MAC0320 - Algoritmos e Estruturas de Dados II
Problema da Conexidade 2D
Relatório

Marcos Kawakami - 8041331

1 O problema

Sejam $P_1, P_2, ...P_n$ pontos no quadrado unitário $[0,1]^2$, e d um real positivo. Dizemos que P_i e P_j são *conexos* se a distância euclidiana entre estes pontos é menor ou igual a d. Queremos determinar se o conjunto $\{P_1, P_2, ..., P_n\}$ possui uma única componente conexa.

2 Implementação

Há diversas abordagens possíveis para este problema. Uma delas seria realizar uma busca no grafo implícito e verificar se todos os pontos são alcançáveis a partir de um ponto inicial arbitrário. Entretanto, preferiu-se utilizar o algoritmo *Union-Find*, pois este permite verificar eficientemente o resultado do problema após a inserção de cada ponto, e tal propriedade será útil na resolução da próxima tarefa.
O algoritmo funciona, em alto nível, como segue:

1. Para cada ponto P_i da entrada, realizamos o seguinte procedimento:
   (a) Criamos um elemento novo na estrutura de *Union-Find*(UF). Isto aumenta em uma unidade a quantidade de componentes conexas do sistema.
   (b) Para cada ponto P_j já inserido na UF e que esteja próximo de P_i, verificamos se $\text{dist}(P_i, P_j) \leq d$. Caso positivo, fazemos a união entre as componentes de P_i e P_j.
2. Após processar cada ponto, verificamos se UF apresenta somente uma única componente conexa.

O passo 1b foi escrito de forma vaga propositalmente. Para encontrarmos "pontos próximos a P_i" usaremos a técnica de *Gridding*: Dividimos o quadrado unitário em $\lceil \frac{1}{d} \rceil \times \lceil \frac{1}{d} \rceil$ células de tamanho $d \times d$ (células que se encontram na lateral direita ou superior do quadrado unitário podem ter tamanhos menores).

Guardamos, para cada célula, uma lista dos pontos que se encontram em seu interior. Dessa forma, dado um ponto P_i numa célula C, os pontos que encontram-se a uma distância d ou inferior só podem estar em C ou nas células adjacentes a C. Isto potencialmente reduz o número de testes que realizamos a cada passo da iteração.

3 Análise do algoritmo

No pior caso todos os pontos se encontram na mesma célula, a uma distância inferior a d uns dos outros. Assim, acabamos testando todos os $O(n^2)$ pares de pontos, efetuando uma operação de união para cada par. A complexidade neste caso é $O(n^2 lg^* n)$.
No entanto, no caso em que a entrada é aleatória, podemos esperar uma distribuição mais uniforme dos pontos nas células do quadrado unitário. Assumindo que haja $O(nd^2)$ pontos em cada célula, a complexidade resultante é $O(n^2 d^4 lg^* n)$. Isto mostra que, quando menor o valor de d, mais eficiente é a técnica de *Gridding*.
No código entregue foram feitas algumas otimizações para que o algoritmo executasse em tempo razoável mesmo para entradas com valores grandes de d. A principal delas foi a interrupção da iteração do passo 1b do algoritmo no instante em que se verifica que os pontos em UF formam uma única componente conexa. Claramente não seriam mais feitas operações de união após este instante, uma vez que todos os pontos pertencem a mesma componente. A corretude do algoritmo, portanto, se mantém.
A motivação desta otimização foi o fato de que, quanto maior o valor de d, maior a probabilidade de que os pontos formem rapidamente uma única componente à medida em que são inseridos na estrutura.