Projeto e Análise de Algoritmos
Trabalho 1

1 Descrição

Este trabalho considera a análise de soluções computacionais (algoritmos e implementações) para o *Problema do Para mais Próximo*.

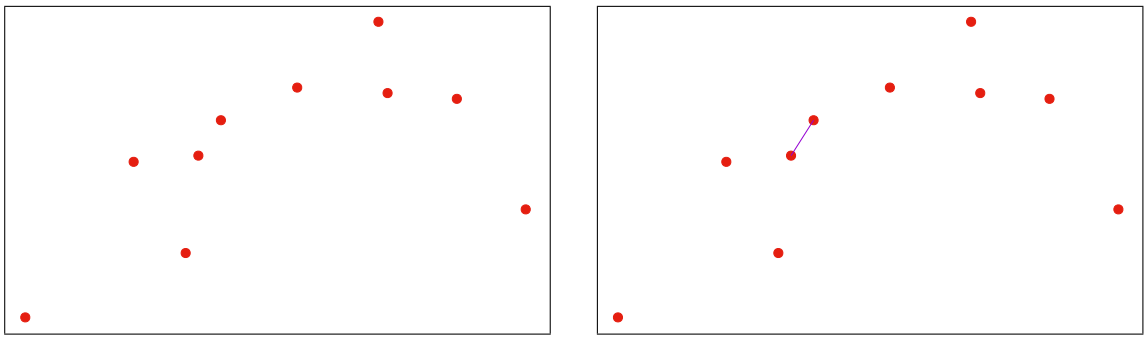
Cada trabalho submetido deverá ser desenvolvido por no máximo três estudantes. Os estudantes autores devem submeter um relatório, contento a análise dos algoritmos para o problema, e códigos, para as implementações dos algoritmos. O relatório e os códigos devem conter cabeçalhos discriminando cada um dos estudantes que desenvolveu o trabalho submetido, com nome completo e registo acadêmico.

A avaliação do trabalho irá considerar o relatório, os códigos e a execução das implementações. Todo o conteúdo submetido pelos estudantes deve ser de autoria deles.

Na sequência deste documento serão descritos os detalhes sobre o trabalho. Leia com atenção. Trabalhos que não respeitarem esta descrição serão penalizados.

2 Problema do Par mais Próximo

Dentre n pontos $p_1, p_2, \ldots, p_n$ no plano, encontrar pontos p_i e p_j, $i \neq j$, tal que a distância entre p_i e p_j é a menor dentre todas as distâncias entre os pares de pontos dados. Se mais de um par de pontos possui a distância mínima, então um destes pares é escolhido arbitrariamente como resposta. Um ponto é dado por duas coordenadas $p_i = (x_i, y_i)$. A distância entre dois pontos é definida como a distância Euclidiana: $d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. As Figuras 1a e 1b ilustram, respectivamente, um conjunto de pontos no plano e a distância entre o par de pontos mais próximo.



(a) (b)

Figura 1: Ilustração de alguns pontos no plano (a) e dos pontos mais próximos (b).

3 Algoritmos para o Problema

Neste trabalho deverão ser considerados dois algoritmos para o problema: algoritmo força bruta e algoritmo divisão e conquista.

O algoritmo força bruta consiste em computar a distância entre todos os pares de pontos. Claramente este algoritmo possui complexidade de tempo $O(n^2)$. O algoritmo de força bruta deve ser usado como base de tempo para as análises feitas no trabalho.

O problema pode ser resolvido também por divisão e conquista. Muito embora seja fácil formular algoritmos de divisão e conquista também com complexidade $O(n^2)$ para o problema, tais soluções não serão consideradas neste trabalho. Há um algoritmo conhecido de divisão e conquista que é assintoticamente mais rápido do que o algoritmo força bruta. No desenvolvimento deste trabalho você deve pesquisar e considerar um algoritmo assintoticamente mais rápido do que $O(n^2)$.

