Algoritmos Randomizados: Geometria Computacional

Celina Figueiredo

Guilherme Fonseca

Manoel Lemos

Vinicius de Sá

26º Colóquio Brasileiro de Matemática

IMPA – Rio de Janeiro – Brasil

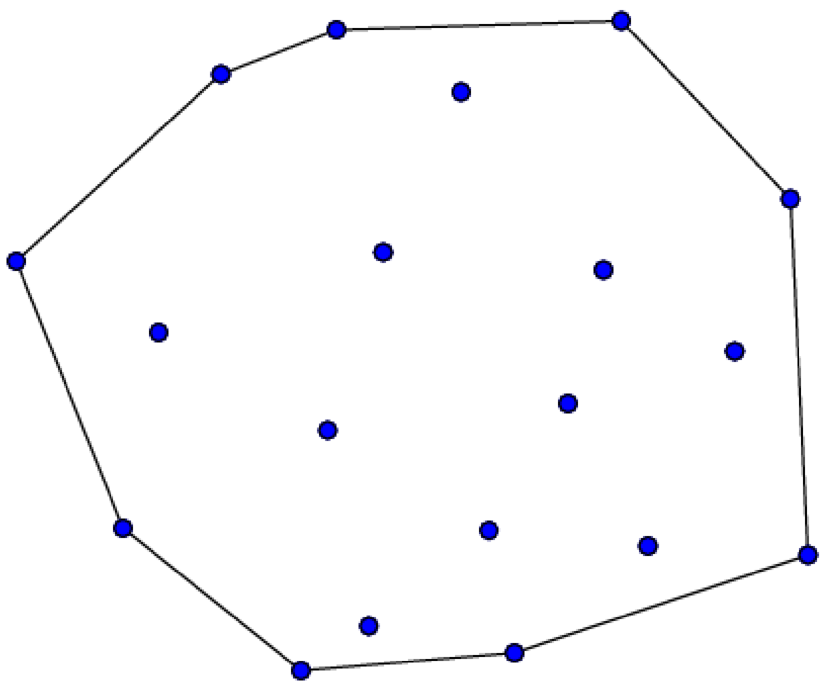
2007

Resumo

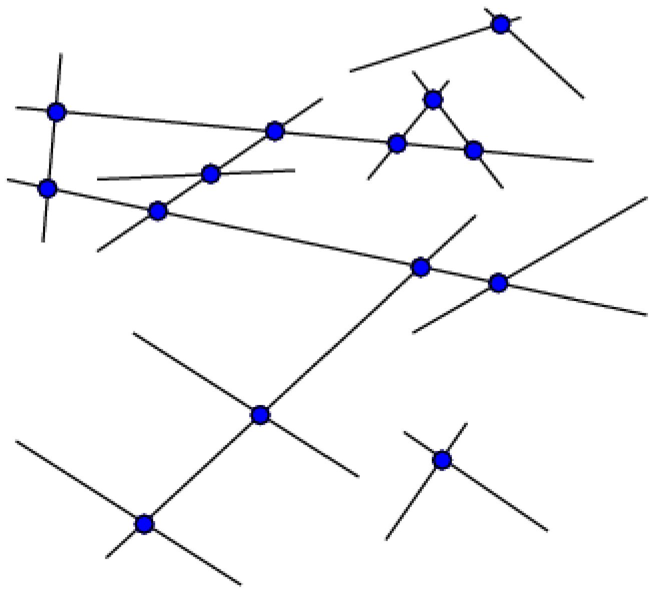
- Introdução à Geometria Computacional
- Programação Linear
- Funções Hash (sem detalhes)
- Par de Pontos Mais Próximos

Geometria Computacional

- Solução de problemas geométricos:



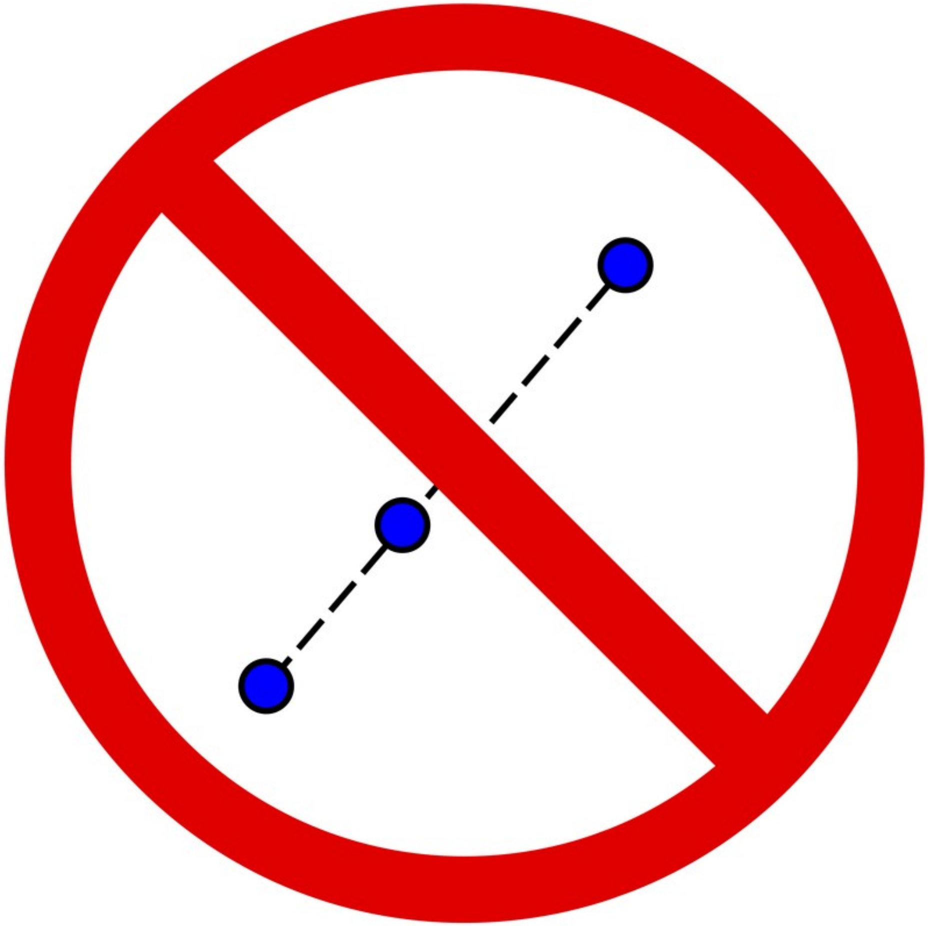
Fecho convexo



Interseção de segmentos

O Que Se Assume

- Modelo computacional *RAM real*. Operações algébricas exatas
- Espaço d-dimensional, onde d é uma *constante*
- Posição geral: ignora situações sensíveis a perturbações infinitesimais
  - Três pontos colineares
  - Retas paralelas
  - Retas verticais



O Poder da Randomização

- Geralmente, os algoritmos randomizados têm a *mesma complexidade de tempo* dos determinísticos
- Porém:
  - São mais simples
  - Mais rápidos na prática
  - Generalizam para dimensões arbitrárias
- *Derandomização*

Programação Linear

- Consiste em maximizar uma função linear (*função objetivo*) satisfazendo um conjunto de desigualdades lineares (*restrições*)

