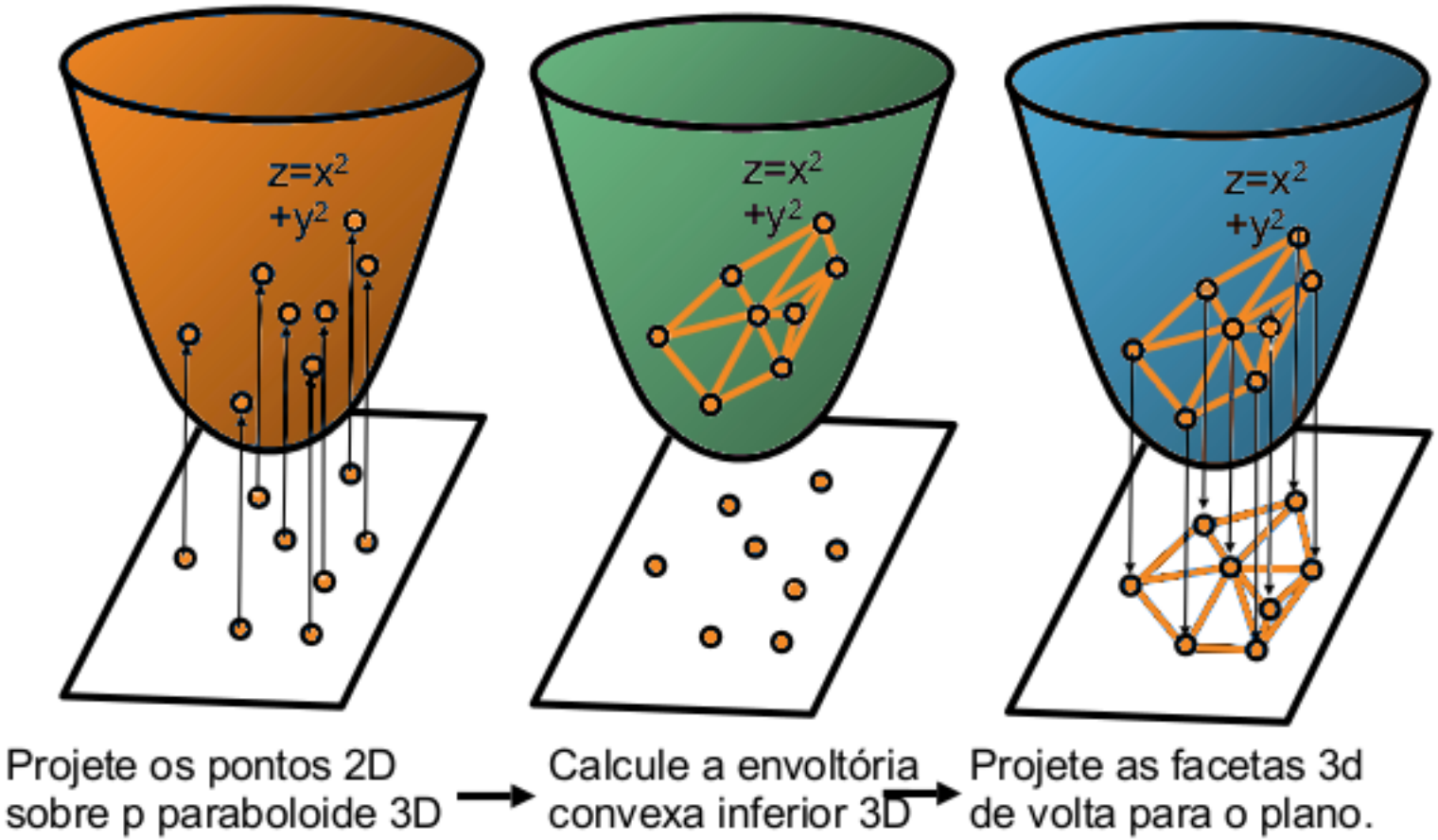
Triangulação de Delaunay

Algoritmo **O(n log n)** para calcular Triangulação de Delaunay

Algoritmo incremental randômico:

- Crie um triângulo que contenha todos os sítios.
- Adicione os sítios um depois do outro em ordem aleatória.
- Se o sítio está dentro de um triângulo:
  - Conecte o sítio aos vértices do triângulo.
  - Cheque se um *flip* pode ser realizado em uma das arestas do triângulo. Caso positivo, cheque recursivamente as arestas vizinhas.
- Se o sítio cai sobre uma aresta:
  - Troque aresta por 4 arestas novas.
  - Cheque se um *flip* pode ser realizado em uma das arestas do triângulo. Caso positivo, cheque recursivamente as arestas vizinhas.

Triangulação de Delaunay e *convex hull*



A triangulação 2D é Delaunay!

Diagrama de Voronoi

- Tesselagens de Dirichlet (1850)
- Descarte (1644)
- **Voronoi** (1907)
- Brown (1954): *Area Potentially available to a tree*
- Mead (1966): *plant polygons*

