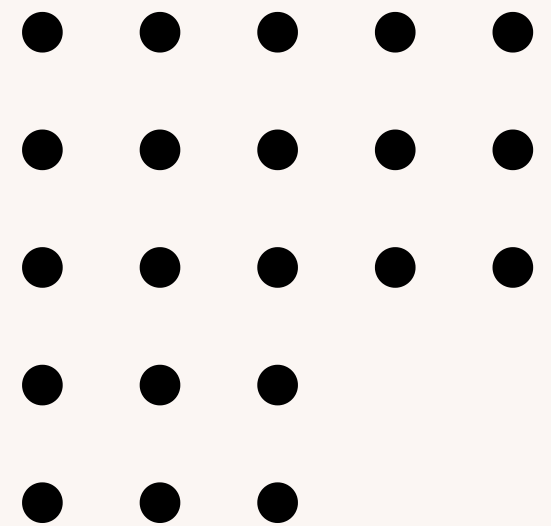
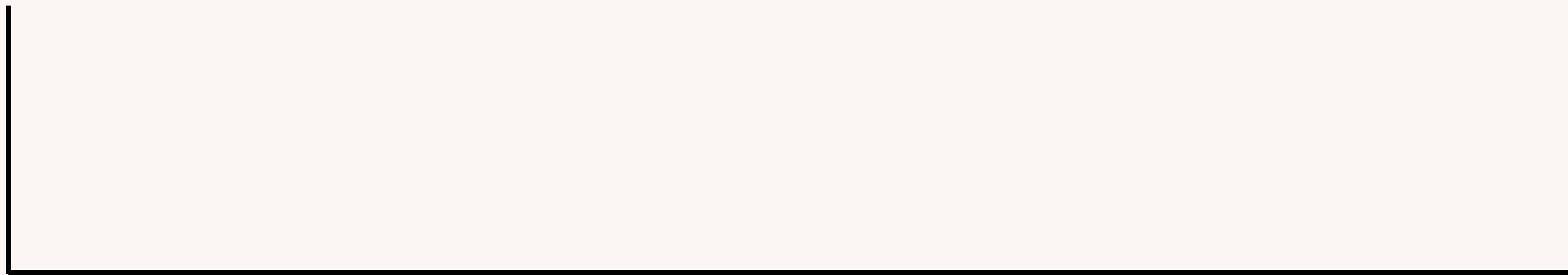
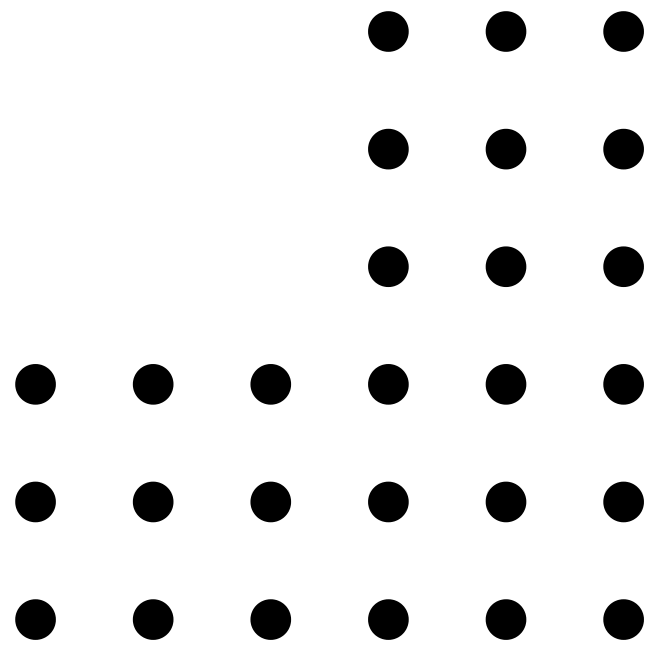