Maximizar:

$$f(x, y) = x - 2y$$

Função objetivo

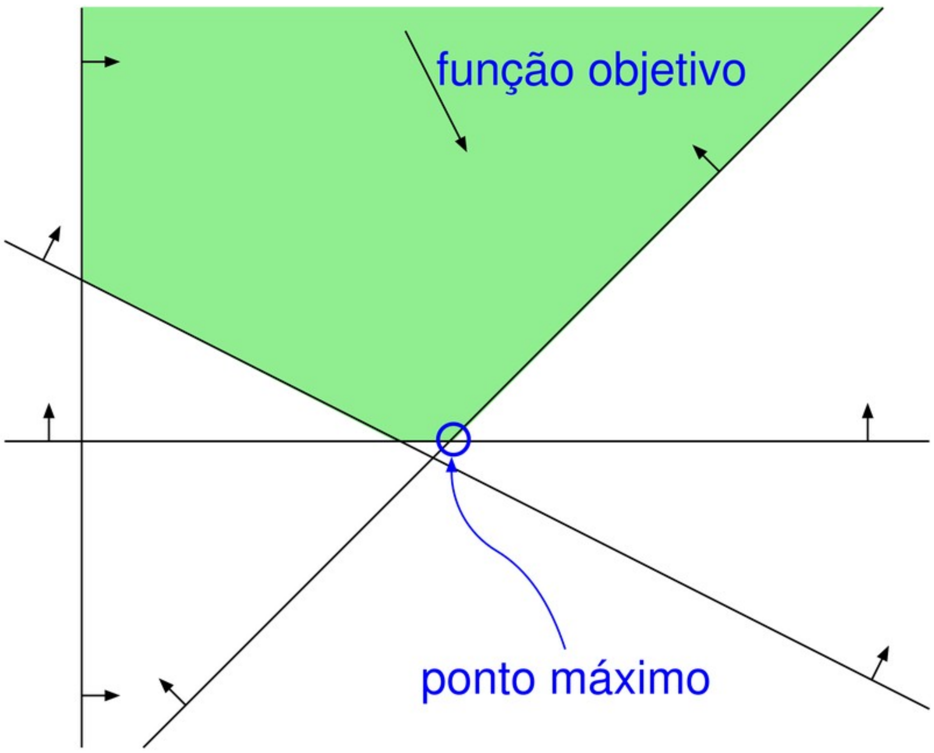
Satisfazendo:

$$\begin{aligned} -2x &\leq 10 \\ -x - 2y &\leq 2 \\ x - y &\leq 2 \\ -3y &\leq 3 \end{aligned}$$

Restrições

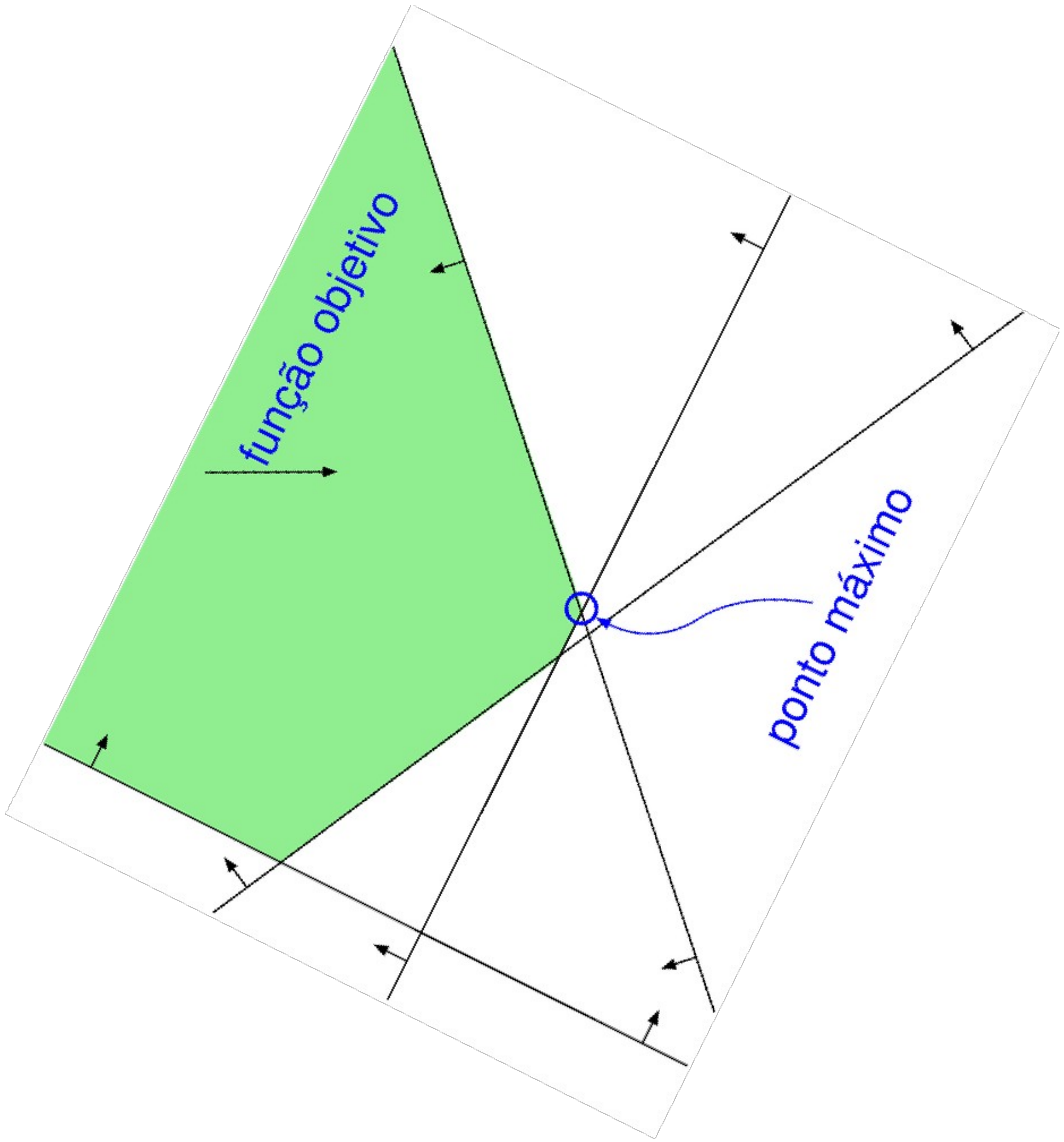
Interpretação Geométrica

- Determinar o ponto máximo em uma dada direção que esteja contido na interseção de um conjunto de semi-espaços (ou semi-planos).



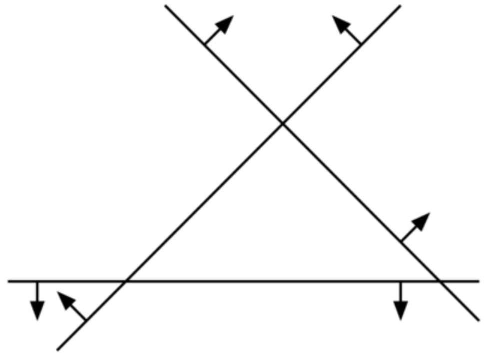
Rotação

- Consideramos que a função objetivo sempre aponta para a direita
- Não há perda de generalidade
- Basta aplicar uma rotação