Diagrama de Voronoi

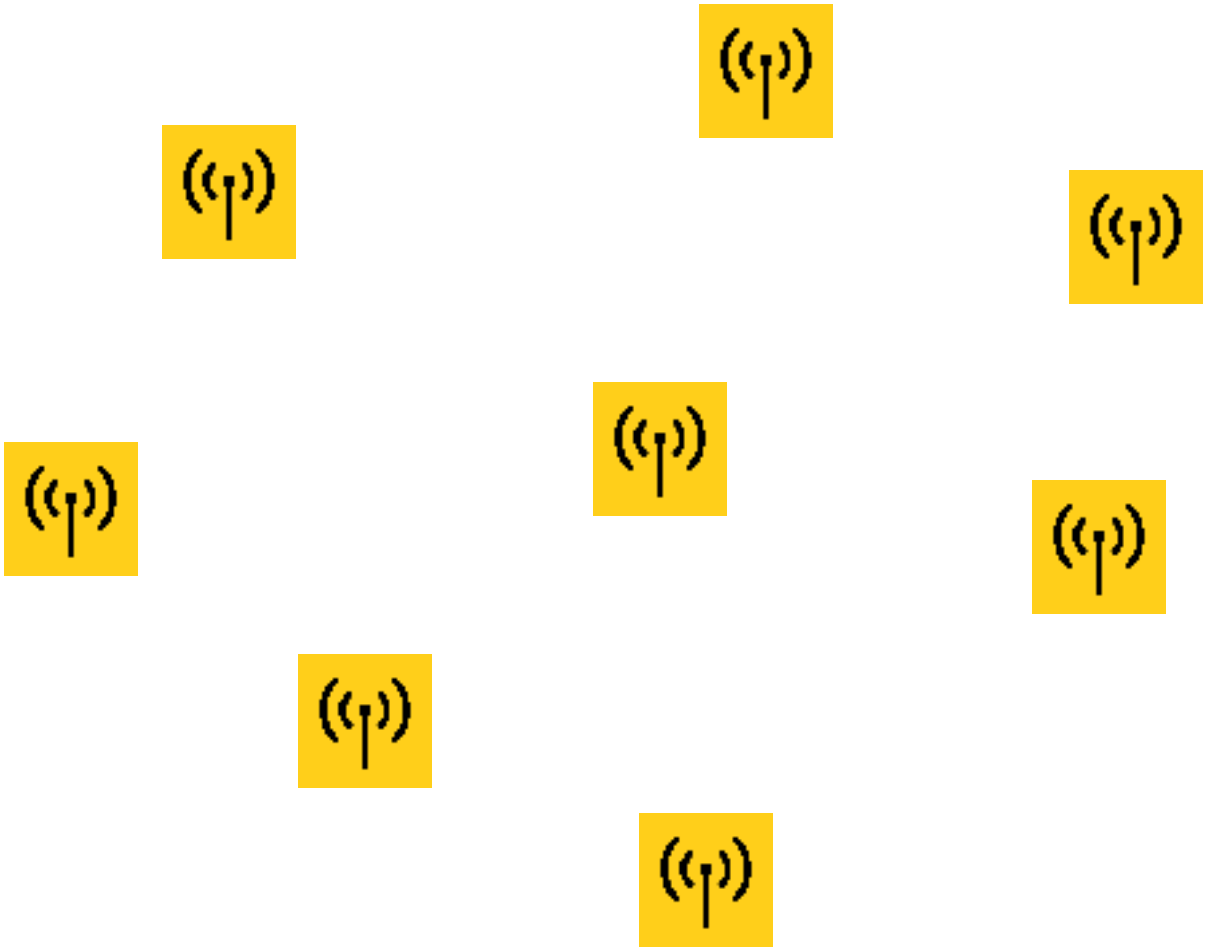


Diagrama de Voronoi

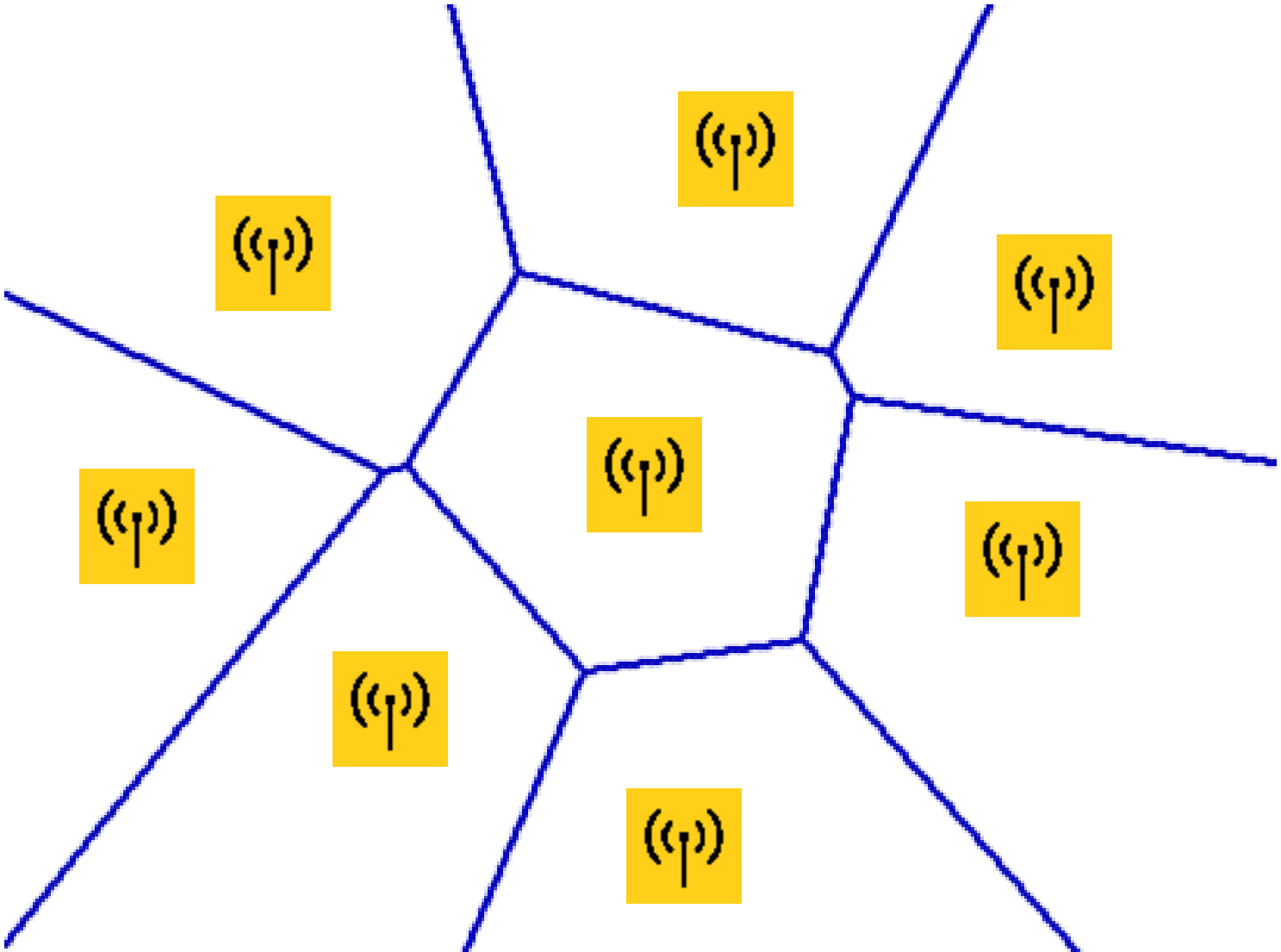


Diagrama de Voronoi

- Teorema: Um ponto **q** é um vértice de Vor(P) se e somente se o seu maior círculo vazio Cp(q) contém 3 ou mais sítios na sua fronteira.