4 Relatório

Um relatório deverá ser desenvolvido focando no algoritmo por divisão e conquista, sua análise de corretude e sua complexidade assintótica de tempo. Também deverá ser apresenta uma análise sobre tempos de execução das implementações. O relatório deve ser entregue no formato PDF seguido o template de artigos da SBC encontrado no seguinte link `https://www.sbc.org.br/documentos-da-sbc/summary/169-templates-para-artigos-e-capitulos-de-livros/878-modelosparapublicaodeartigos`.

O relatório deverá conter as seguintes seções:

1. Resumo
2. Algoritmo Força Bruta
3. Algoritmo Divisão e Conquista

3.1 Algoritmo

3.2 Análise de Complexidade Tempo

4. Experimentos
5. Conclusão
6. Referências

Cada uma das seções do relatório será avaliada considerando a qualidade do texto, clareza das ideias, coesão e completude do conteúdo. Algumas considerações necessários para cada seção, sobre conteúdo e forma, são descritas a seguir. As descrições dos algoritmos devem ser textuais e, possivelmente, com o auxílio de pseudocódigos. Não devem ser adicionados códigos de sua implementação no relatório.

1. Resumo Um parágrafo de no máximo 10 linhas definindo o problema, as soluções algorítmicas encontradas e os resultados experimentais obtidos.

2. Algoritmo Força Bruta Descrição do algoritmo implementado e análise de complexidade de tempo. No máximo 1 página.

3. Algoritmo Divisão e Conquista Seção sobre o algoritmo de divisão e conquista encontrado e implementado, citando referências usadas na pesquisa da solução. No máximo 1/2 página.

3.1 Algoritmo Descrição do algoritmo de divisão e conquista. Deve conter descrições claras e completas sobre as etapas de divisão, conquista e combinação usadas pelo algoritmo. No máximo 3 páginas.

3.2 Análise de Complexidade Tempo Deve descrever a relação de recorrência derivada do algoritmo. A complexidade das etapas de divisão e combinação devem ser apresentadas e explicadas. A complexidade assintótica do algoritmo deve ser apresentada e justificada pela análise da relação de recorrência. No máximo 2 páginas.

4. Experimentos Reportar tempos de execução para os algoritmos força bruta e divisão e conquista implementados, descrevendo e comparando as particularidades de tempos para ambos os algoritmos em diferentes tamanhos de entrada. Apresentar e discutir gráficos de tempo por tamanho de entrada. No máximo 2 páginas.

5. Conclusão Descrever de forma sucinta as conclusões sobre as soluções, qualidades e falhas da solução encontrada (algoritmos e/ou implementação). Máximo de 20 linhas.

6. Referências Referências pesquisadas e citadas.

5 Implementação e Execução

As implementações devem ser feitas em linguagem C ou C++ e compilar, respectivamente, usando os compiladores gcc e g++, em sistema operacional Linux.

Um arquivo Makefile deve ser disponibilizado para compilar e gerar o arquivo executável. A compilação deve ser feita usando o comando *make* em um terminal de comandos do Linux, conforme o exemplo abaixo. O símbolo $ representa o prompt de comando do terminal.

```
$ make
```

O programa gerado pelo comando *make* deve ter o nome "closest" e receber como argumento um arquivo TXT com os pontos de entrada para execução das implementações do algoritmos força bruta e divisão e conquista (o formato do arquivo será descrito na sequência). A execução deve ser iniciada em um terminal de comandos do Linux, como no exemplo abaixo.

```
$ ./closest input.txt
```

O programa "closest" deve computar os pares mais próximos tanto para o método força bruta quanto para o método de divisão e conquista. Assim que terminar a execução dos métodos, o programa deve apresentar a saída (descrita na sequência), na saída padrão do terminal Linux, e finalizar a execução.