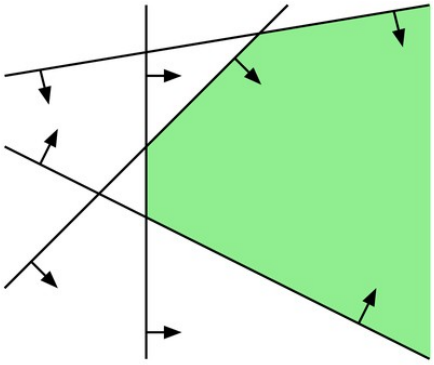


Tipos de Solução

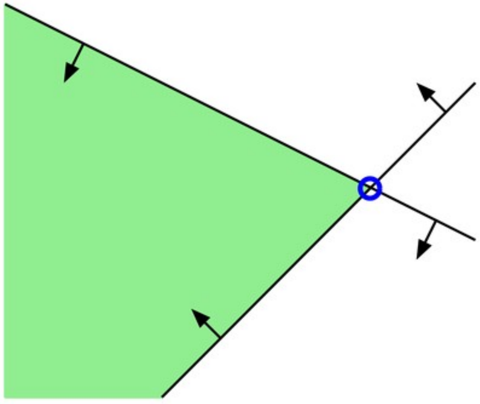
- Problema inviável



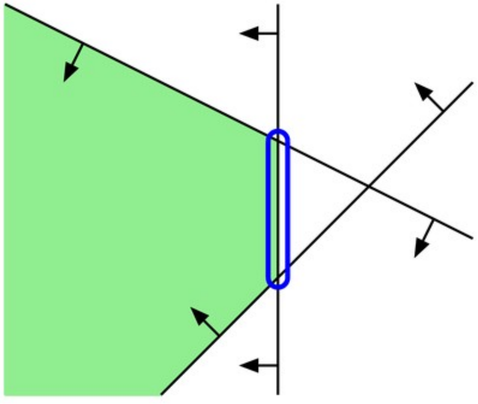
- Problema ilimitado



- Solução única

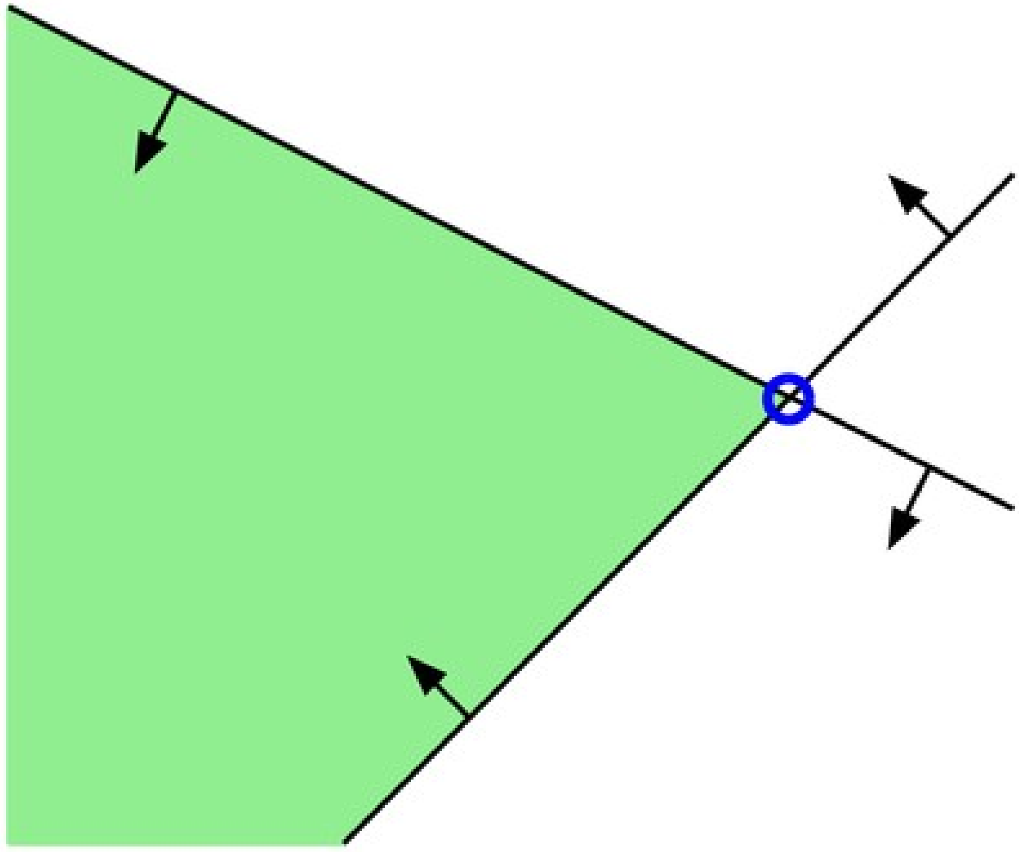


- Infinitas soluções



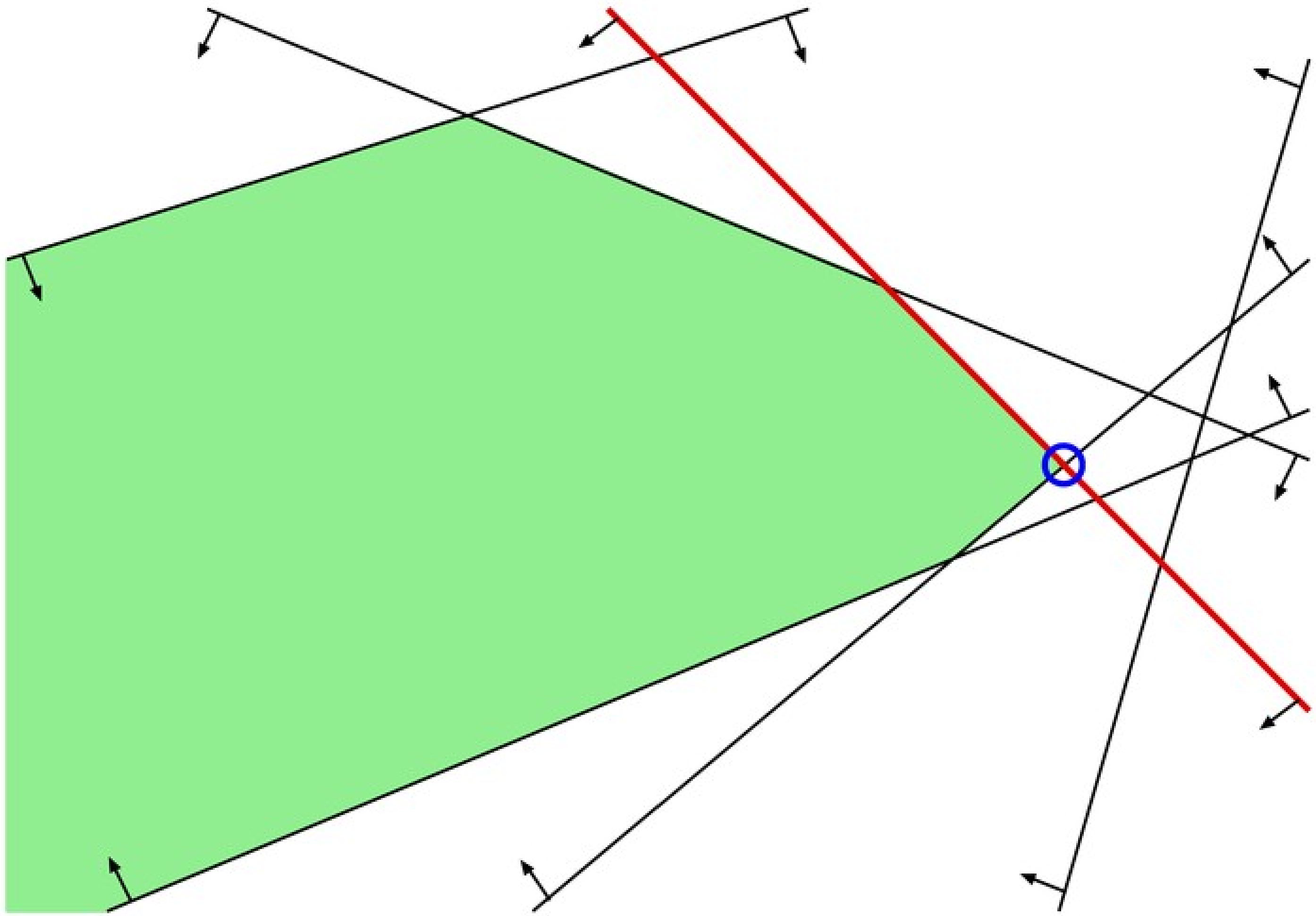
Simplificações

- Posição geral
- Somente 2 variáveis
  (fácil de generalizar)
- Conhecemos 2 restrições que *limitam* o problema
  (fácil para d=2, nem tão fácil no geral)

