Teoria dos Grafos
COS-242

Trabalho 3

Alunos:

Ary Neto
João Pedro de Lacerda
Lucas Santiago

Projeto @ github:
https://github.com/familia-grafos/grafos/

I) Introdução:

O intuito desse trabalho é tratar do problema do Caixeiro Viajante Euclideano, em inglês, The Salesman Problem.

II) Por que C++?

A linguagem foi escolhida para permitir um melhor manuseio da memória do computador. Ela também suporta a criação de classes e a orientação a objetos, torna mais fácil a implementação de algoritmos.

III) Decisões e Complexidades:

Implementamos duas heurísticas: Algoritmo da Casca Convexa (Convex Hull) e o Closest First com a otimização 2-opt.

O algoritmo da casca convexa consiste em criar uma casca inicial dos pontos dados e esta casca é tratada como uma sub-rota. A partir daí, ir buscando rotas melhores onde dados dois pontos, busca-se um ponto que a distância do novo caminho, seja menor que a distância original. O algoritmo pára quando todos os vértices internos forem adicionados a sub-rota.
$O(n^2 \log n)$.

O algoritmo do Closest First é similar ao Prim. Ele funciona da seguinte forma: dado uma raiz, o algoritmo vai caminhando e construindo a rota com o ponto mais próximo.
$O(n+m)$

A otimização 2-opt, funciona de forma local, reduzindo a distância entre dois pares de vértices próximos e que, inicialmente, possuem arestas

que se sobrepõem. O 2-opt, checa se há possibilidade de combinar esses pontos de forma que as novas arestas não se cruzam.
$O(n^3)$, pois o swap foi implementado de maneira global, conferindo o caminho todo com O(n), neste caso. O swap ficou em processo de otimização para O(1).

IV) Classes e estruturas:

Utilizamos uma classe chamada ***ETSP*** (Euclidean TSP), que é a representação do grafo.
A classe possui os seguintes atributos:

- um vetor de vértices com as coordenadas;
- um vetor de booleanos para controle de vértices visitados;
- um vetor de vetores de adjacência com os valores;
- uma matriz de adjacência;
- um vetor com o caminho mínimo, que contém os índices;
- o número de vértices;
- um inteiro para a representação do grafo (se utiliza matriz ou vetores de adjacência);
- um vetor auxiliar para o 2-opt;
- uma pilha auxiliar pro algoritmo de casca convexa;

E os seguintes métodos:

- um método de inicialização as variáveis (initEssentials);
- um método de inicialização das estruturas (init);
- um método para adicionar distâncias à estrutura (addDistance);
- um método para retornar a distância de dois vértices (getDistance);
- um método de otimização (twoOpt);
- um método pra rota do vizinho mais próximo (closestFirst);
- um método da inserção mais barata (cheapestInsertion);
- e quatro métodos da heap (inserção, ordenações, remoção);

Uma classe ***Vertex*** que possui um inteiro para identidade, 2 floats para a posição e um operador < pra comparar a proximidade de vértices (preferência ao eixo Y).

Também é usada uma estrutura ***Tuple***, que é composta por 3 inteiros:
verA -> vértice contido na subrota;
verB -> vértice contido na subrota;
verR -> vértice ainda não contido na subrota;
e um float:
dist -> peso ao adicionar o vértice à subrota;

V) Problemas encontrados:

Encontramos um gargalo no 2-opt, na implementação do swap, onde cairia de O(n) para O(1).

VI) Estudos de casos:

Os valores aqui apresentados, são referentes ao algoritmo Closest First com a otimização 2-opt.

Grafo	**Time (s)**	**Minimum Distance**
points-5.txt	0,00158	1.728.245
points-10.txt	0,0091	2.921.770,25
points-20.txt	0,04686	4.199.096,00
points-50.txt	0,36033	6.571.397,00
points-100.txt	2,71764	9.140.787,00
points-200.txt	15,5747	12.699.448,00
points-500.txt	228,531	18.026.410,00
points-1000.txt	1496,16	25.698.446,00
points-2000.txt	532,466	37.692.660,00
points-5000.txt	-	inf
points-7500.txt	-	inf
points-10000.txt	-	inf