Geometria Computacional

Sweep line: algoritmos

Prof. Edson Alves

2019

Faculdade UnB Gama

Sumário

Par de pontos mais próximo

Par de pontos mais próximo

- Dado um conjunto S de N de pontos no plano bidimensional, o problema de encontrar o par de pontos mais próximo consiste em encontrar dois pontos $P, Q \in S$ tal que

$$\mathrm{dist}(P, Q) = \min\{\mathrm{dist}(P_i, P_j)\}, \ \ \forall P_i \in S \quad \text{com} \quad i \neq j$$

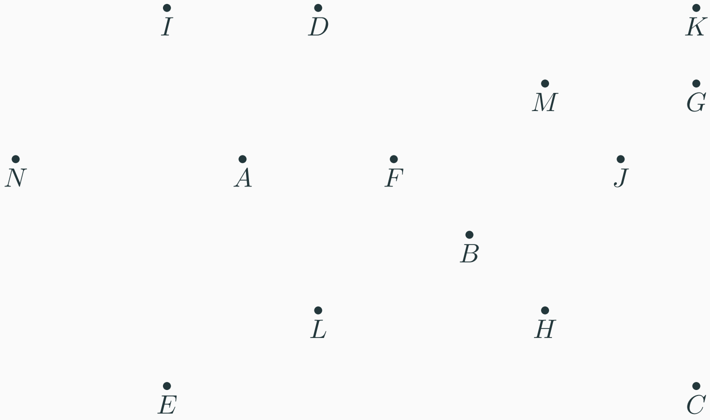
- Uma abordagem de busca completa consiste em computar as distância entre todos os pares de pontos possível, tendo complexidade $O(N^2)$
- Contudo, o problema pode ser resolvido em $O(N \log N)$ através do *sweep line*
- Os pontos deve ser ordenados em ordem lexicográfica

Par de pontos mais próximo

- Seja $d = \mathrm{dist}(P_1, P_2)$
- Agora, para todos pontos $P_3, P_4, \ldots, P_N$, deve-se computar todos os pontos vizinhos de $P_i = (x, y)$ tais que as coordenadas x estejam no intervalo $[x - d, x]$ e que as coordenadas y estejam no intervalo $[y - d, y + d]$
- Estes pontos podem ser identificados mantendo-se um conjunto de pontos cujas coordenadas estejam entre $[x - d, x]$, ordenado em ordem crescente de coordenada y
- Caso a distância de P_i para algum destes pontos seja inferior a d, o valor de d é atualizado e a varredura continua com este novo valor
- O ponto principal é que existem, no máximo, $O(1)$ pontos neste retângulo, o que faz com que a complexidade do algoritmo seja $O(N \log N)$, por conta da ordenação

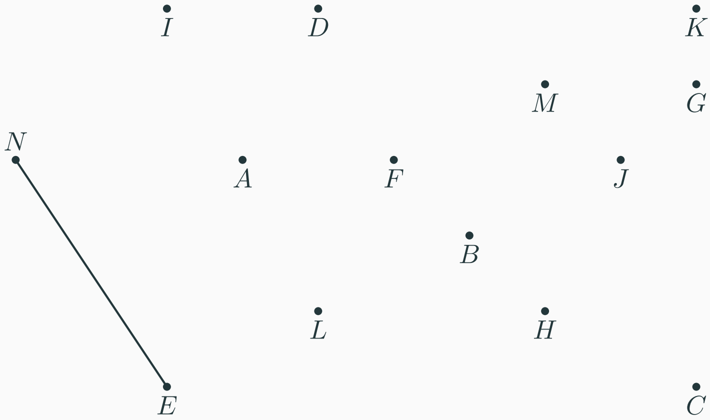
Visualização de identificação do par de pontos mais próximo