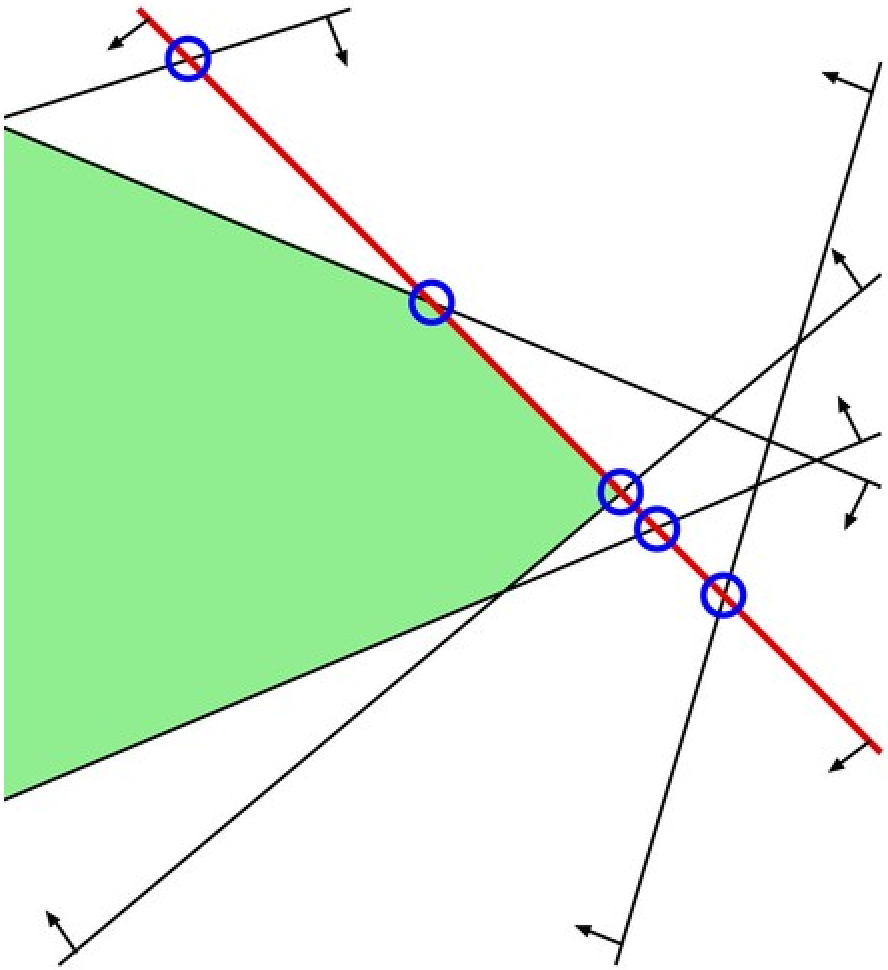


Construção Incremental

- Inicia-se resolvendo um problema trivialmente pequeno
- A cada passo, adiciona-se um elemento aleatório da entrada
- Atualiza-se a solução
- Repete até que todos os elementos tenham sido adicionados



Tempo de Execução



- Inserir a i-ésima restrição leva tempo O(1) no melhor caso, mas O(i) no pior caso
- Portanto, o tempo de execução é, no pior caso:

$$T(n) = \sum_{i=3}^{n} O(i) = O(n^2)$$

- Note que esse pior caso realmente pode ocorrer

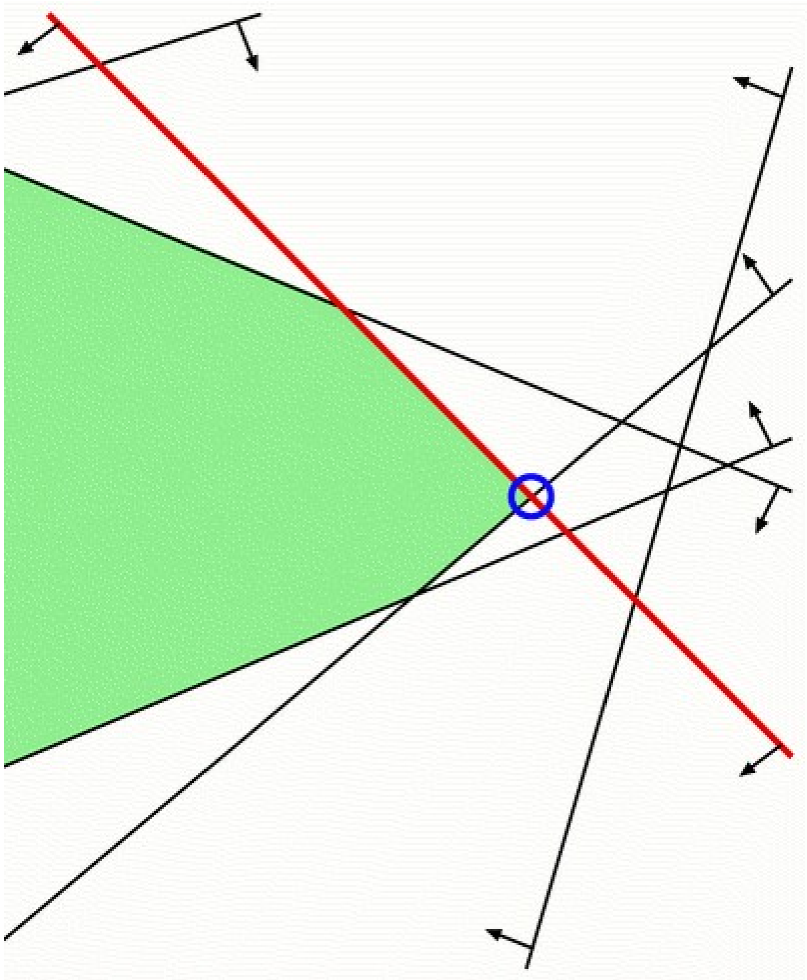
Tempo de Execução 2

- Porém, podemos usar o fato das restrições serem inseridas em ordem aleatória
- Seja q_i a probabilidade da i-ésima restrição alterar o valor da solução
- O valor esperado do tempo de execução é:

$$\sum_{i=3}^{n} (q_i O(i) + (1 - q_i) O(1))$$

Análise De Trás Para Frente

- Podemos imaginar que as restrições são removidas uma a uma aleatoriamente
- Qual a probabilidade q_i da solução ser alterada quando uma restrição aleatória (dentre i restrições) é removida?



