Trabalho Prático 2

Algoritmos I

Entrega: 14/05/2019

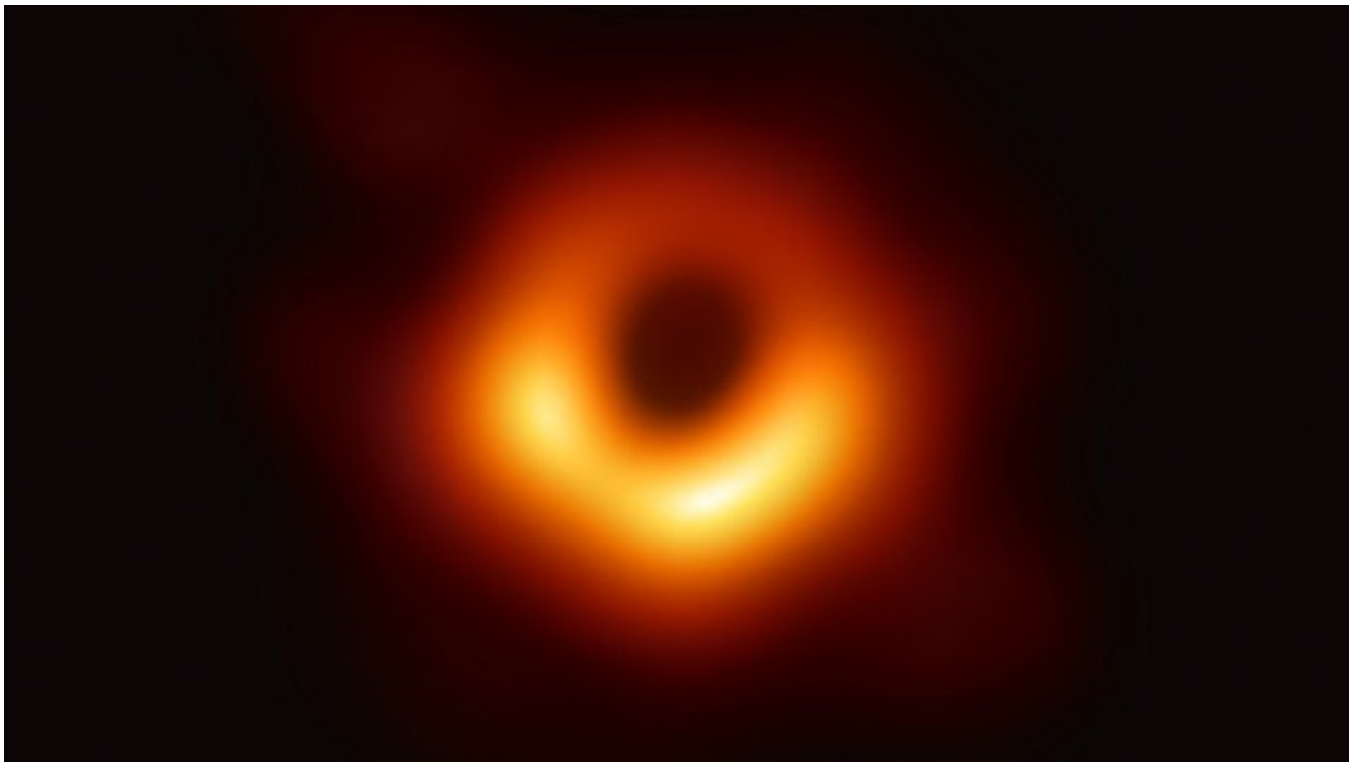


Figura 1: Primeira fotografia de um buraco negro

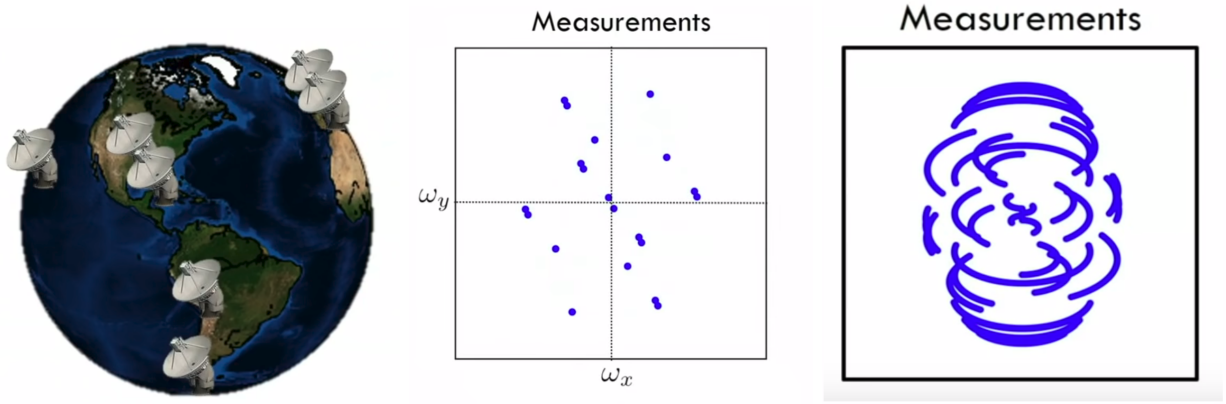
1 Introdução

No dia 10 de abril de 2019, o mundo viu pela primeira vez uma fotografia de um buraco negro, localizado na galáxia Messier 87 a 55 milhões de anos luz da Terra, possuindo uma massa de aproximadamente 6.5 bilhões de vezes a massa do Sol. A fotografia traz consigo diversas contribuições para o meio científico, como o estudo de um dos objetos mais extremos no nosso universo como também a validação das predições da teoria geral da relatividade de Einstein. A fotografia foi tirada pelo Event Horizon Telescope (EHT), uma colaboração internacional constituído por oito radio telescópios distribuídos pelo planeta, e montada por uma equipe de mais 200 pesquisadores.

Conseguir a fotografia do buraco negro não foi um feito simples. Em um vídeo feito por uma das

pesquisadoras do projeto[1], podemos ver que há uma simples equação para calcular o menor tamanho de um objeto que conseguimos ver dado o comprimento da onda e o tamanho do telescópio (tamanho objeto = $\frac{\text{comprimento de onda}}{\text{tamanho do telescópio}}$). Com essa função, conseguimos calcular que, para ver uma laranja na superfície da lua, precisaríamos de um telescópio do tamanho da terra. Como o objeto está tão distante da Terra, muito mais distante do que a lua, tirar essa fotografia se torna muito difícil.