Par mais próximo: –



Visualização de identificação do par de pontos mais próximo

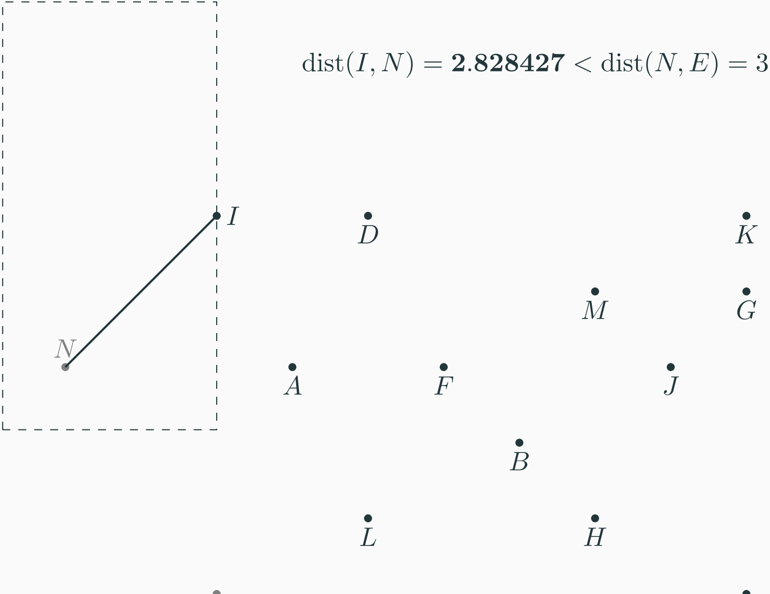
Par inicial, $\mathrm{dist}(N, E) = \mathbf{3.605551}$



Visualização de identificação do par de pontos mais próximo

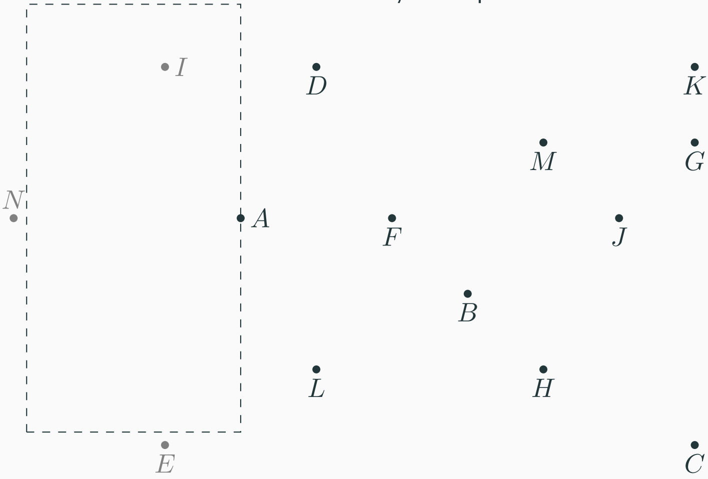
Visualização de identificação do par de pontos mais próximo

$$\mathrm{dist}(I, N) = \mathbf{2.828427} < \mathrm{dist}(N, E) = 3.605551$$



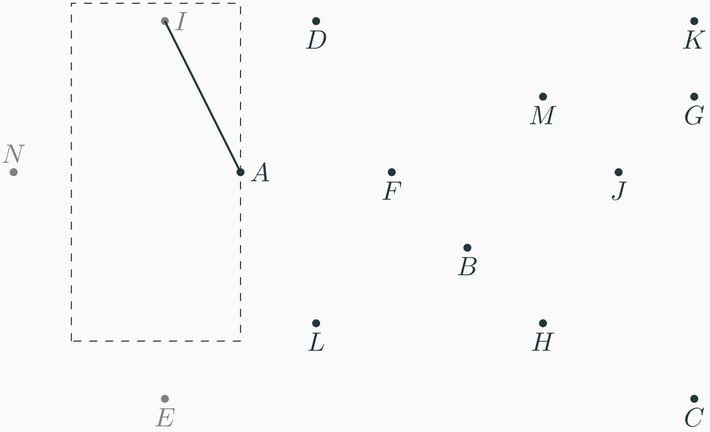
Visualização de identificação do par de pontos mais próximo

Avaliação do ponto A



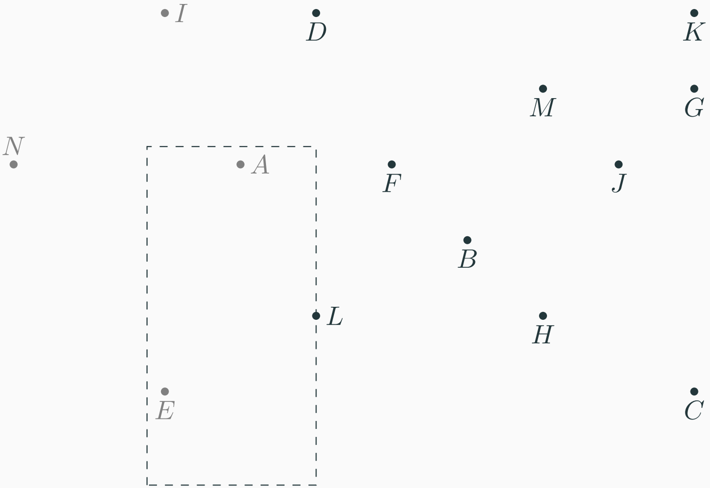
Visualização de identificação do par de pontos mais próximo