Análise De Trás Para Frente

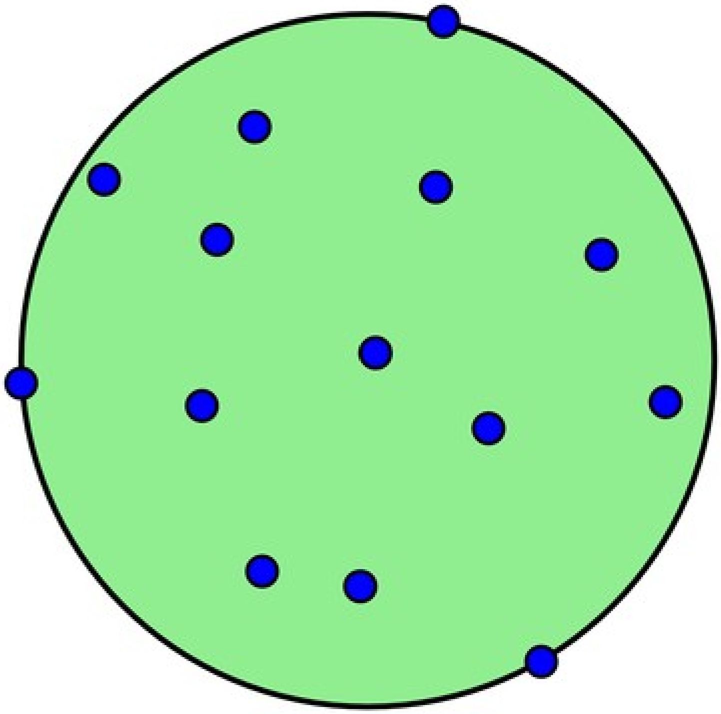
- Podemos imaginar que as restrições são removidas uma a uma aleatoriamente
- Qual a probabilidade q_i da solução ser alterada quando uma restrição aleatória (dentre i restrições) é removida?
- No máximo 2 dentre i-2 restrições alteram a solução
- Portanto:

$$q_i \leq \frac{2}{i-2} = O\left(\frac{1}{i}\right)$$

- E concluímos:

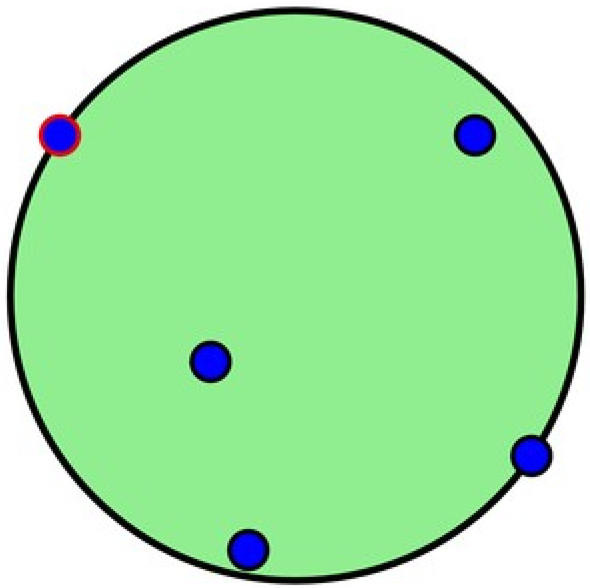
$$\begin{aligned} E(T(n)) &= \sum_{i=3}^{n} \left(O\left(\frac{1}{i}\right) O(i) + \left(1 - O\left(\frac{1}{i}\right)\right) O(1) \right) \\ &= \sum_{i=3}^{n} O(1) = O(n) \end{aligned}$$

Círculo Mínimo



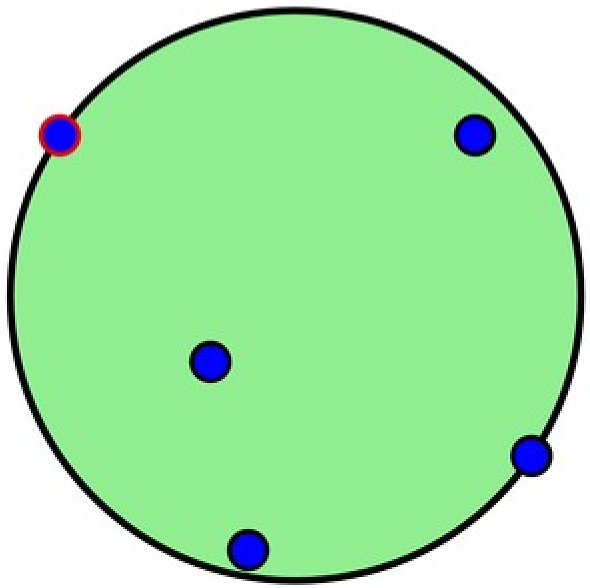
- Determinar o menor círculo que contém um conjunto de pontos
- Solução em tempo O(n) usando um algoritmo randomizado incremental
- Semelhante à programação linear (com d=3)

Círculo Mínimo



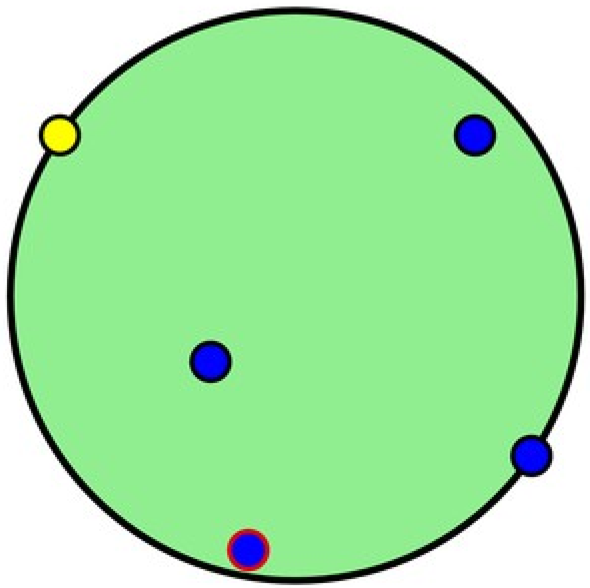
- Iniciar com 2 pontos
- Inserir um ponto por vez
- Dentro do círculo é fácil!
- Fora do círculo é mais difícil
- Como resolver?
- Dica: novo ponto está na borda

Círculo Mínimo Restrito



- Acrescenta-se a restrição de que um dado ponto deve estar na borda
- Resolve-se com construção randomizada incremental
- Agora semelhante a programação linear com d=2

Círculo Mínimo Restrito

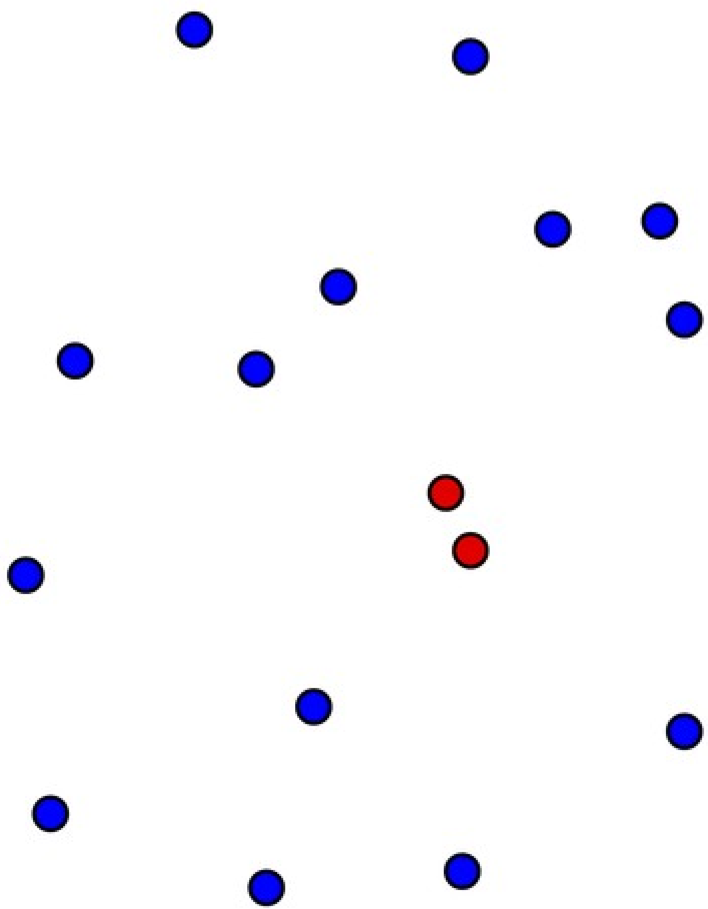


- Acrescenta-se a restrição de que um dado ponto deve estar na borda
- Resolve-se com construção randomizada incremental
- Agora semelhante a programação linear com d=2
- Tempo: O(n)