5.1 Arquivo de Entrada

O arquivo de entrada deve ser formatado da seguinte maneira:

- a primeira linha contém um inteiro n com o número de pontos no arquivo a serem lidos como entrada para o programa;
- as n linhas que seguem contêm as coordenadas dos pontos em valores reais. Cada linha representa um ponto e contém os valores de x e depois de y, separados em duas colunas por um espaço.

Um exemplo de arquivo de entrada com 10 pontos é mostrado abaixo.

```
10
3091.627826 601.006476
3248.708916 4082.621268
3845.638928 184.651283
7545.393290 1154.317298
7748.718331 9864.560519
2608.641732 838.303203
5728.058273 6604.539820
702.610314 8500.248091
8794.220145 6218.940344
405.380470 2500.490589
```

No Moodle da disciplina foi disponibilizado um código para gerar um arquivo de pontos aleatórios neste formado. O código tem nome "genpoints.c". Para compilar basta executar o comando abaixo.

```
$ gcc genpoints.c -o genpoints
```

O programa recebe como argumento o número de pontos desejados, conforme o exemplo abaixo,

```
./genpoints 1000000
```

e gera um arquivo chamado "input.txt" no formato da entrada.

5.2 Formatação da Saída

A saída do programa "closest" deve ser formada em uma única linha da seguinte maneira:

$$fbt \quad fbd \quad fbx_1 \quad fby_1 \quad fbx_2 \quad fby_2 \quad dct \quad dcd \quad dcx_1 \quad dcy_1 \quad dcx_2 \quad dcy_2$$

tal que

- fbt é o tempo de execução do método força bruta em segundos;
- fbd é a distância entre os pontos mais próximos encontrados pelo método força bruta;
- fbx_1, fby_1, fbx_2 e fby_2 são as coordenadas para os dois pontos mais próximos encontrados pelo método força bruta;

- *dct* é o tempo de execução do método de divisão e conquista em segundos;
- *dcd* é a distância entre os pontos mais próximos encontrados pelo método de divisão e conquista;
- dcx_1, dcy_1, dcx_2 e dcy_2 são as coordenadas para os dois pontos mais próximos encontrados pelo método de divisão e conquista.

Os valores devem ser separados por um espaço e com uma quebra de linha após o último valor.

Considerando o arquivo de exemplo descrito na seção anterior, ao executar o programa, um exemplo de saída esperada seria o seguinte (apenas para ilustração, o tamanho da fonte da saída foi diminuído para caber em uma linha). Os valores devem ser impressos em ponto flutuante conforme o exemplo.

```
./closest input.txt
```

```
0.000004 538.131 3091.627826 601.006476 2608.641732 838.303203 0.000003 538.131 2608.641732 838.303203 3091.627826 601.006476
```

6 Entrega

As entregas deverão ser realizadas usando o Moodle da disciplina dentro do prazo limite estabelecido no mesmo. Não serão aceitas entregas fora do prazo.

Os autores do trabalho devem reunir os códigos-fonte e o relatório em um diretório nomeado com os números dos seus registros acadêmicos (RAs) separados por '_'. Por exemplo, se dois estudantes desenvolveram o trabalho e têm RAs 123456 e 789012, então o diretório deve ter o nome 123456_789012.

Dentro do diretório deverão estar os códigos-fonte, o arquivo Makefile e o relatório. Por exemplo, o diretório poderia conter os arquivos listados abaixo (não coloque acentos ou espaços em nomes de arquivos).

```
123456_789012/closest.c
123456_789012/Makefile
123456_789012/123456_789012.pdf
```

O nome do relatório entregue também deve seguir o padrão para nomear o diretório dos arquivos. Portanto, no exemplo, o relatório tem o nome 123456_789012.pdf.

O arquivo de entrega deve ser um ZIP do diretório. O arquivo ZIP deve inclusive conter o diretório e ser nomeado da mesma maneira que o diretório. No nosso exemplo: 123456_789012.zip.