$$\mathrm{dist}(A, I) = \mathbf{2.236068} < \mathrm{dist}(I, N) = 2.828427$$



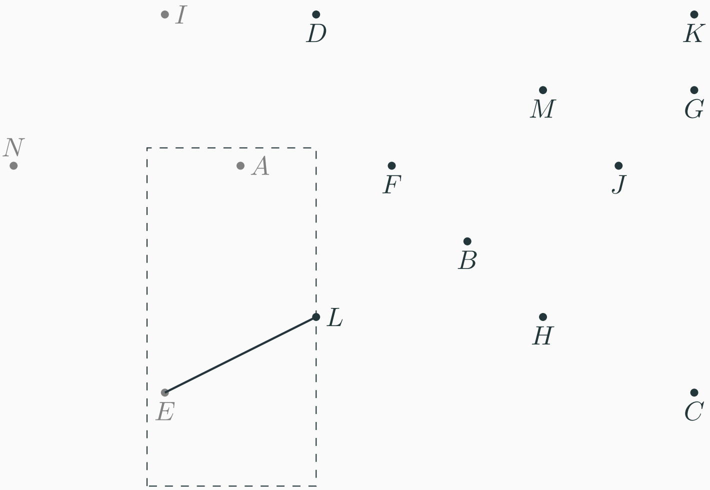
Visualização de identificação do par de pontos mais próximo

Avaliação do ponto L



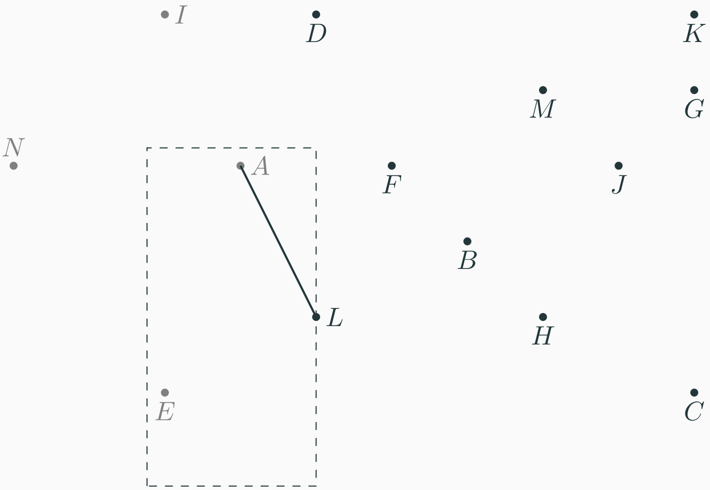
Visualização de identificação do par de pontos mais próximo

$$\mathrm{dist}(L, E) = \mathrm{dist}(A, I) = \mathbf{2.236068}$$



Visualização de identificação do par de pontos mais próximo

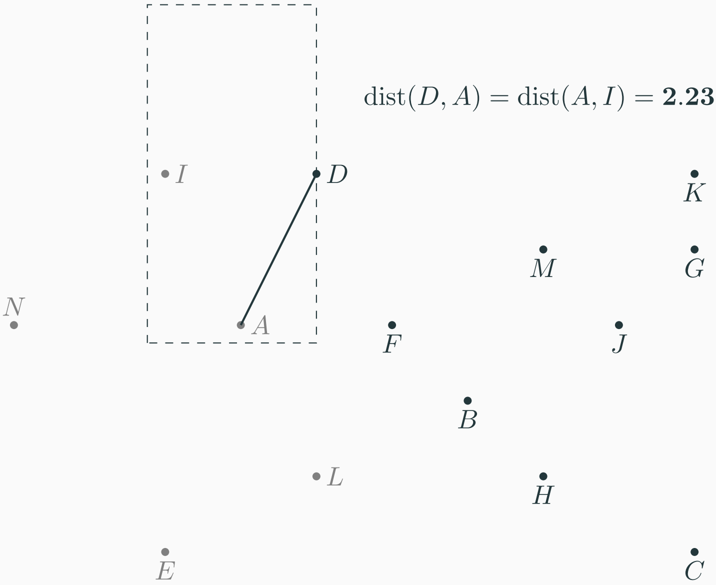
$$\mathrm{dist}(L, A) = \mathrm{dist}(A, I) = \mathbf{2.236068}$$



