

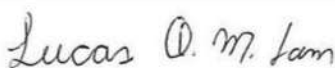
MAC0323 ALGORITMOS E ESTRUTURAS DE DADOS II

FOLHA DE SOLUÇÃO

Nome: Lucas Quaresma Medina Lam

Número USP: 11796399

Assinatura



Sua assinatura atesta a autenticidade e originalidade de seu trabalho e que você se compromete a seguir o código de ética da USP em suas atividades acadêmicas, incluindo esta atividade.

Exercício: 6

Data: 22/07/2021

SOLUÇÃO

Pergunta 1. Como parecem crescer os tempos de execução dos três programas acima conforme aumentamos k ?

Os tempos de execução dos algoritmos DistanceAcyclic e DistanceBF aparentam ser logarítmico, para os k 's testados iguais a 10, 20, 40, 80, 160 e 320.

Já os tempos de execução do algoritmo DistanceDijkstra2, aparentam ser exponencial, para os k 's testados iguais a 10, 15, 20, 25, 28 e 30

Pergunta 2. Explique, de forma convincente, por que o tempo de execução de DistanceDijkstra2.java cresce como você observou.

Sabemos que o algoritmo de Dijkstra para calcular distâncias, tem complexidade $O(E + V \cdot \log(v))$ sem considerar arestas negativas. Porém, analisando o algoritmo DistanceDijkstra2, que permite a existência de arestas negativas no grafo, vemos que esse cresce de forma exponencial. Conseguimos observar essa diferença, pois enquanto o algoritmo original passa apenas 1 vez pelos vertices os "relaxando", mantendo esse invariante até o final do algoritmo, o DistanceDijkstra2 pode passar várias e várias vezes até encontrar o resultado final. Isso ocorre do fato de que se temos arestas negativas, caminhos que antes eram os menores possíveis, passam a ter que serem recalculados, tendo de mudar todo o `edgeTo[]` e o `distTo[]`, passando novamente pelos vertices e calculando as distancias para cada par de ponto, o que acaba tornando o algoritmo muito mais lento que o original a ponto de não compensar utilizá-lo quando temos arestas negativas.