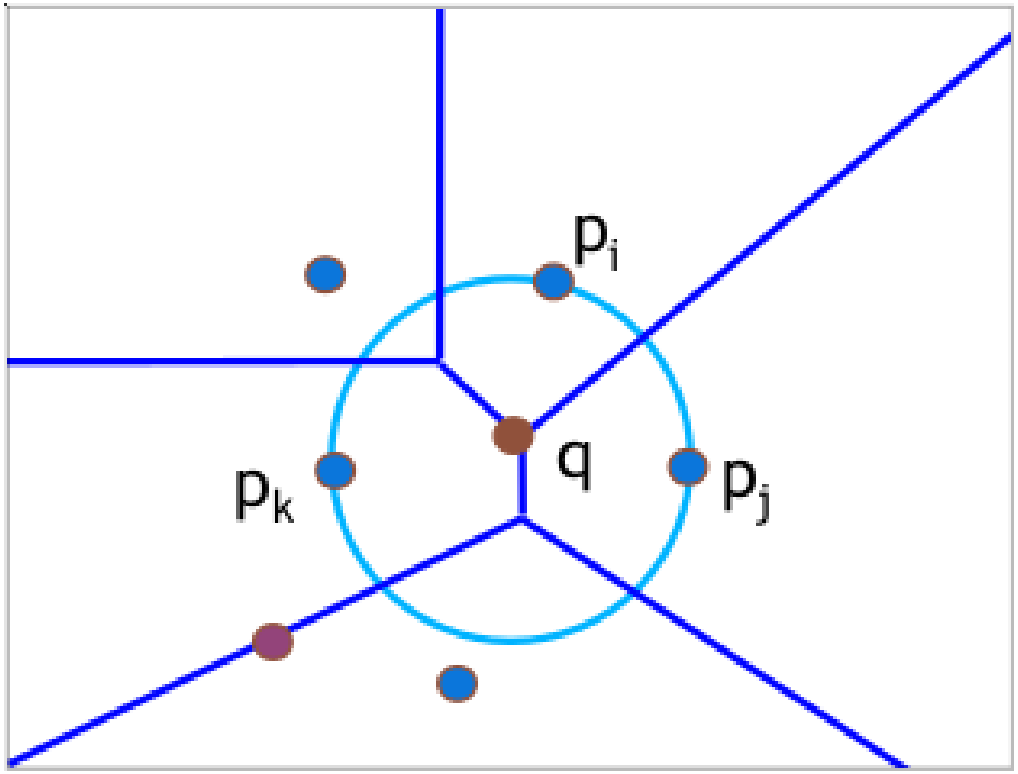


Diagrama de Voronoi

- Teorema: O bissetor entre os sítios p_i e p_j definem uma aresta de Vor(P) se e somente se existe um ponto q tal que Cp(q) contém ambos p_i e p_j na sua fronteira, e mais nenhum sítio.

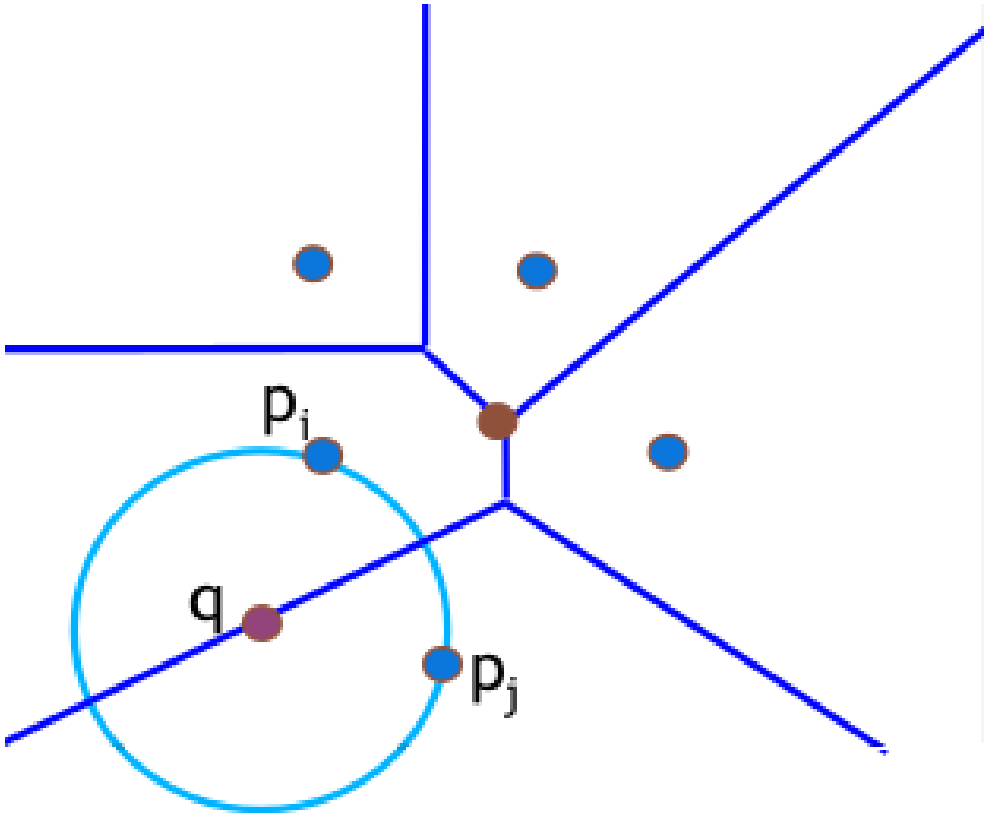


Diagrama de Voronoi

- Corolário: Cada célula em um diagrama de Voronoi é um polígono convexo, possivelmente não fechado.
- Construção:
  - Construir o bissetor entre um sítio e todos os outros
  - Uma célula de Voronoi é a interseção de todos semi-espaços definidos pelos bissetores.
  - Complexidade: O(n log n) para cada célula.

Diagrama de Voronoi

- Construção:
  - Usar observação que as células são interseções de semi-espaços
  - Calculo de Interseções de semi-espaço: O(n log n) para cada célula
  - Calculo de Voronoi **(n^2 log n)**

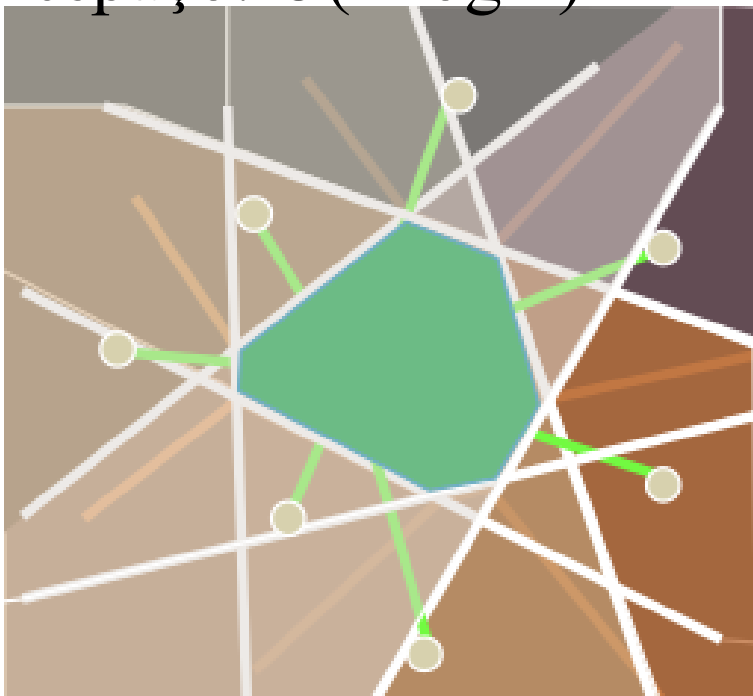


Diagrama de Voronoi

- Construção:
  - Usar observação que as células são interseções de semi-espaços
  - Calculo de Interseções de semi-espaço: O(n log n) para cada célula
  - Calculo de Voronoi **(n^2 log n)**

Dá para fazer melhor?