Visualização de identificação do par de pontos mais próximo

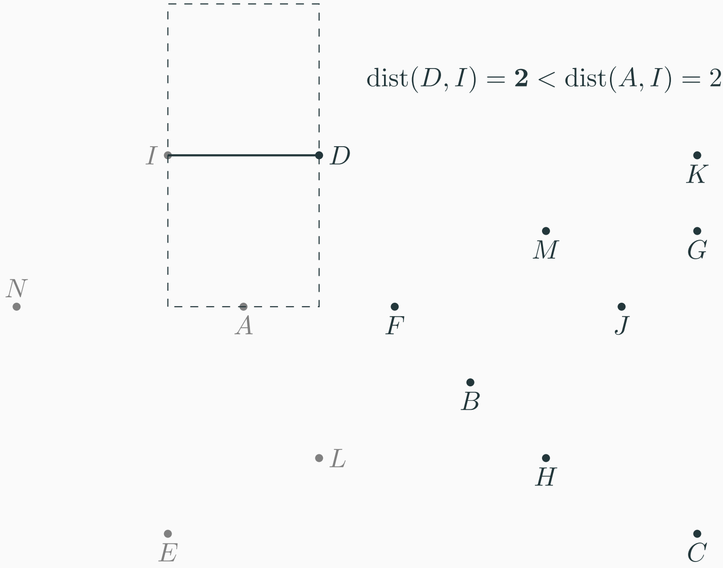
Visualização de identificação do par de pontos mais próximo

$$\mathrm{dist}(D, A) = \mathrm{dist}(A, I) = \mathbf{2.236068}$$



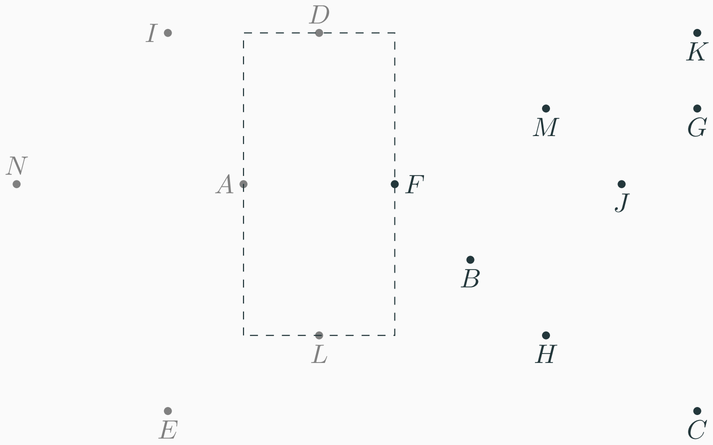
Visualização de identificação do par de pontos mais próximo

$$\text{dist}(D, I) = \mathbf{2} < \text{dist}(A, I) = 2.236068$$



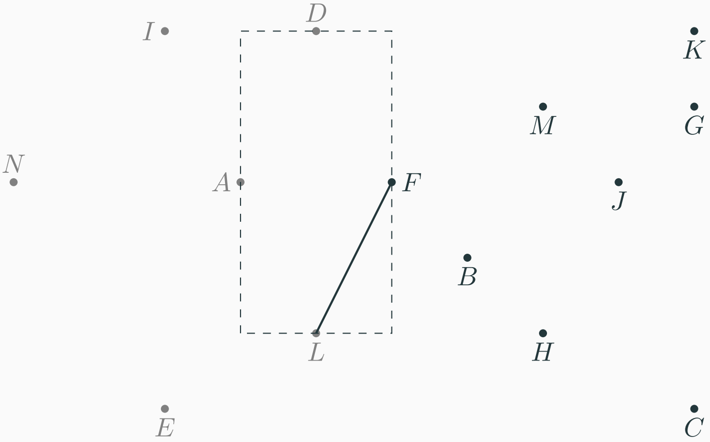
Visualização de identificação do par de pontos mais próximo