Funções Hash

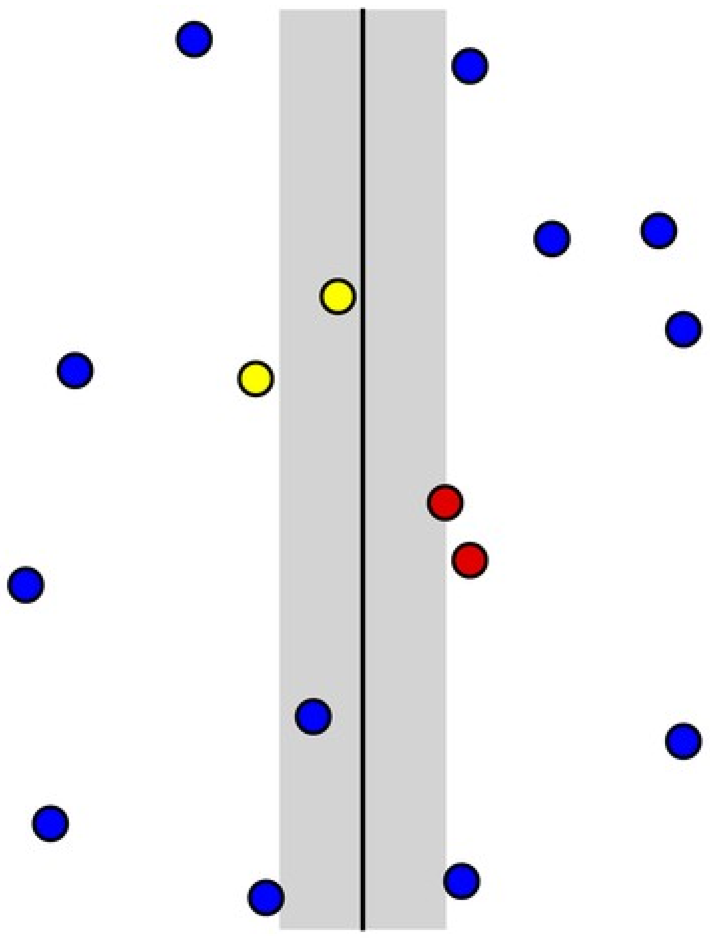
- Considere uma função f:[0..k-1]→[0..m-1] amostrada uniformemente dentre o espaço de m^k funções
- A probabilidade de $f(x)$=$f(y)$ para $x \neq y$ é 1/m
- Infelizmente, tal função f pode ser difícil de representar e avaliar
- Felizmente, é possível escolher f dentre um subespaço bem menor (apenas O(k^2) funções), satisfazendo a propriedade acima
- Tal função f pode ser avaliada em tempo O(1)
- Mais detalhes na apostila do curso

Par de Pontos Mais Próximos



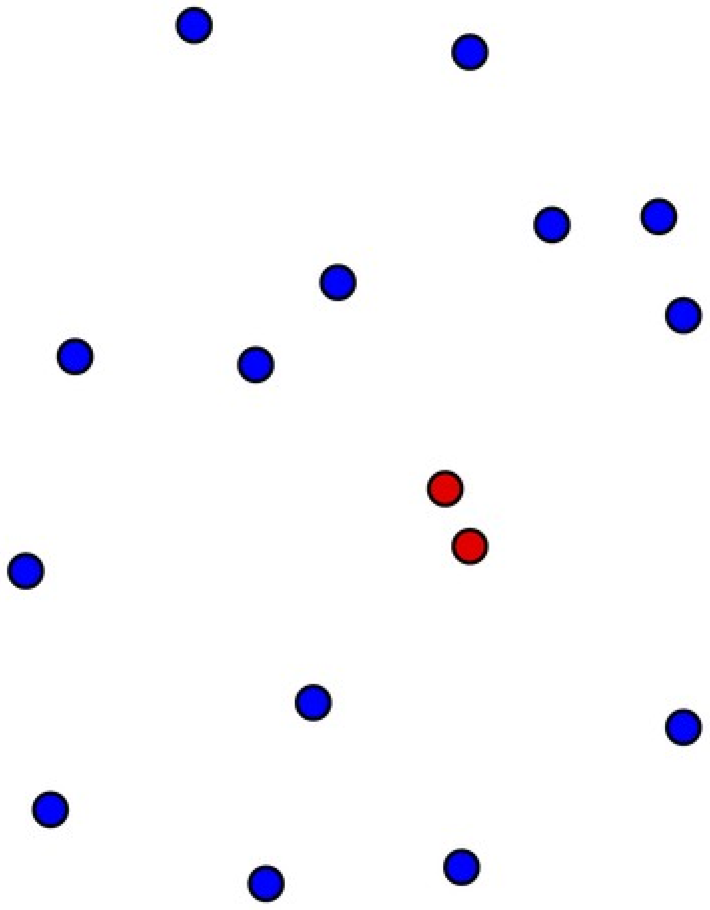
- Dado um conjunto de n pontos (em um espaço Euclideano d-dimensional), determinar o par de pontos mais próximos
- Algoritmo ingênuo leva tempo O(n^2), independente da dimensão (constante)
- Será possível fazer melhor?

História do Problema



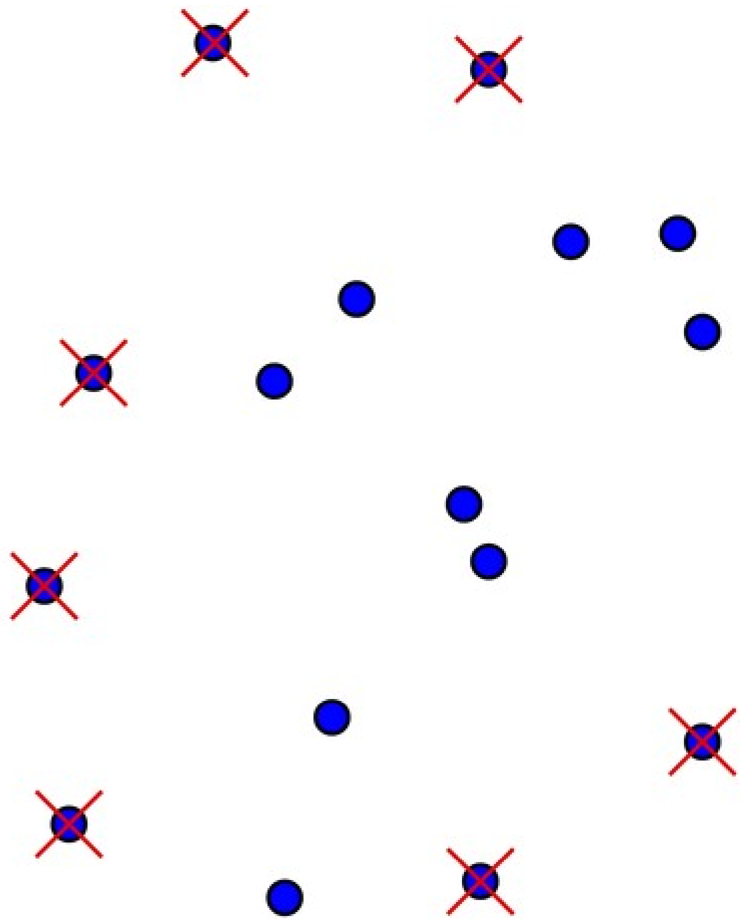
- Diversos algoritmos determinísticos com tempo O(n log n) são conhecidos
- Limite inferior de Ω(n log n) usando árvores de decisão algébrica
- Algoritmo randomizado com tempo O(n)
- Algoritmo determinístico com espaço ilimitado e tempo O(n log log n)
- Função piso é usada!