Diagrama de Voronoi

- Sweep de Plano
  - Reduz um problema n dimensional para um (n-1) dinâmico
  - Varrer um espaco nD por um hiperplano (n-1)D
  - Manter as interseções do hiperplano com um subconjunto de elementos (conjunto ativo)
  - Interseções atualizam-se continuamente com tempo, com exceção de quando um evento discreto acontece, novos elementos tornam-se ativos ou deixam de ser.

Diagrama de Voronoi - Algoritmo de Fortune

- A interseção de dois cones (referentes ao sítios p e q) é uma curva (hipérbole) contida num plano vertical
- Se projetada no plano que contém p e q, é igual ao bissetor de p e q.

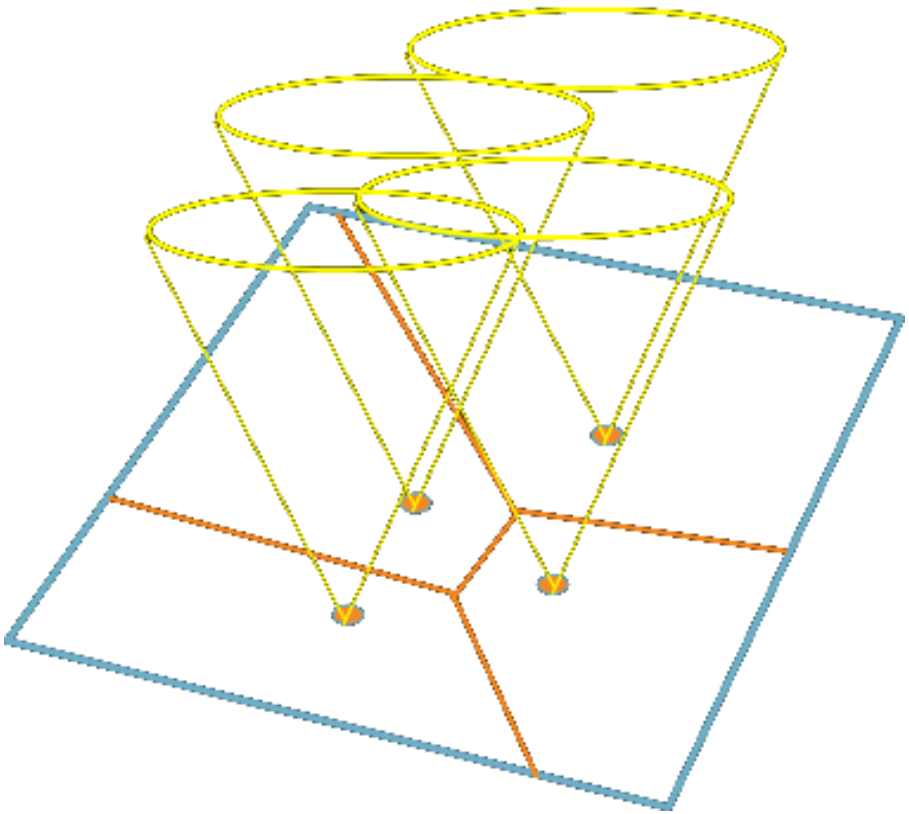


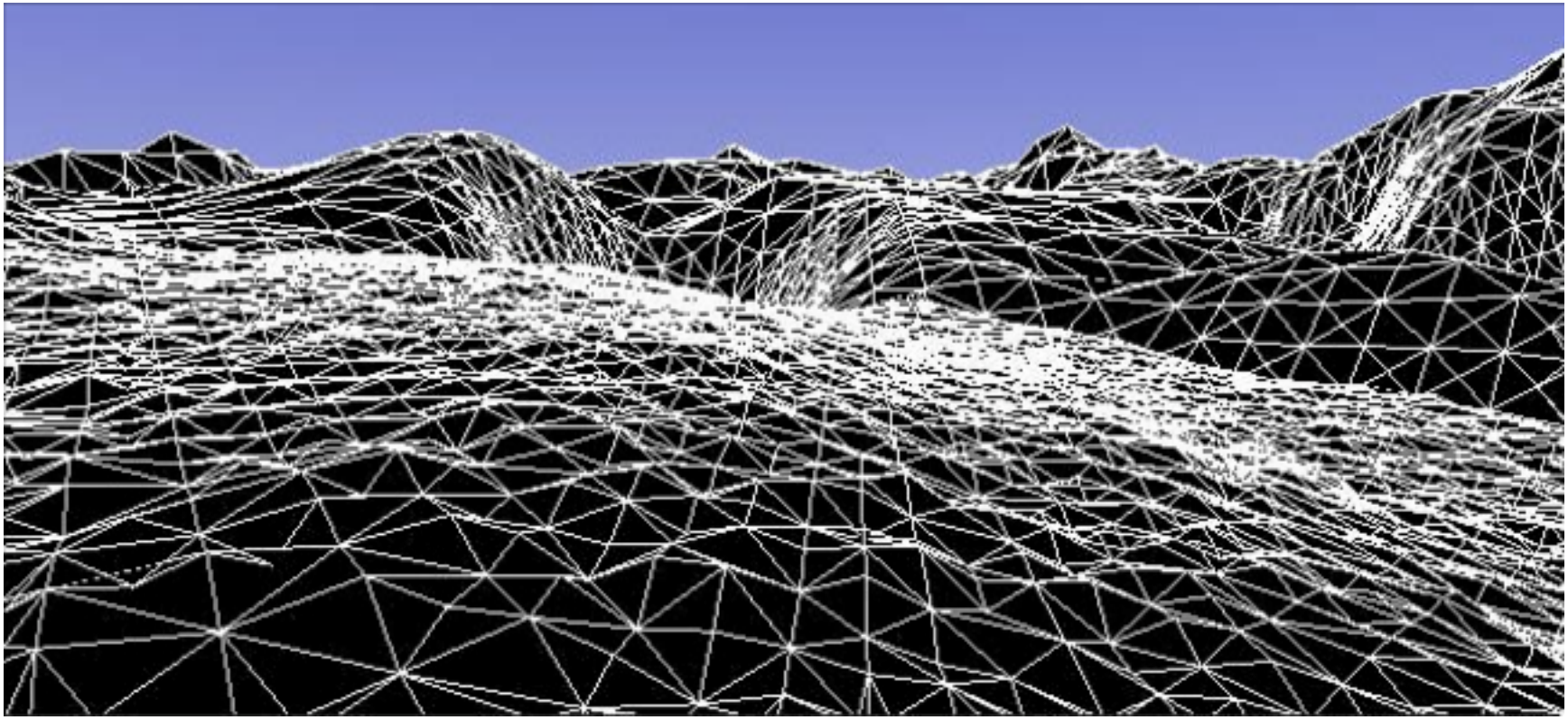
Diagrama de Voronoi - Algoritmo de Fortune