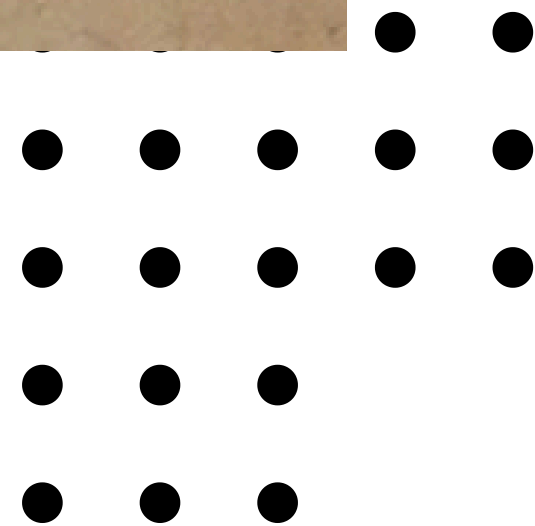


DIVIDIR PARA CONQUISTAR



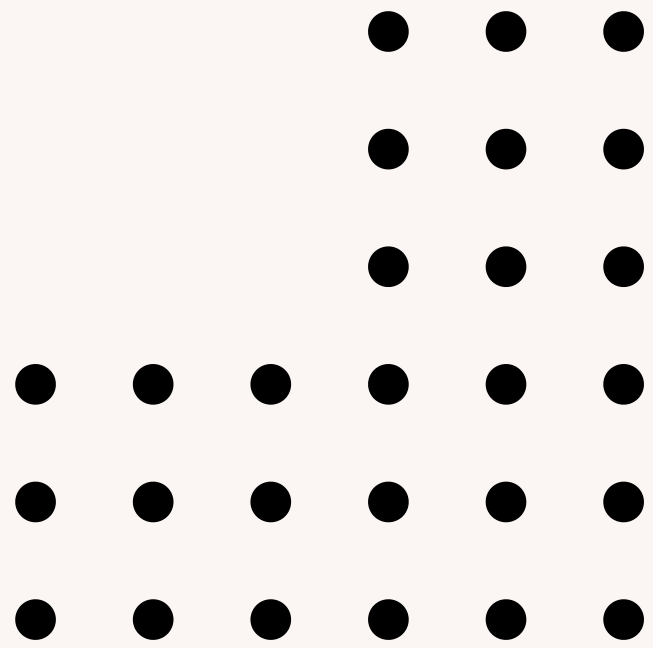


- "Dividir para Conquistar" O que é?
- Importância?

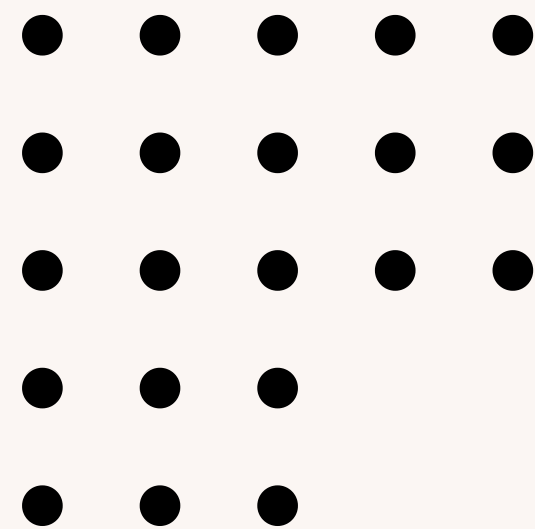
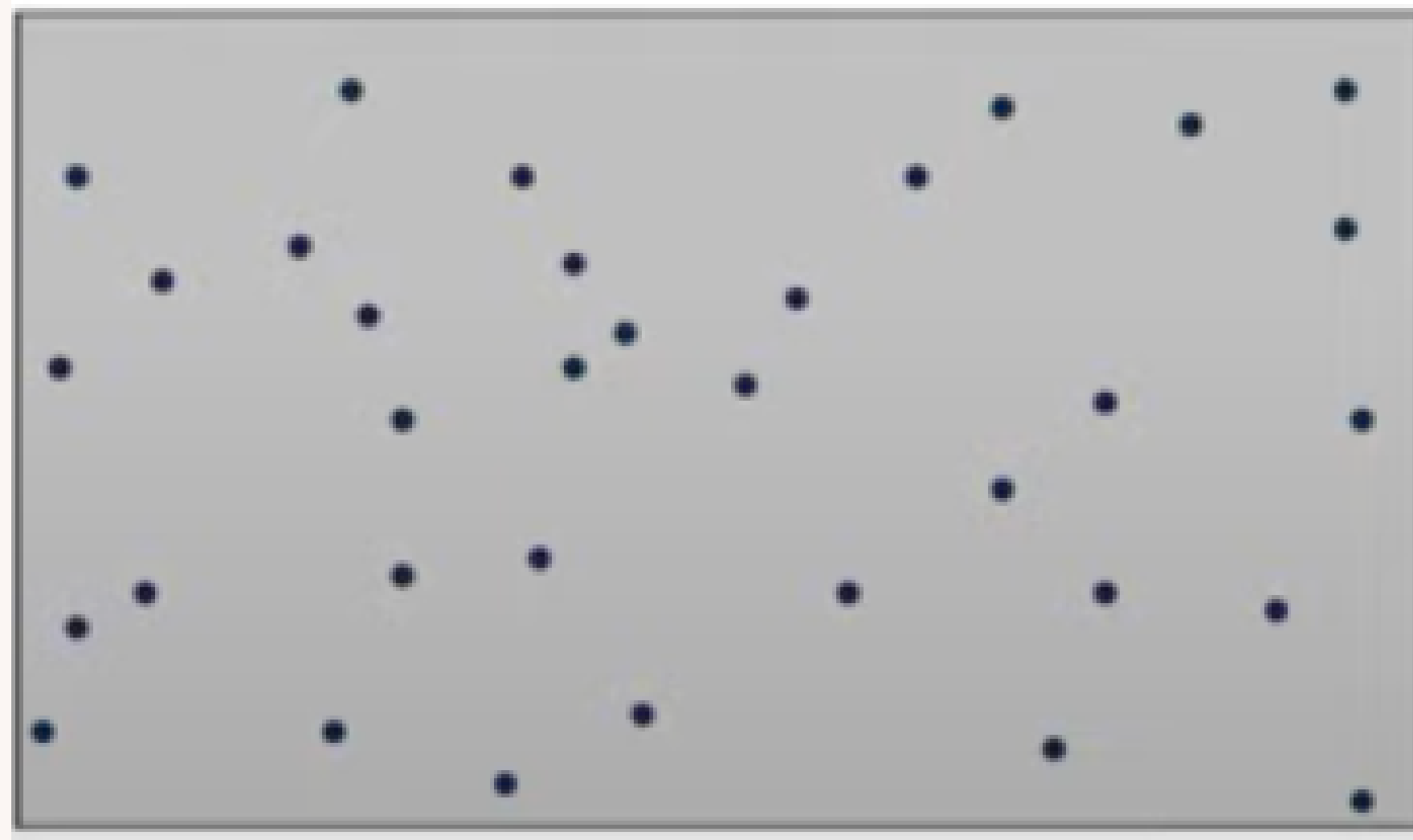