Sobre o Algoritmo



- Utliza randomização de duas maneiras:
  - Função hash
  - Amostragem aleatória
- Obtém primeiro uma solução aproximada para depois converter na solução exata
- Utiliza técnicas de dizimar e quadriculados

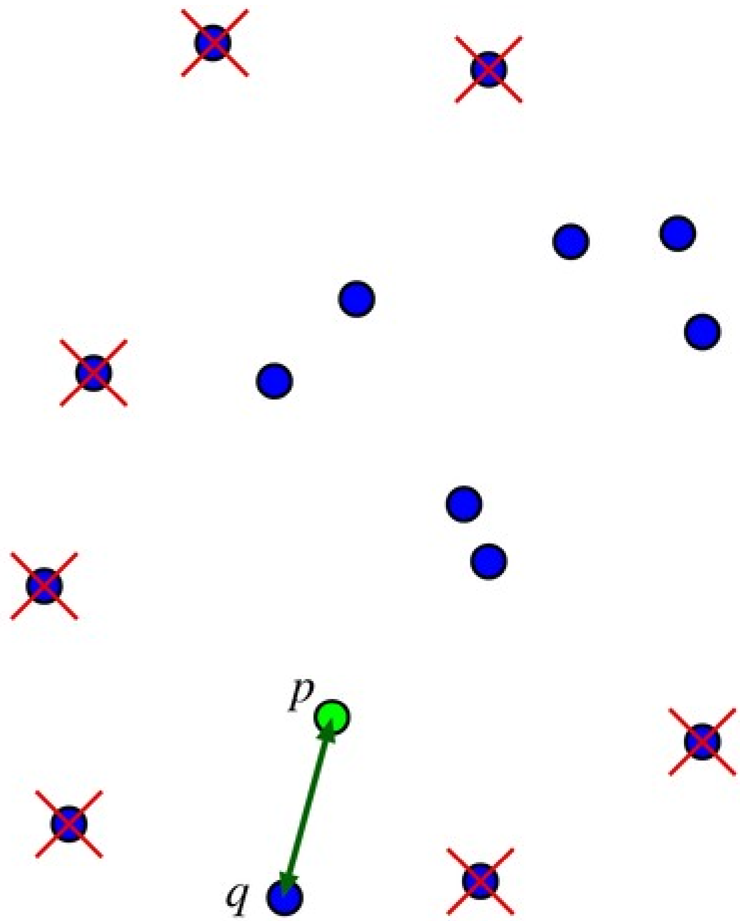
Dizimando



- Desejamos, em tempo O(n), eliminar *metade* dos pontos da entrada, sem alterar o par mais próximo
- Deste modo, executamos o algoritmo até restarem apenas 2 pontos em tempo

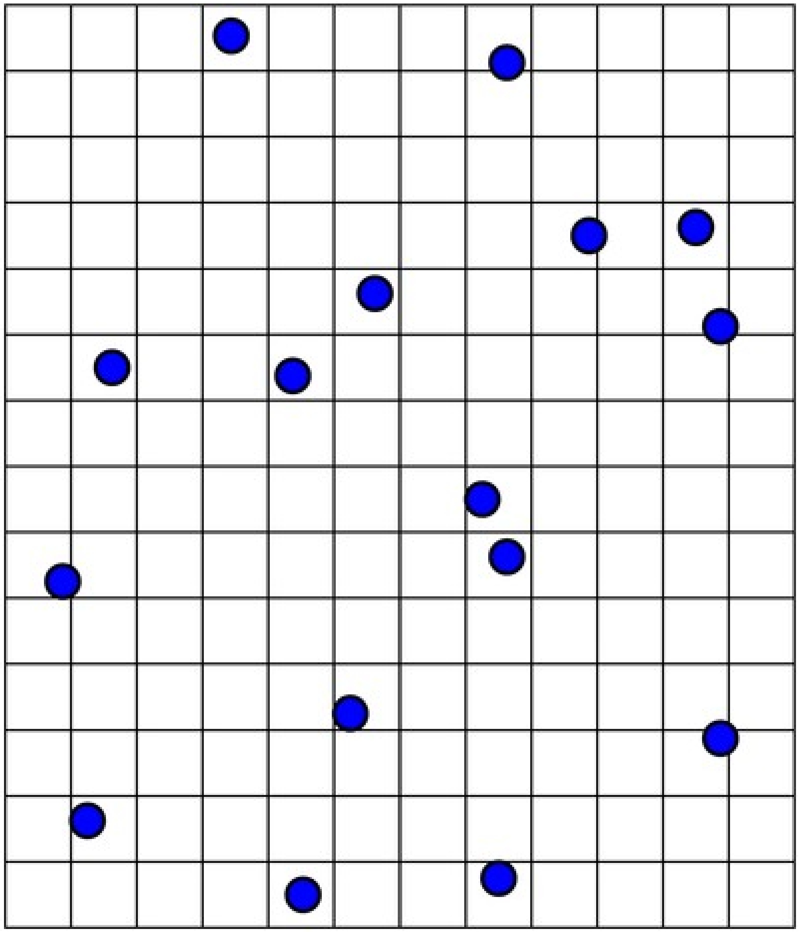
$$n + \frac{n}{2} + \frac{n}{4} + \ldots \leq 2n = O(n)$$

Dizimando



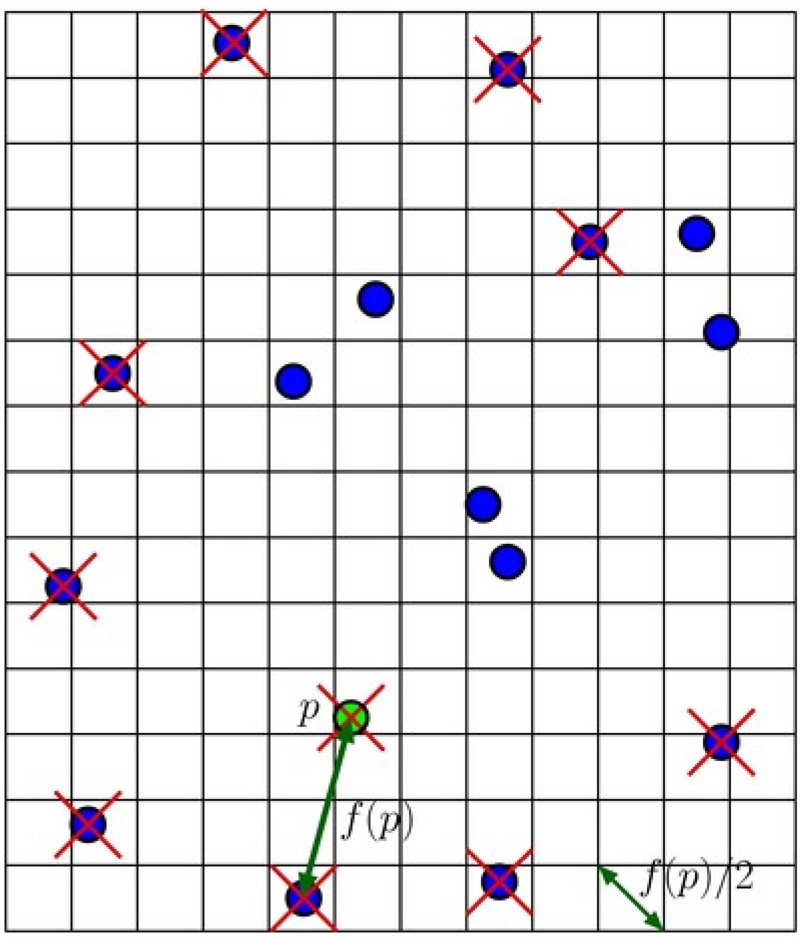
- Escolhemos um ponto p aleatório
- Determinamos q, o vizinho mais próximo de p
- Removemos os pontos cujos vizinhos mais próximos distam mais que $|pq|$
- Em média, metade dos pontos são eliminados