- Complexidade
  - O algoritmo roda em O(n log n) e usa O(n) de memória
- Estrutura de dados
  - Operações em T: O(log n)
  - Operações na lista de aresta: constante
  - Operações na fila de eventos: O(log n)
  - Operações em eventos: constante
- Custo de um evento: O(log n)
- n eventos de sitio
- número de eventos de círculo: 2n-5 no máximo

LOD (*Level Of Detail* – Nível de Detalhe)

- Permite várias representações de um mesmo objeto, sendo usadas de acordo com a distância entre o objeto e o observador.
- Em uma cena ideal, todos os polígonos devem ter o mesmo tamanho em *pixels* na tela.
- Pela natureza da projeção perspectiva, a cena deve ter mais polígonos próximos ao observador do que longe dele.

LOD (*Level Of Detail* – Nível de Detalhe)



LOD (*Level Of Detail* – Nível de Detalhe)

69,451 triangles

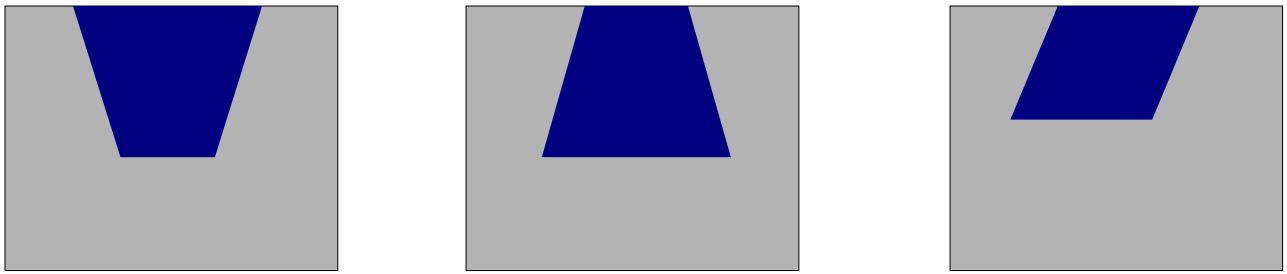
2,502 triangles

251 triangles

76 triangles

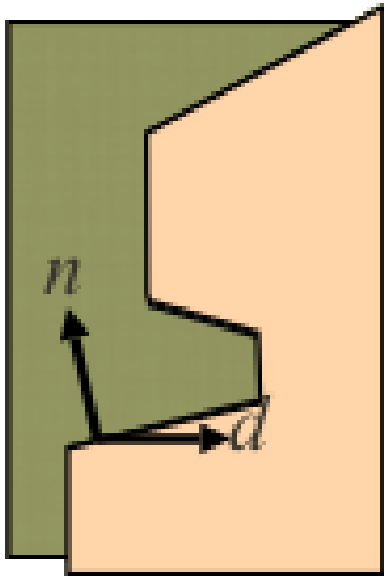
Problema do Molde

- Como remover uma peça de um molde?
- Restrições:
  - Molde formado por uma única peça;
  - Apenas objetos formados por poliedros;
  - Remover realizando apenas uma translação.



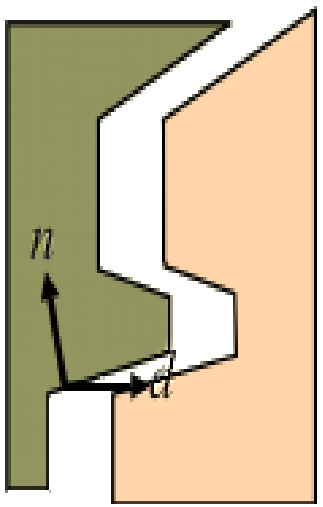
Problema do Molde

- Verificar se é possível remover o objeto
- Encontrar uma direção viável
  - O que é uma direção viável para remoção?