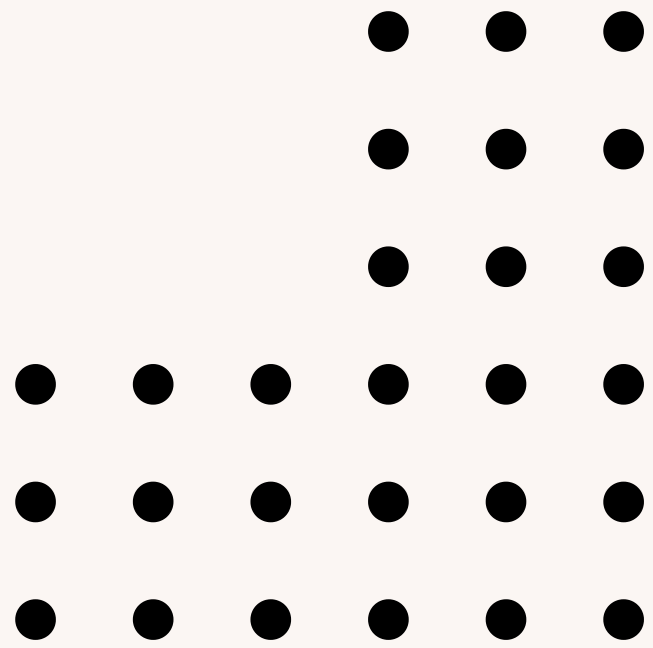
Subproblemas



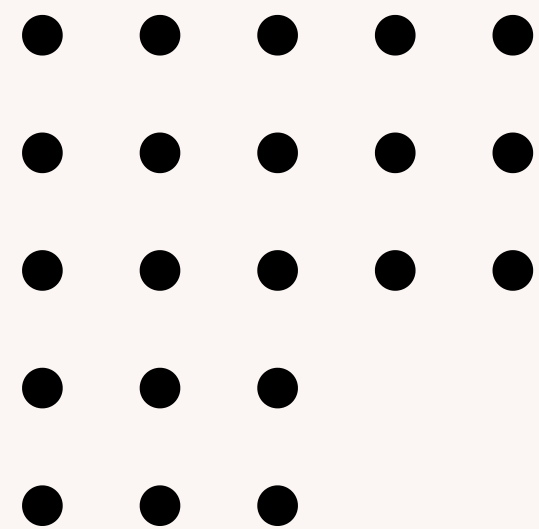


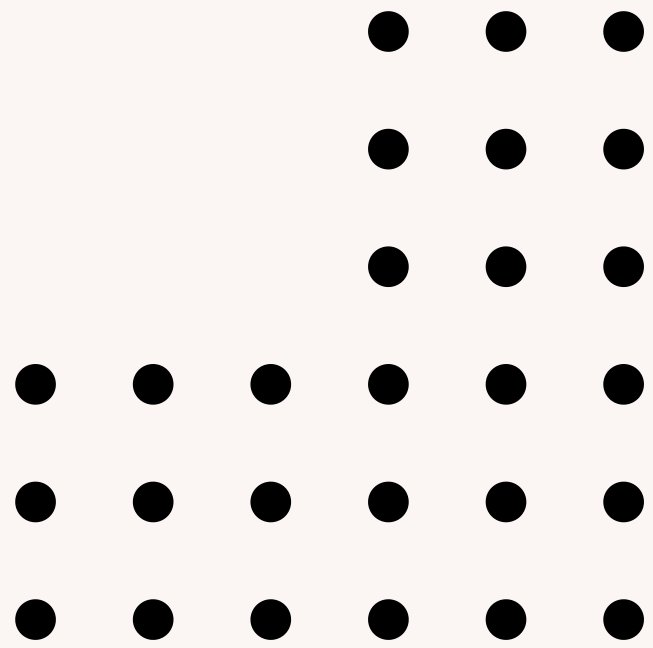