O problema foi solucionado da seguinte forma. Primeiramente, foram utilizados radio telescópios, que conseguem capturar ondas de um comprimento muito maior que o espectro visível. Em sequência, oito radio telescópios distribuídos pela Terra foram apontados para a direção do buraco negro (Figura 2(a)) e começaram a registrar as informações recebidas daquela direção (Figura 2(b)). Devido à rotação e à translação da terra, esses oito radio telescópios foram capturando diferentes pontos da imagem (Figura 2(c)) ao longo de vários dias. Entretanto, como é possível observar na Figura 2(c) ainda não obtivemos a imagem completa. Para resolver esse problema, uma equipe construiu algoritmos para preencher os espaços vazios dessa imagem com os pixeis mais prováveis dado todo o nosso atual conhecimento e observação do universo (Para os curiosos, como o algoritmo foi criado e modelado pode ser visto no TED Talk da Katie Bouman no rodapé).



(a) Event Horizon Telescope (EHT) (b) Observações do buraco negro dado que a terra fique parada (c) Observações do buraco negro com a rotação e translação da terra

Figura 2: Escopo geral do problema de fotografar um buraco negro

A fotografia do buraco negro gerou um grande interesse por astrofotografia no mundo inteiro: pessoas dos mais diversos tipos ficaram interessadas em construir uma grande rede de radio telescópios ao redor do mundo para obter uma maior parte da imagem original e assim precisar que menos espaços sejam preenchidos pelo algoritmo.