Problema do Molde

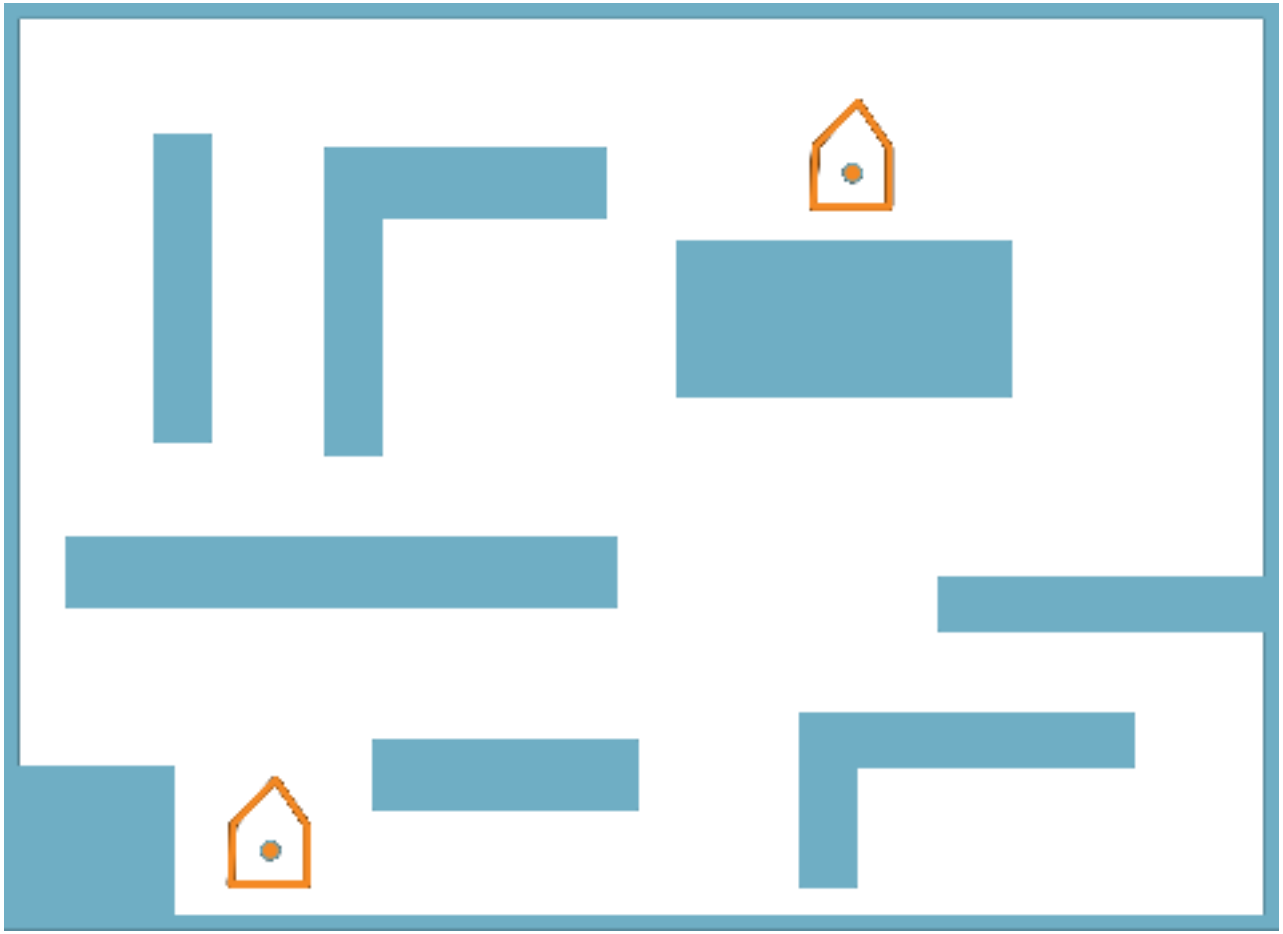
- Verificar se é possível remover o objeto
- Encontrar uma direção viável
  - O que é uma direção viável para remoção?
  - Ângulo com a normal de cada face interna do molde deve ser maior que 90 graus



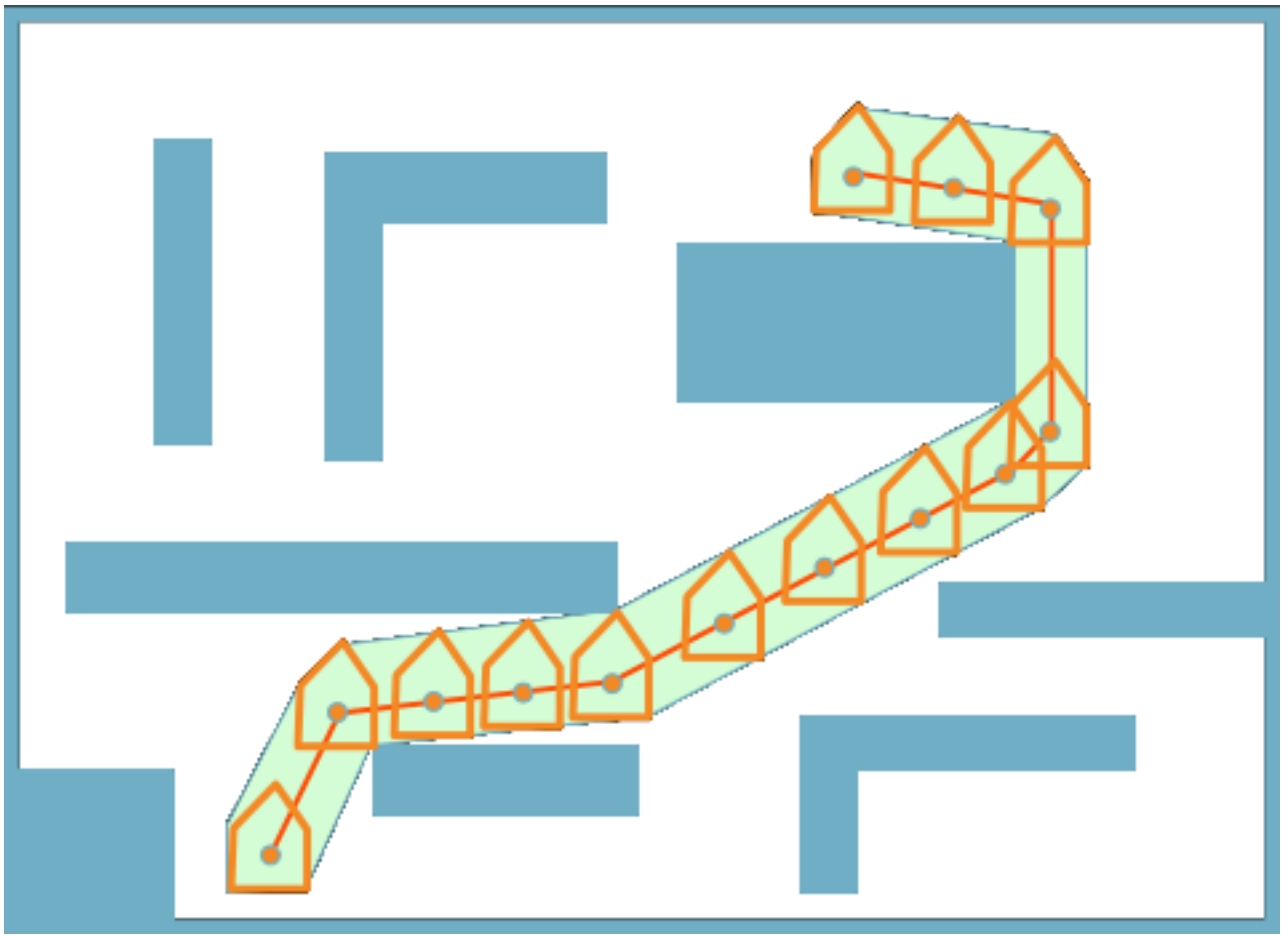
Problema do Molde

- Técnicas para resolver o problema:
  - Intersecção de semiplanos
  - Intersecção de regiões convexas
  - Programação linear incremental
  - Programação linear estocástica ("randomizada")