Quadriculados e Hash



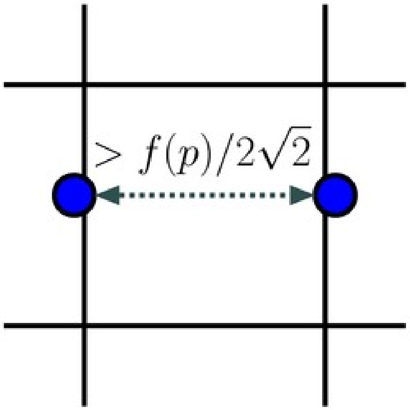
- Usando funções hash, podemos distribuir n pontos em um quadriculado com k células, tal que
- O espaço seja O(n)
- Determinar o número de pontos numa célula leve tempo O(1)
- Determinar a lista de pontos numa célula leve tempo O(1)

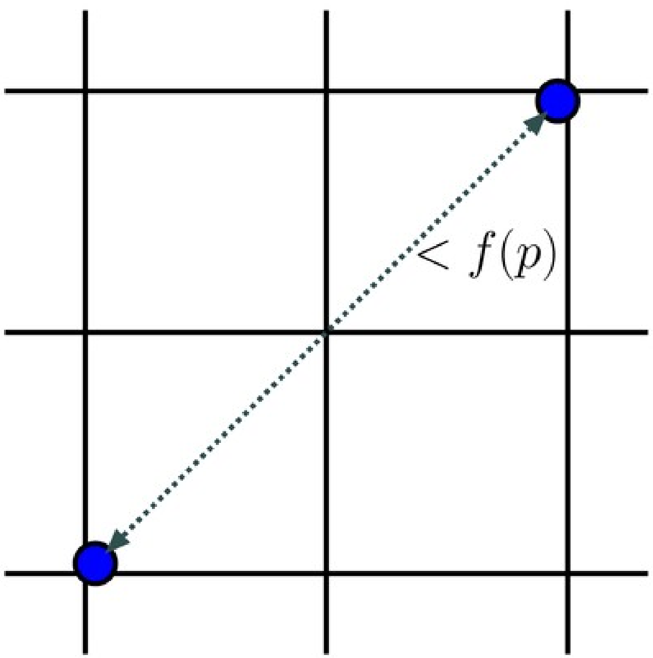
Dizimando Com Quadriculados



- Seja $f(p)$ a distância entre um ponto aleatório p e seu vizinho mais próximo
- Criamos um quadriculado com células de diâmetro $f(p)/2$
- Removemos pontos que estão sozinhos com relação as 9 células adjacentes
- O que se pode dizer sobre os pontos removidos?

Pontos Removidos





- Se um ponto q é removido, então
  $f(q) > f(p)/2\sqrt{2}$
- Se um ponto q **não** é removido, então
  $f(q) < f(p)$
- Remove alguns pontos que não deviam ser removidos
- Remove todos os pontos que deviam ser removidos

Pseudo-Código

<u>Entrada</u>:

P: Conjunto de n pontos.

<u>Saída</u>:

p, p': Par de pontos mais próximos.

<u>Observações</u>:

$vmp(p)$: Vizinho mais próximo de p.

pontosMaisPróximos(P):
- enquanto $P \neq \varnothing$:
  - $p \leftarrow$ ponto aleatório de P
  - $p' \leftarrow vmp(p)$
  - $P \leftarrow$ removerPontos(P,$|p - p'|$)
- retorne p, p'

<u>Observações</u>:

$c(p)$: Célula do ponto p.

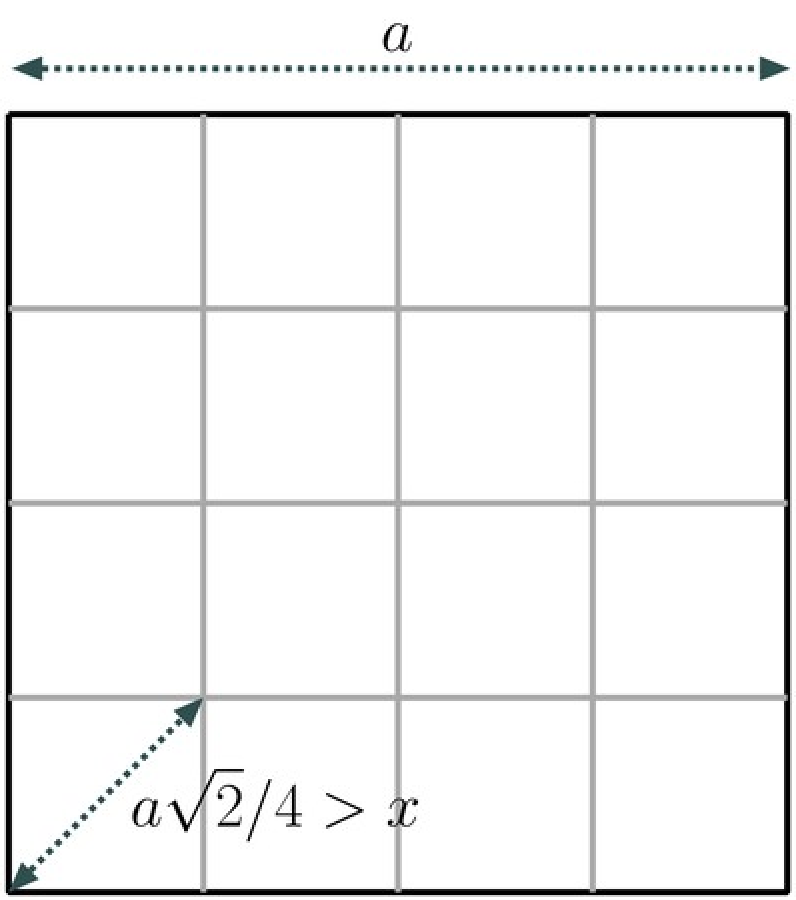
v: dicionário usando função hash.

removerPontos(P,fp):
- $P' \leftarrow \{\}$
- $v \leftarrow \{\varnothing, ...\}$
- para cada $q \in P$:
  - $v[c(q)] \leftarrow v[c(q)] \cup \{q\}$
- para cada $q \in P$:
  - $remover \leftarrow 1$
  - para cada célula x na adjacência de $c(q)$:
    - se $x = c(q)$ e $|v[x]| \geq 2$:
      - $remover \leftarrow 0$
    - se $x \neq c(q)$ e $|v[x]| \geq 1$:
      - $remover \leftarrow 0$
  - se $remover \neq 1$:
    - $P' \leftarrow P' \cup \{q\}$
- retorne P'

Tempo de Execução

- O algoritmo encontra uma aproximação de fator $2\sqrt{2}$ do par de pontos mais próximos
- A cada passo, em média, metade dos pontos são eliminados
- O algoritmo termina quando todos os pontos forem eliminados
- Usando linearidade do valor esperado, a soma do tempo em todos os passos é O(n)
- Porém, desejamos uma solução exata!

Aproximado → Exato



- Seja a a distância aproximada e x distância exata, então: $a/2\sqrt{2} < x < a$
- Criamos quadriculado com células de lado a
- O par mais próximo está em células adjacentes
- Cada célula contém no máximo 16 pontos
- Em tempo O(n) se determina o par de pontos mais próximos (exato)