Um investidor misterioso está muito interessado em entrar no ramo. Entretanto, ele está preocupado com como esses telescópios se comunicam uns com os outros, dado que, para a geração da fotografia do buraco negro, foram necessários Petabytes de dados. Esse investidor pretende construir uma rede de n telescópicos de forma que sempre exista um caminho de comunicação entre qualquer par de telescópios. Para isso, existem dispositivos de comunicação que conseguem transmitir dados em uma velocidade muito alta. Entretanto, cada dispositivo só consegue transmitir até uma distância D, e quanto maior o limite de distância D, mais caro o dispositivo é. Dado que o custo do dispositivo de comunicação aumenta exponencialmente com D, faz-se necessário minimizar o valor de D. Além disso, esses dispositivos só conseguem se comunicar uns com os outros se eles forem exatamente do mesmo tipo (todos são configurados com o mesmo valor de D, tendo um alcance máximo de comunicação $\leq D$). Por fim, ele também quer

[1]How to take a picture of a black hole - Katie Bouman: `https://www.youtube.com/watch?v=BIvezCVcsYs`

simplificar a comunicação entre os telescópios e quer que só exista um caminho entre dois pontos e que não existam ciclos de comunicação.

Dado isso, o investidor então contratou você para construir um algoritmo que, dadas n posições na Terra onde ele tem a intenção de construir telescópios (pode ser no meio dos oceanos), forneça a distância mínima D para que todos os telescópios fiquem interligados por uma rede de comunicação (lembre que o preço dos comunicadores aumenta exponencialmente conforme seu alcance máximo de comunicação D!), que a ligação seja acíclica e que só exista um caminho entre cada par de telescópios.

Note que o problema pode ser modelado através de um grafo ponderado completo $G = (V, A)$ com $|V| = n$ vértices, um para cada dispositivo, e $|A| = m = n(n-1)/2$ arestas, cujos custos são as distâncias entre cada par de dispositivos. Potencialmente, todo par de dispositivos $(u, v) \in V$ pode se comunicar, desde que $D \geq d_{uv}$), onde d_{uv} é o custo da aresta $(u, v) \in E$. Finalmente, dado que o nosso investidor está com muita pressa, ele requer que o algoritmo para computar a infra-estrutura ótima de comunicação especificada acima tenha uma complexidade linear em função do número de arestas do grafo G, ou seja, $O(m)$.

2 O que fazer?

O objetivo deste trabalho é, dado um conjunto de n posições de telescópios ao redor da Terra, onde cada posição corresponde a um par $(lat, long)$, onde $-90 \leq lat \leq 90$ e $-180 \leq long \leq 180$ com precisão de 5 casas decimais, modelar um grafo $G(V, A)$ com a instância do problema, onde V são as posições dos telescópios e existe uma aresta (u, v) com distância d_{uv} entre cada par de cidades, construir um algoritmo de complexidade até $O(m)$, onde $m = |A|$ a quantidade de arestas, em que ele deve retornar uma distância D que representa a menor distância necessária para que todas as cidades fiquem interligadas **sem ciclos** e **sem que nenhuma ligação (aresta) ultrapasse a distância** D. Ou seja, queremos interligar todos os telescópios da melhor forma possível (sem precisar comprar um comunicador que consiga transmitir uma distância maior que o necessário e portanto seja mais caro que o necessário). É importante notar que o que queremos é **minimizar a distância máxima necessária para conectar todos os pares de cidades sem ciclos com o comunicador mais barato possível**, dado que **todos os comunicadores precisam ser idênticos para comunicarem uns com os outros**.