Avaliação do ponto F



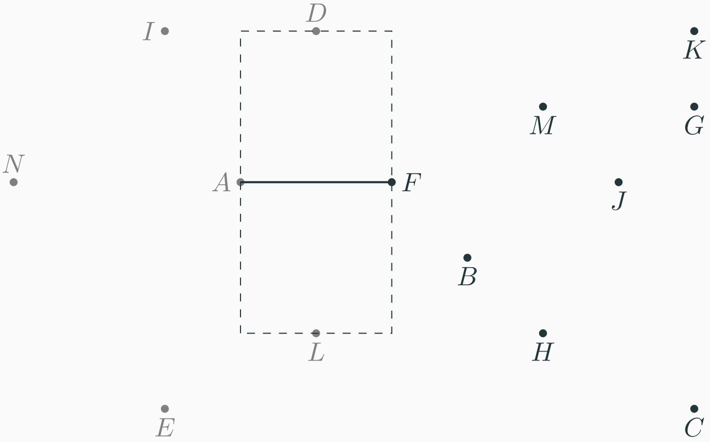
Visualização de identificação do par de pontos mais próximo

$$\text{dist}(F, L) = 2.236068 > \text{dist}(D, I) = \mathbf{2}$$



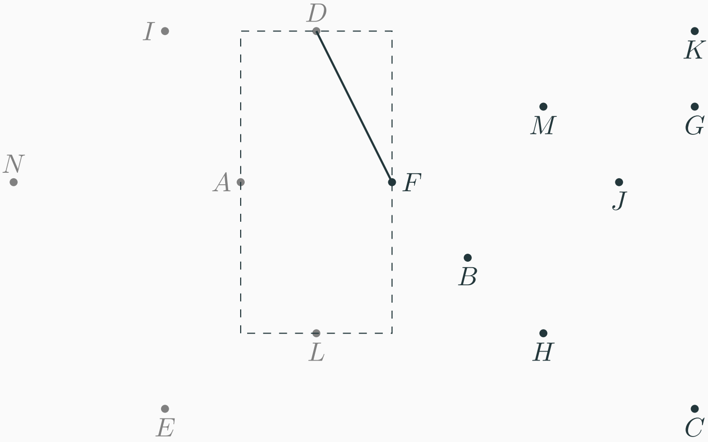
Visualização de identificação do par de pontos mais próximo

$$\text{dist}(F, A) = \text{dist}(D, I) = \mathbf{2}$$



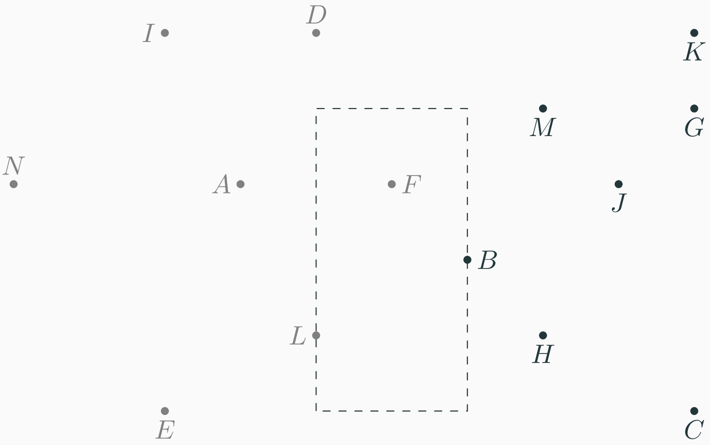
Visualização de identificação do par de pontos mais próximo

$$\text{dist}(F, D) = 2.236068 > \text{dist}(D, I) = \mathbf{2}$$



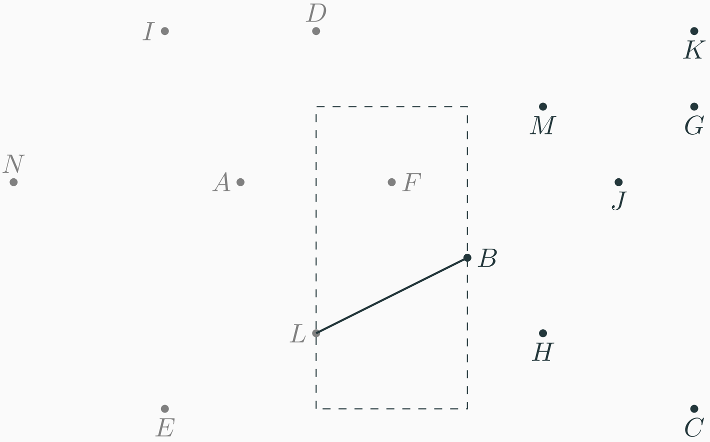
Visualização de identificação do par de pontos mais próximo