Movimento de Robôs



Movimento de Robôs

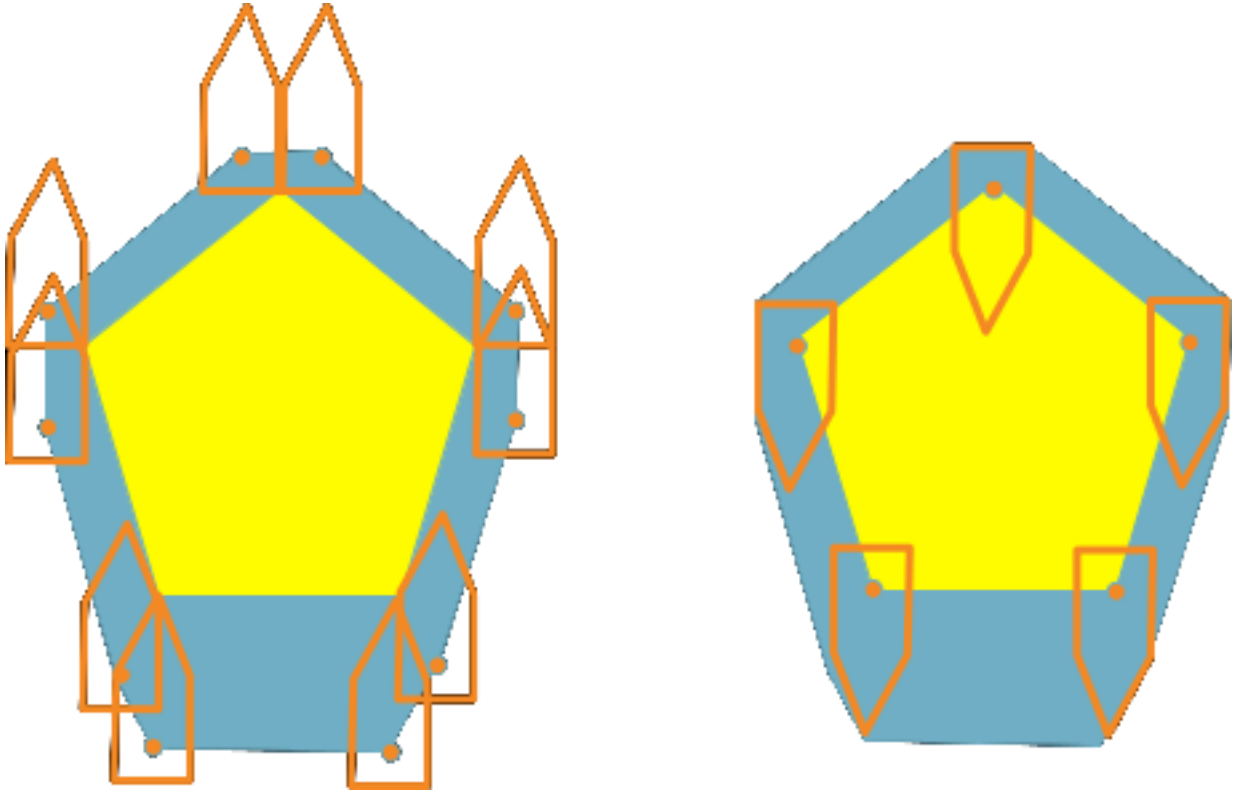


Movimento de Robôs

- Espaço de Configuração Proibido
  - Espaço dos parâmetros de um robô R onde o robô colide com o ambiente
- Espaço de Configuração Livre
  - Espaço dos parâmetros de um robô R onde o robô não colide com o ambiente
- Espaço de Configuração Obstáculos
  - Espaço dos parâmetros dos obstáculos mapeados para o espaço de configuração

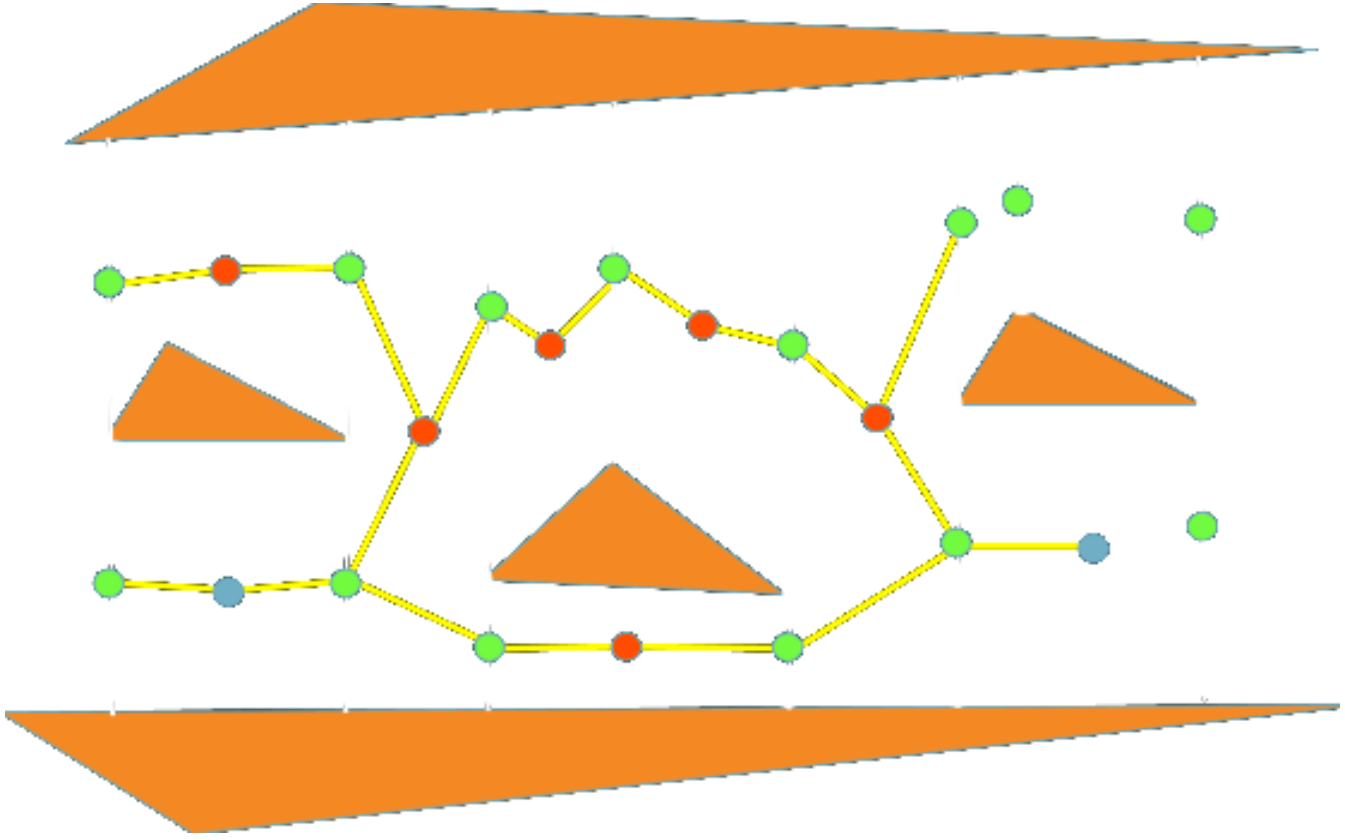
Soma de Minkovsky

- Aumentar as paredes (dilatação morfológica), para possibilitar levar em consideração a movimentação do robô ou personagem como sendo pontual.