Par de Pontos Mais Próximo



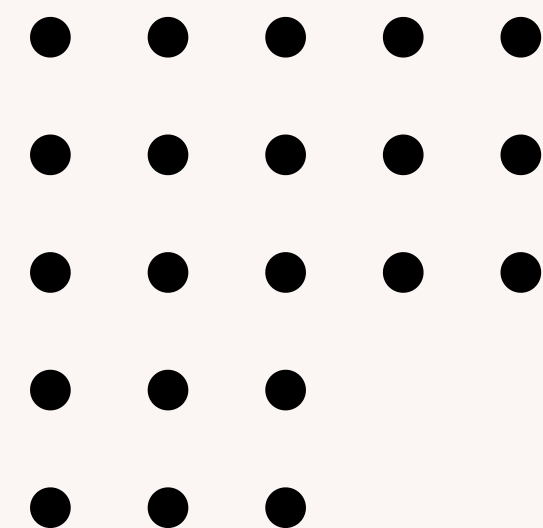
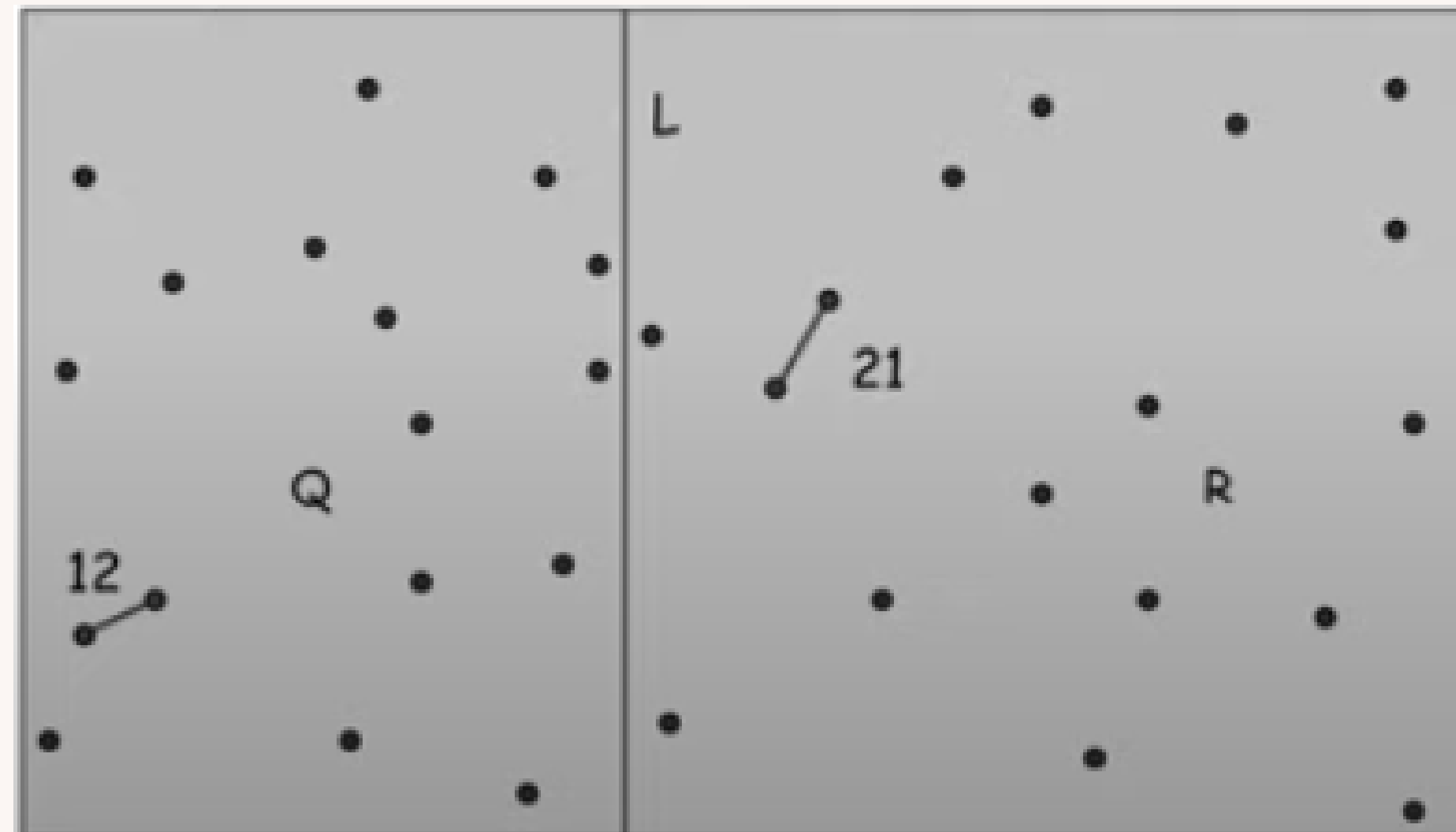