Avaliação do ponto B



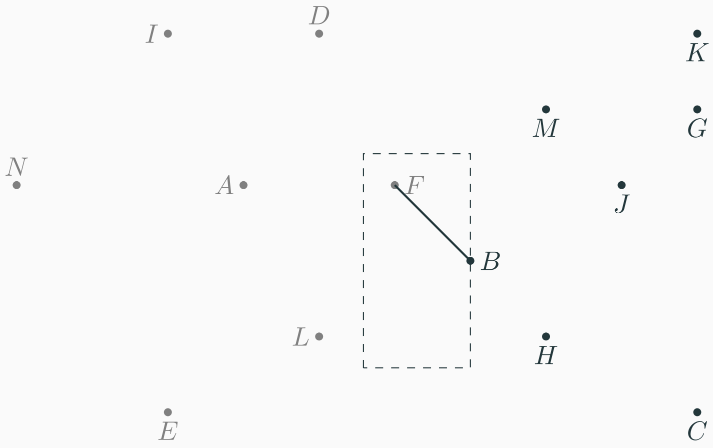
Visualização de identificação do par de pontos mais próximo

$$\mathrm{dist}(B, L) = 2.236068 > \mathrm{dist}(D, I) = \mathbf{2}$$



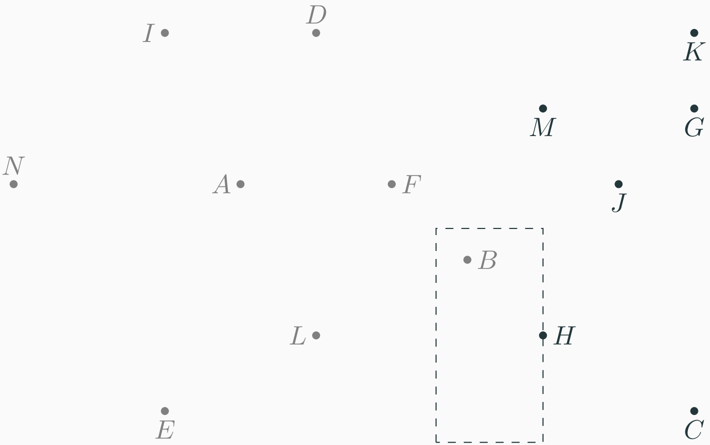
Visualização de identificação do par de pontos mais próximo

$$\mathrm{dist}(B, F) = \mathbf{1.414213} > \mathrm{dist}(D, I) = 2$$



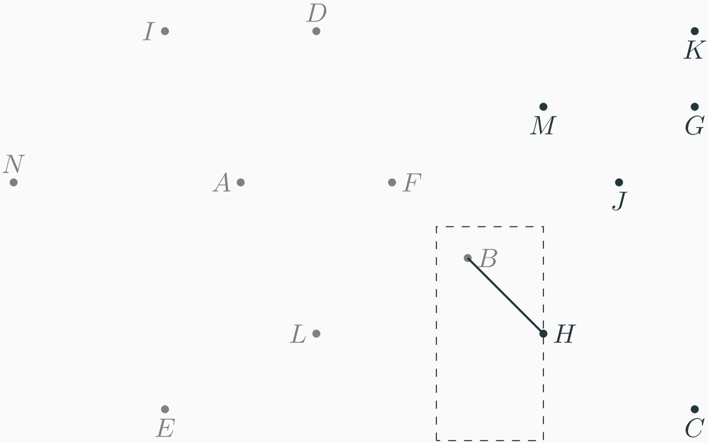
Visualização de identificação do par de pontos mais próximo

Avaliação do ponto H



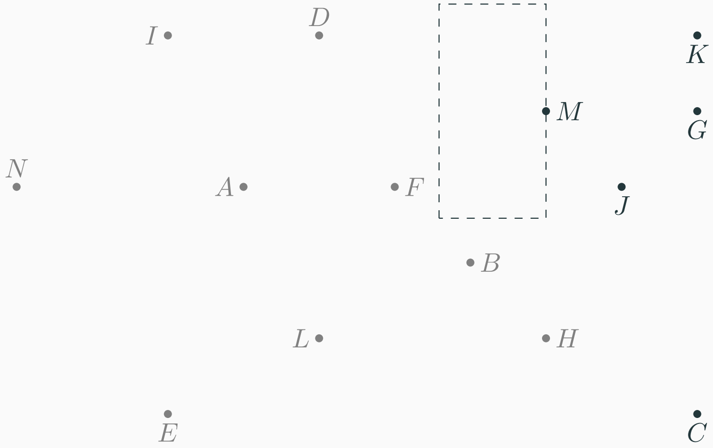
Visualização de identificação do par de pontos mais próximo