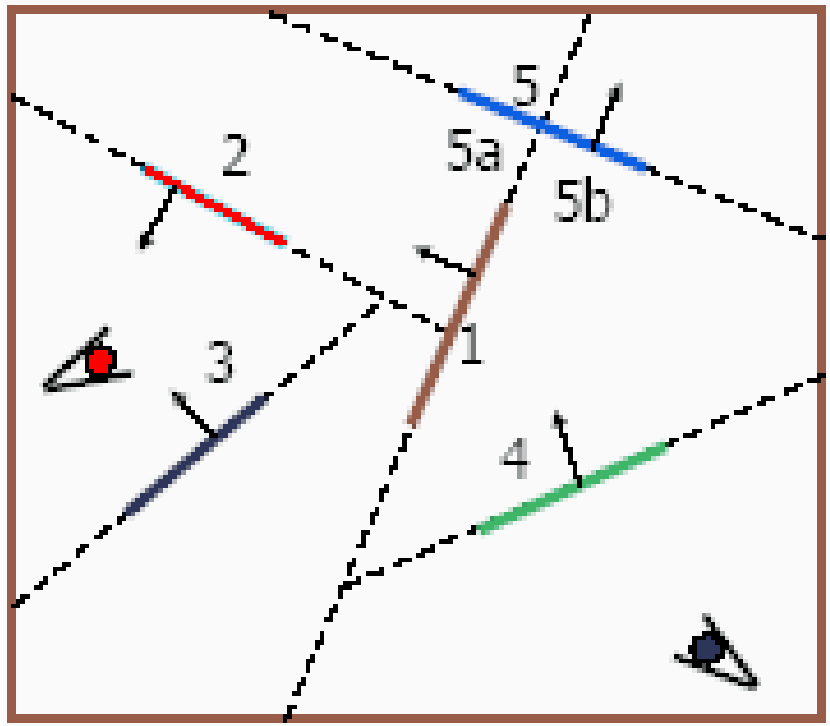


Movimento de Robôs

- Considerar algoritmo de visibilidade e algoritmos de caminhos mais curtos, como A*.

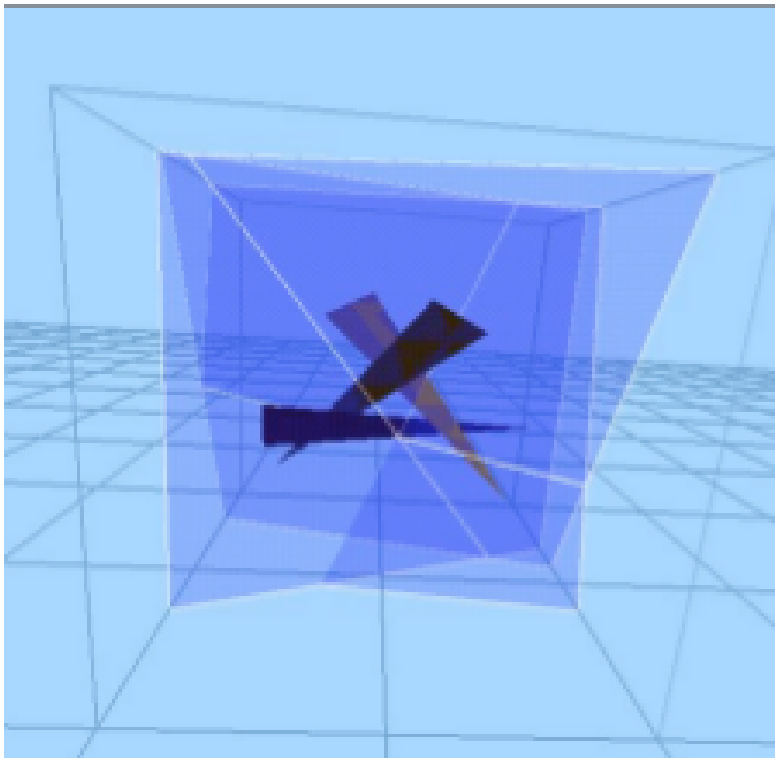


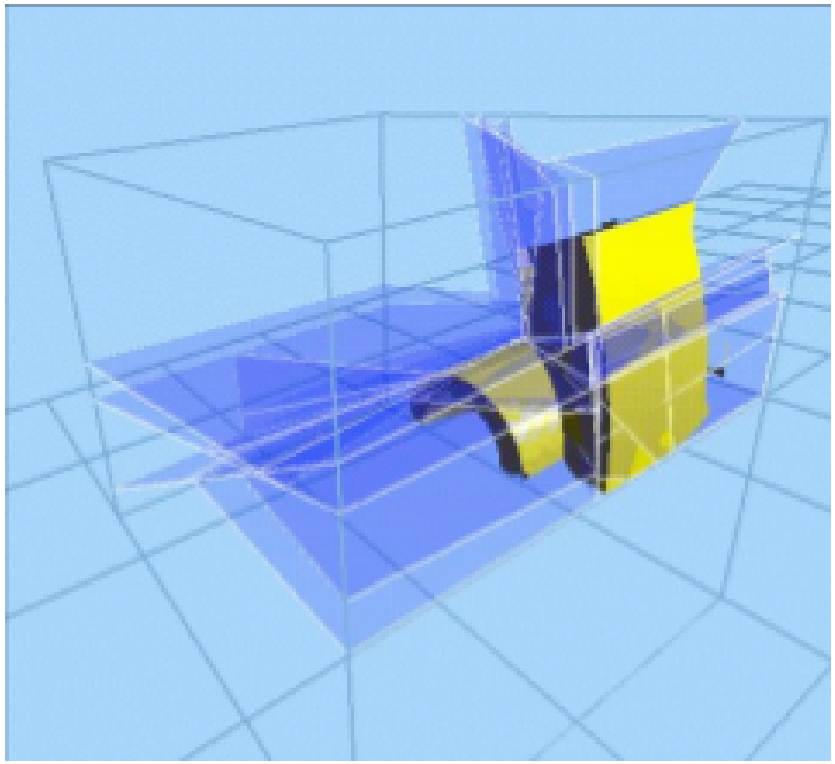
BSP-Trees para visibilidade



BSP-Trees para visibilidade

- BSP em 3D





- Topological BSP-Tree in 3D (ver trabalho Comba)