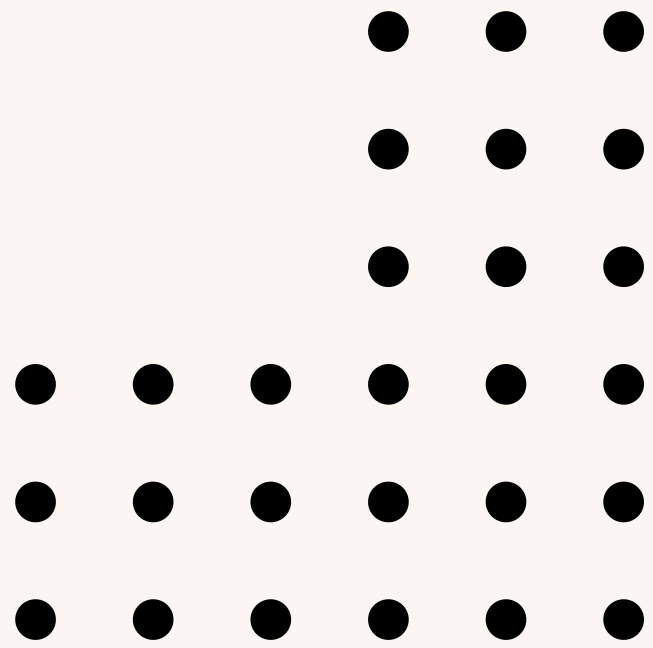
Problemas



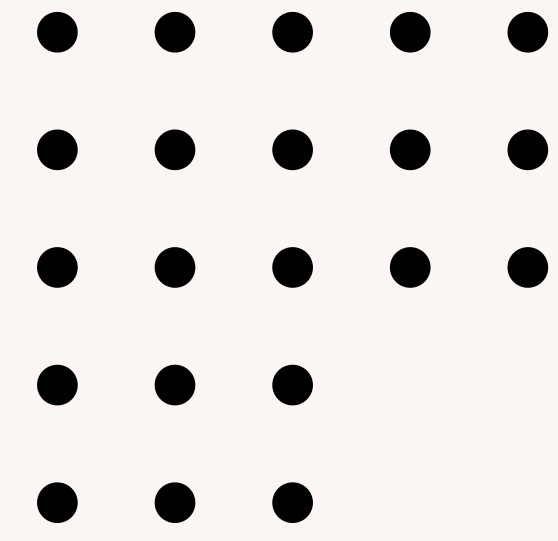
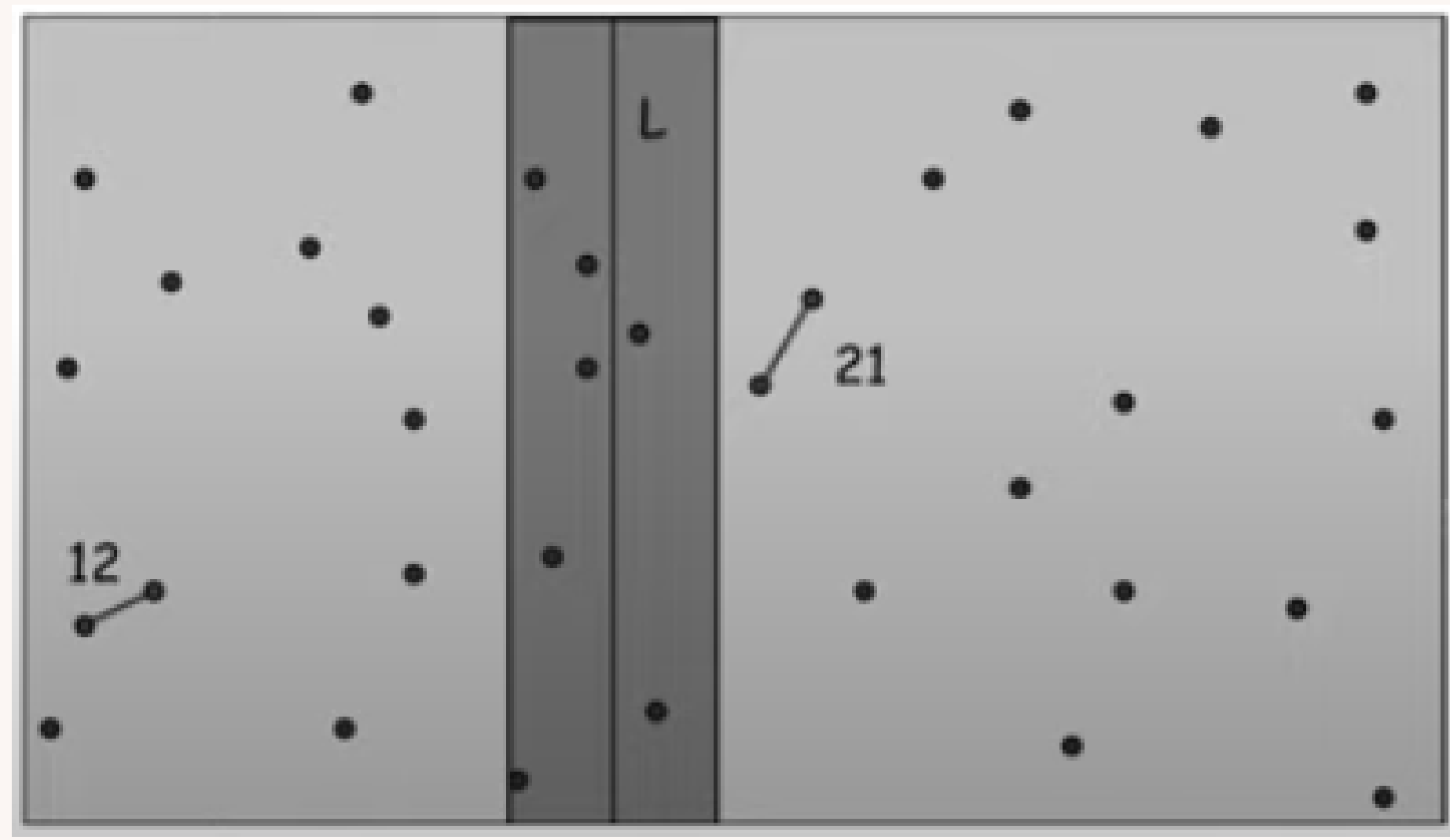


