Trabalho Prático 0

Raul H. C. Lopes

23 de Junho de 2008

1 Objetivo do trabalho

Estudar variações de *binary trees* e suas aplicações como estruturas de indexação e *heap*.

2 O problema em foco

O problema do *Symmetric TSP* (*STSP*) estará no foco central do trabalho. Dado um conjunto de pontos no plano Euclideano de duas dimensões, objetiva-se encontrar o menor *tour* que passa por todos os pontos. Assuma que voce quer viajar por todos os pontos e voltar ao ponto de partida e que a distância entre dois pontos quaisquer é a distância Euclideana. Mais informações sobre este problemas e possíveis heísticas para resolvê-lo podem ser encontradas em [1], em especial no capítulo 9, *Experimental Analysis of Heuristics for the STSP*, de autoria de Johnson and McGeoch.

3 Os exercícios

Dois exercícios são propostos e ambos consistem em implementar heurísticas para resolver o problema do *STSP*.

Seu objetivo consiste em:

- propor estruturas e algoritmos para implementar as heurísticas;
- definir conjecturas sobre eficiência dos seus algoritmos;
- implementar seus algoritmos;
- testar exaustivamente seus algoritmos;

- relatar sobre seus testes, comparando resultados obtidos com as conjecturas iniciais.

O trabalho exigiráfundamentalmente a integração dos seguintes tipos de estrututas de dados:

- Estruturas de indexação espacial que permitam a execução rápida de consultas de: exclusão e localização de um ponto, busca de viznho mais próximo. Candidatos aqui são as variações multi-dimensionais de árvore binária e listas.
- Estruturas de *heap* ou filas de prioridade, que permitam recuperação rápida do maior ou menor item dentro de um conjunto e dada uma relação de precedência. Variações de árvores binárias e listas são candidatos óbvios.
- Estruturas para representação de conjuntos de inteiros.

Em todo o trabalho você esta' restrito a usar os algoritmos aqui especificados e as estruturas apresentadas em sala de aula.

3.1 Furthest Insertion

O primeiro exercício consite em implementar e testar exaustivamente algoritmo para a heurística *Furthest Insertion*.

A implementação desta heurística pode ter ganhos eficiência pela combinação de estruturas espaciais de indexação para auxílio, por exemplo, em pesquisas do tipo *Nearest Neighbour* e pelo uso de uma estrutura de heap para recuperação rápida de próximo ponto a inserir no *tour*.

Cada grupo apresentará ao menos duas implementações diferentes da heurística usando combinações de diferentes filas de prioridade com alguma estrutura de indexação espacial.

3.2 DMST

O segundo exercício consiste em implementar heurística que usa uma *Minimum Spanning Tree* para construir uma aproximação inicial para o *tour* e depois usa uma pilha para eliminar vértices duplicados.

A implementação desta heurística pode apresentar ganhos em eficiênica pela combinação de uma estrutura de indexação espacial com uma estrutura rápida para representação de conjuntos de pontos.

Cada grupo apresentará ao menos duas implmentações de *MST* para construir a heurística DMST. Opções que podem ser consideradas:

- Variações sobre algoritmo de Kruskall que usa uma estrutura que permite agrupar conjuntos de arestas.
- Variações sobre algoritmo de Dijkstra.

4 Entrada e Saída

Seu programa lerá os pontos para computação do *tour* de *standard input*. Um ponto por linha: coordenada x, seguida de espaço, seguida de coordenada y.

Seu programa produzirá em *standard output* os pontos do *tour*. Um ponto por linha: coordenada x, seguida de espaço, seguida de coordenada y.

Qualquer desvio desta especificação implicará na anulação do trabalho.

O trabalho deve ser submetido para

raulh.ufes at gmail dot com

Anexe um *tarball* contendo seu trabalho. Seu trabalho será descompactado e compilado com a seguinte seqüência de comandos:

```
tar -jxf cw0.tar.bz2
cd cw0
cd fi0
make
cd fi1
make
cd dmst0
make
cd dmst1
make
```

Qualquer falha em compilar implicará na anulação do trabalho.

5 Etapas do Trabalho

O trabalho pode ser desenvolvido individualmente, o que concede ao autor um bônus de 20% sobre a nota obtida, ou em grupos de dois alunos. Importante: no caso de desenvolvimente em grupo, os testes exaustivos finais e o relatório final ainda serão individuais. Nas etapas individuais, qualquer compartilhamento de informação será encarado como plágio levando à anulação do trabalho dos envolvidos. Na etapa de desenvolvimento, todos os

componentes do grupo devem conhecer/dominar o trabalho completamente. Se isso nao ocorrer em relação a qualquer algoritmo usado, o grupo todo perde a nota do trabalho.

O trabalho será desenvolvido de acordo com o seguinte cronograma:

1. Proposta de implementação.

   Cada grupo produzirá nesta etapa relatótio (um relatório por grupo) propondo:

   - Estruturas de dados e algoritmos que usará nas implementações.
   - Conjecturas sobre a eficiência das estruturas escolhidas.

2. Implementação dos algoritmos propostos por cada grupo.

3. Desenvolvimento de testes exaustivos, ajustes finais e relatório final.

   Esta etapa é estritamente individual e competitiva.

IMPORTANTE: documentos em *Word* serão imediatamente redirecionados para o lixo da história.

6 Referências úteis

Algumas referências úteis como fontes de algoritmos:

- Estruturas de dados espaciais: *Samet, H.*, "Foundations of Multidimensional and Metric Data Structures".
- Árvores binárias e árvores B: *Knuth, D.E.*: "The Art of Computer Programming", Vol I e Vol III.
- Self-adjusting structures: *Tarjan, R., Sleator, D.*, "Self-adjusting binary trees".

7 Datas importantes

- Especificação do Trabalho 0: 19 de maio.
- Especificação final do Trabalho 0: 26 de maio.
- Proposta de implementação: 02 de junho.
- Especificação Final do Trabalho 1: 11 de junho.

- Entrega de implementação do trabalho 0: 23 de junho, 19h.
- Entrega de proposta de desenvolvimento do trabalho 1: 20 de junho.
- Entrega de relatórios de testes do trabalho 0: 23 de junho.
- Entrega da implementação do trabalho 1: 7 de julho.
- Entrega de relatórios de testes do trabalho 1: 10 de julho.
- Prova (*Test 1*): 9 de julho.

Referências

[1] Gregory Gutin and Abraham P. Punen. *Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishers, 2002.