$$\operatorname{dist}(H, B) = \operatorname{dist}(B, F) = \mathbf{1.414213}$$



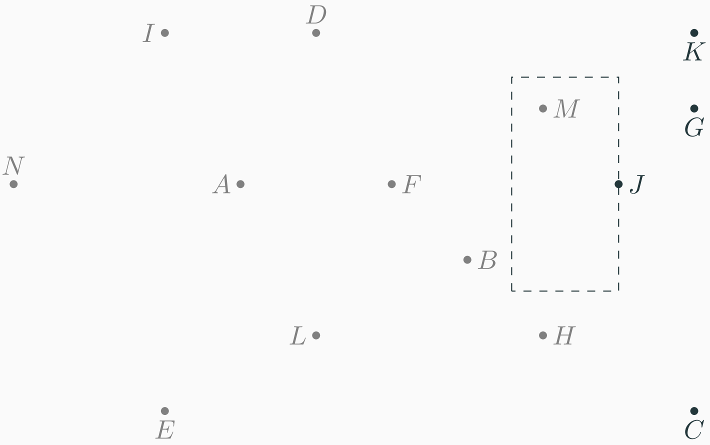
Visualização de identificação do par de pontos mais próximo

Avaliação do ponto M



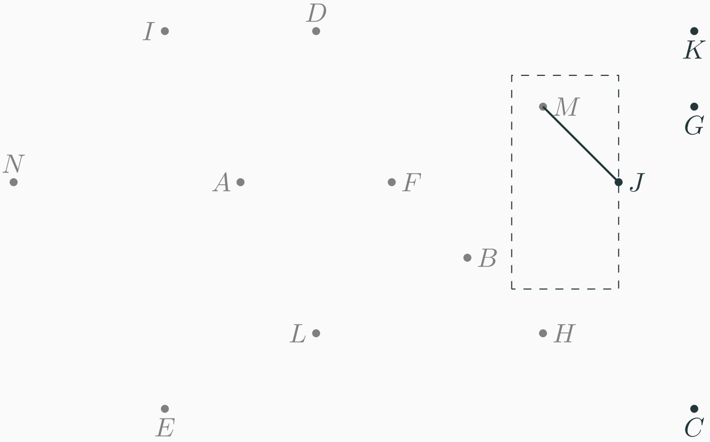
Visualização de identificação do par de pontos mais próximo

Avaliação do ponto J



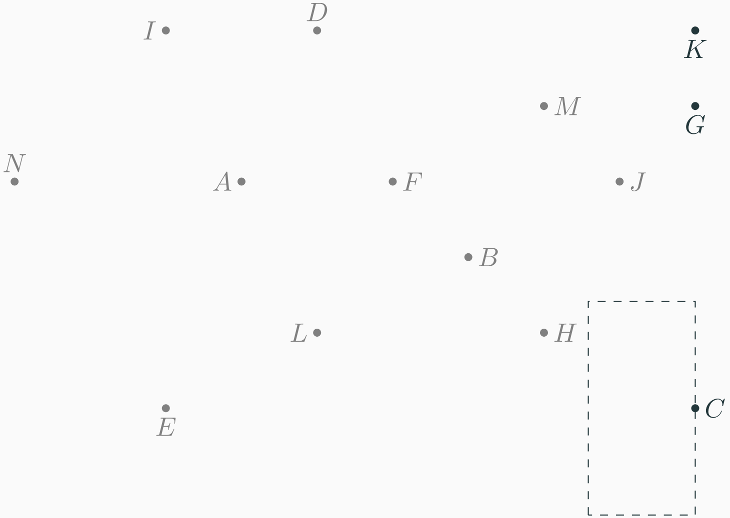
Visualização de identificação do par de pontos mais próximo

$$\text{dist}(J, M) = \text{dist}(B, F) = \mathbf{1.414213}$$



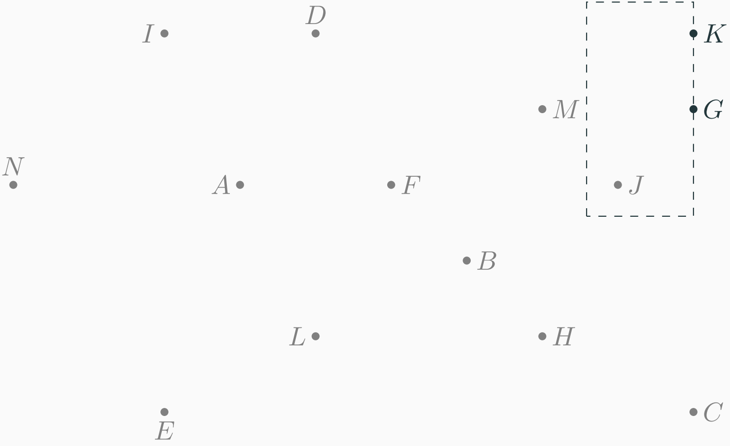
Visualização de identificação do par de pontos mais próximo