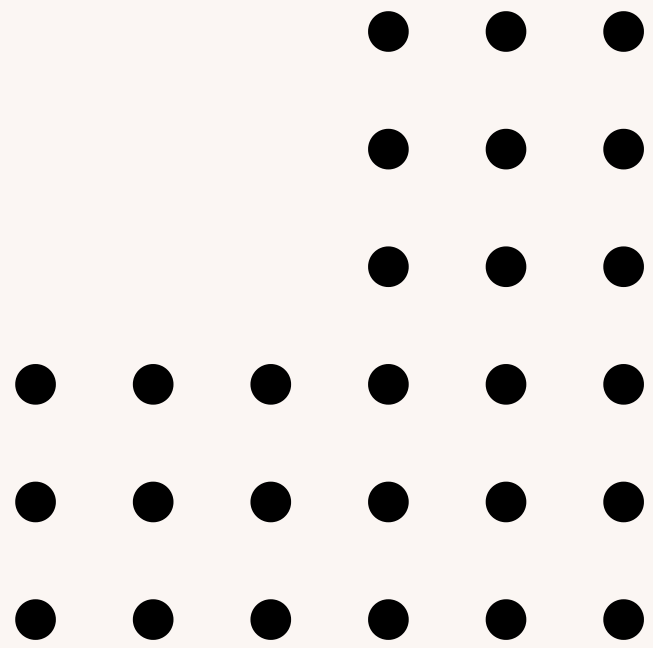
Solução



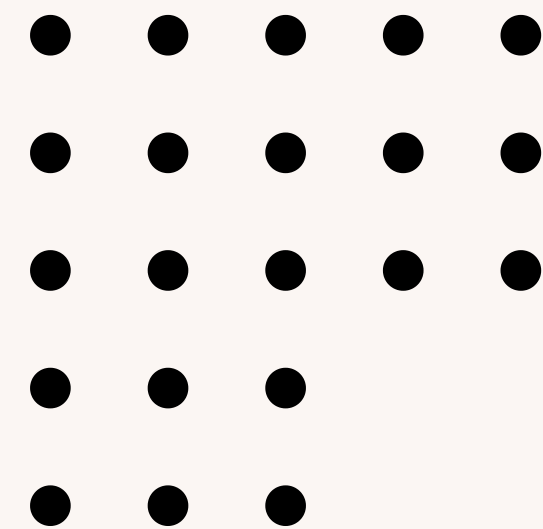
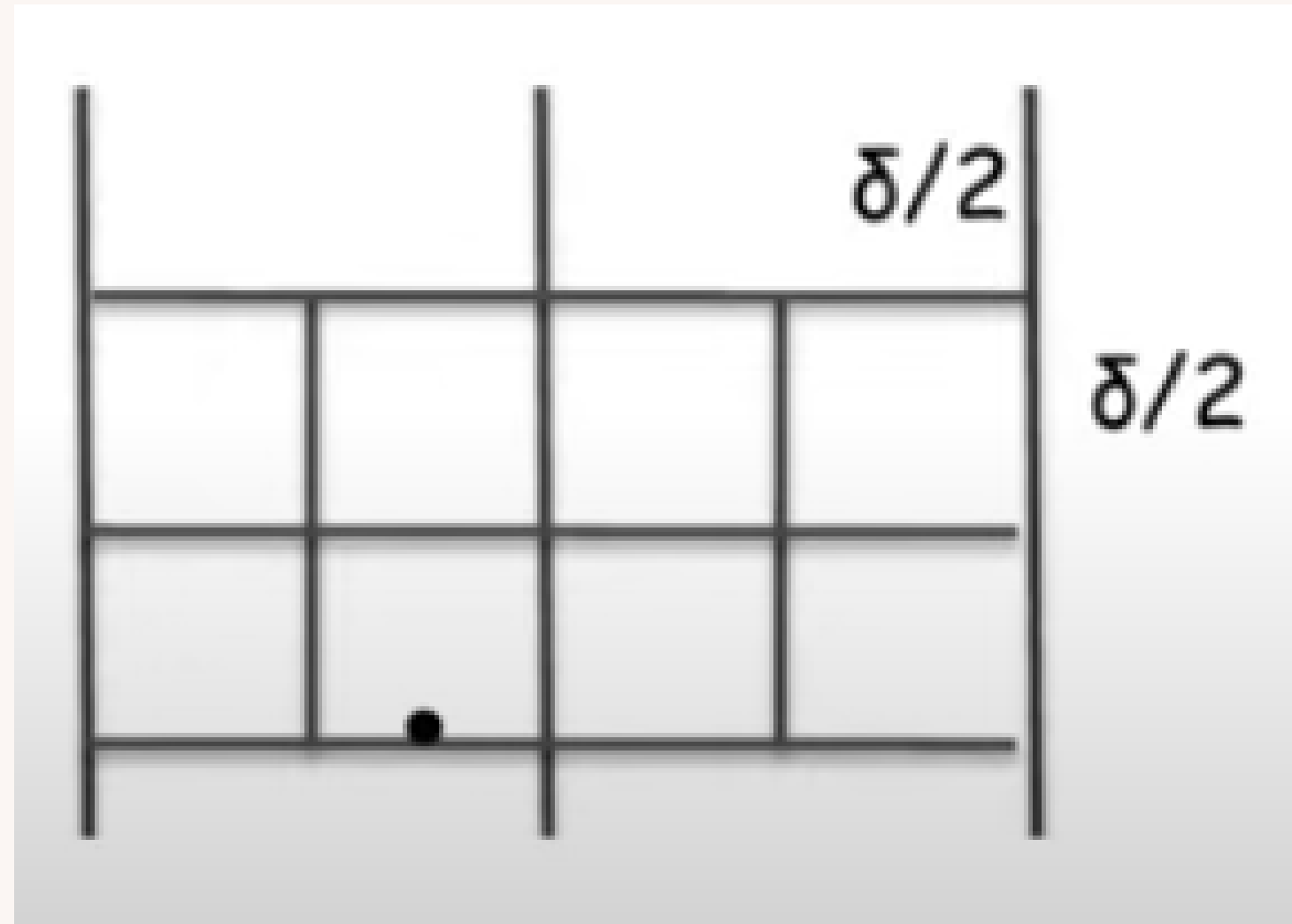