Avaliação do ponto C



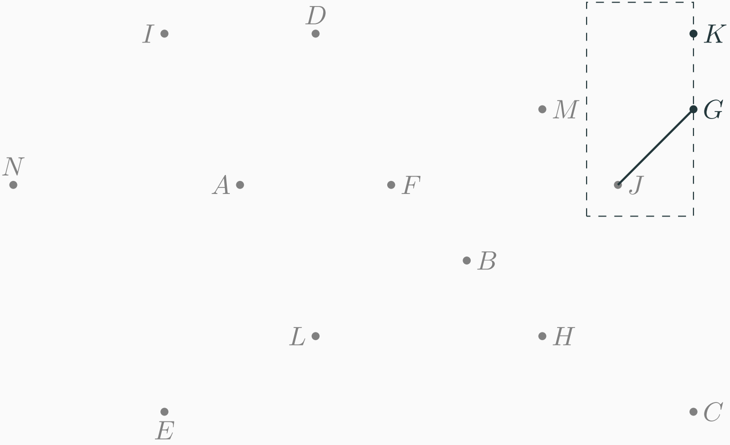
Visualização de identificação do par de pontos mais próximo

Avaliação do ponto G



Visualização de identificação do par de pontos mais próximo

$$\text{dist}(G, J) = \text{dist}(B, F) = \mathbf{1.414213}$$

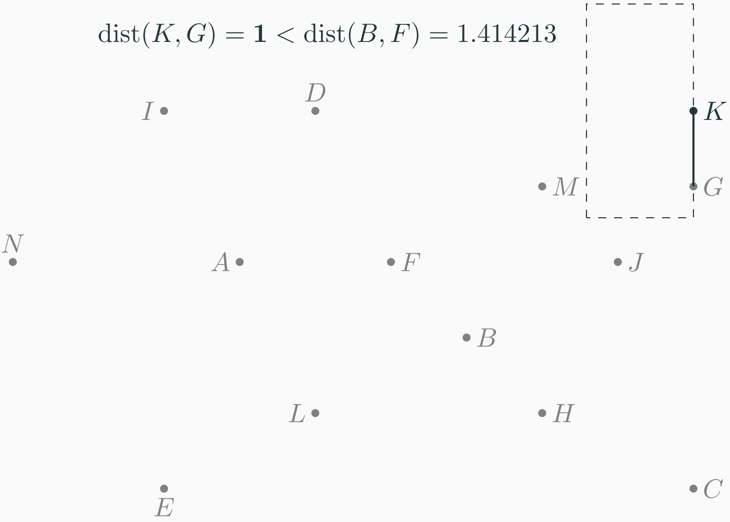


Visualização de identificação do par de pontos mais próximo

Avaliação do ponto K

Visualização de identificação do par de pontos mais próximo

$$\mathrm{dist}(K, G) = \mathbf{1} < \mathrm{dist}(B, F) = 1.414213$$



Implementação da identifacação do par mais próximo

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

using namespace std;
using ii = pair<double, double>;
using point = pair<double, double>;

#define x first
#define y second

double dist(const point& P, const point& Q)
{
    return hypot(P.x - Q.x, P.y - Q.y);
}

pair<point, point> closest_pair(int N, vector<point>& ps)
{
    sort(ps.begin(), ps.end());

    // Assume que N > 1
    auto d = dist(ps[0], ps[1]);
    auto closest = make_pair(ps[0], ps[1]);
```

Implementação da identifacação do par mais próximo

```

    set<ii> S;
    S.insert(ii(ps[0].y, ps[0].x));
    S.insert(ii(ps[1].y, ps[1].x));

    for (int i = 2; i < N; ++i)
    {
        auto P = ps[i];
        auto it = S.lower_bound(point(P.y - d, 0));

        while (it != S.end())
        {
            auto Q = point(it->second, it->first);

            if (Q.x < P.x - d)
            {
                it = S.erase(it);
                continue;
            }

```

Implementação da identifacação do par mais próximo

```
            if (Q.y > P.y + d)
                break;

            auto t = dist(P, Q);

            if (t < d)
            {
                d = t;
                closest = make_pair(P, Q);
            }

            ++it;
        }

        S.insert(ii(P.y, P.x));
    }

    return closest;
}
```

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.