Solução





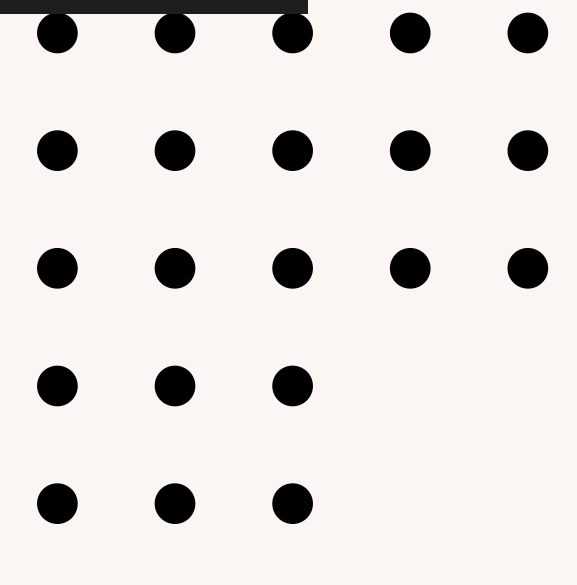
Solução





Resolução

```
typedef struct {  
    int x, y;  
} Ponto;  
  
int compararX(const void* a, const void* b) {  
    Ponto* p1 = (Ponto*)a;  
    Ponto* p2 = (Ponto*)b;  
    return (p1->x - p2->x);  
}  
  
int compararY(const void* a, const void* b) {  
    Ponto* p1 = (Ponto*)a;  
    Ponto* p2 = (Ponto*)b;  
    return (p1->y - p2->y);  
}
```



Resolução

```
double distancia(Ponto p1, Ponto p2) {  
    return sqrt((p1.x - p2.x) * (p1.x - p2.x) + (p1.y - p2.y) * (p1.y - p2.y));  
}  
  
double faixaMaisProxima(Ponto faixa[], int tamanho, double d) {  
    int i, j;  
    double min = d;  
    qsort(faixa, tamanho, sizeof(Ponto), compararY);  
    for (i = 0; i < tamanho; ++i) {  
        for (j = i + 1; j < tamanho && (faixa[j].y - faixa[i].y) < min; ++j) {  
            double dist = distancia(faixa[i], faixa[j]);  
            if (dist < min) {  
                min = dist;  
            }  
        }  
    }  
    return min;  
}
```

Resolução

```
double parMaisProximoRec(Ponto pontos[], int n) {
    int i = 0;
    int j = 0;
    if (n <= 3) {
        double minDist = DBL_MAX;
        for (i = 0; i < n; ++i) {
            for (j = i + 1; j < n; ++j) {
                double dist = distancia(pontos[i], pontos[j]);
                if (dist < minDist) {
                    minDist = dist;
                }
            }
        }
        return minDist;
    }

    int meio = n / 2;
    Ponto pontoMeio = pontos[meio];

    double distEsquerda = parMaisProximoRec(pontos, meio);
    double distDireita = parMaisProximoRec(pontos + meio, n - meio);

    double d = fmin(distEsquerda, distDireita);

    Ponto faixa[n];

    for (i = 0; i < n; i++) {
        if (abs(pontos[i].x - pontoMeio.x) < d) {
            faixa[j] = pontos[i];
            j++;
        }
    }

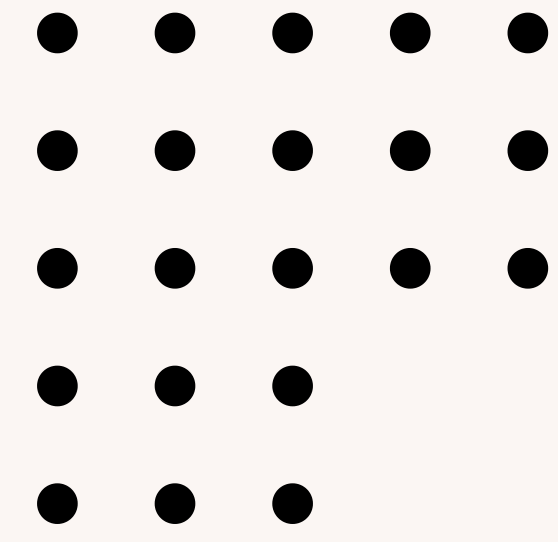
    return fmin(d, faixaMaisProxima(faixa, j, d));
}
```

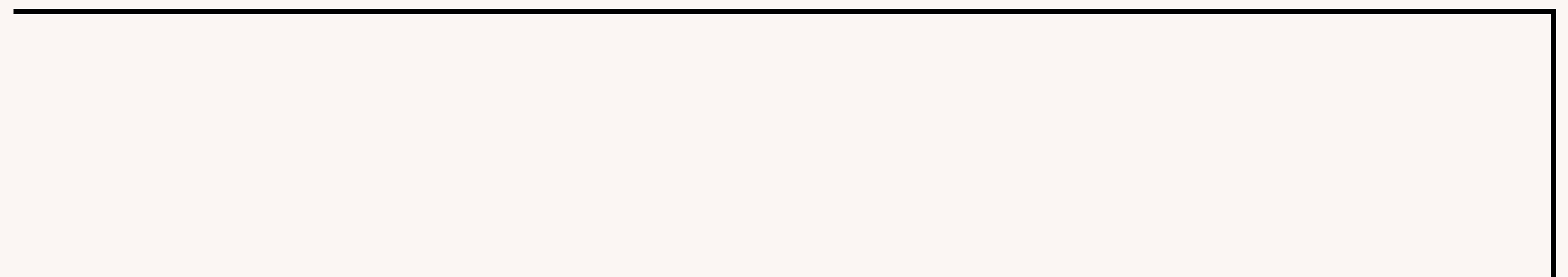
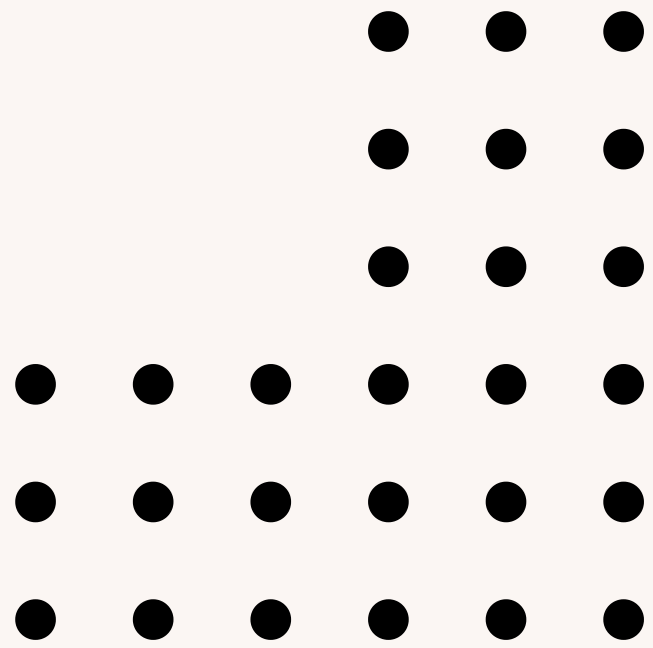


Resolução

```
double parMaisProximo(Ponto pontos[], int n) {
    qsort(pontos, n, sizeof(Ponto), compararX);
    return parMaisProximoRec(pontos, n);
}

int main() {
    Ponto pontos[] = {{2, 3}, {12, 30}, {40, 50}, {5, 1}, {12, 10}, {3, 4}};
    int n = sizeof(pontos) / sizeof(pontos[0]);
    printf("A menor distância é %f\n", parMaisProximo(pontos, n));
    return 0;
}
```





FIM