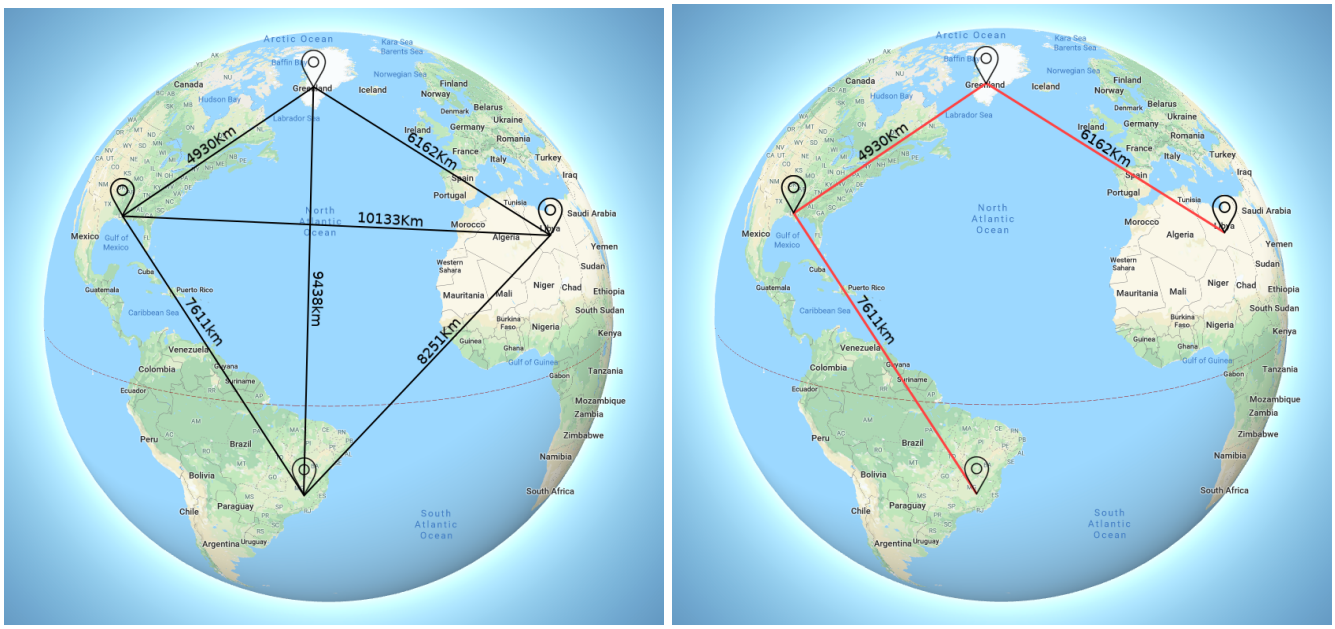
A Figura 3 ilustra uma instância simplificada do problema. Temos quatro telescópios ao redor do globo e queremos conectá-los. Com base nas localizações, podemos visualizar o grafo completo na Figura 3(a), e se quisermos que todos os telescópios se comuniquem entre si desta forma, precisaríamos que todos os comunicadores conseguissem se comunicar numa distância D de 10,133 quilômetros, que corresponde à maior distância do grafo. Entretanto, podemos conectar todos os telescópios com a rota descrita na Figura 3(b), e com isso, conseguimos reportar que todos os comunicadores precisarão se comunicar até uma distância de 7,611 quilômetros, que representa a maior distância da rota proposta.

Para calcular a distância entre dois pontos no globo dadas suas respectivas latitudes e longitudes, use o algoritmo listado a seguir. Note que estamos simplificando a distância convertendo-a para um valor inteiro. Para compilar, como estamos utilizando a biblioteca math, lembre-se de utilizar a tag $-lm$ no gcc (e.g., gcc main.c -o main -lm)

```
#include <math.h>
#define earthRadiusKm 6371.0

double deg2rad(double deg) {
    return (deg * M_PI / 180);
}

int distanceEarthKm(double lat1d, double lon1d, double lat2d, double lon2d) {
    double lat1r, lon1r, lat2r, lon2r, u, v;
```



(a) Localização dos telescópios de interesse e o grafo completo

(b) Uma possível interligação dos telescópios onde a maior distância necessária para os comunicadores é mínima (7,611 quilômetros)

Figura 3: Exemplo de entrada

```
    lat1r = deg2rad(lat1d);
    lon1r = deg2rad(lon1d);
    lat2r = deg2rad(lat2d);
    lon2r = deg2rad(lon2d);
    u = sin((lat2r - lat1r)/2);
    v = sin((lon2r - lon1r)/2);
    return (int) 2.0 * earthRadiusKm * asin(sqrt(u * u + cos(lat1r) * cos(lat2r) * v * v));
}
```

3 Os arquivos de entrada

Arquivo contendo candidatos

```
4                                   // <Qtd Cidades>
-20.133724 -44.128386               // <lat> <long>
31.083062 -88.773256
63.604391 -43.435838
25.868161 15.836897
```

4 O arquivo de saída

Saída Esperada
7424

5 Avaliação Experimental

Para a avaliação experimental o aluno deverá criar entradas para o problema de forma a avaliar se o tempo de execução do algoritmo está de acordo com a complexidade reportada na documentação, aumentando o número de vértices da entrada, guardando os tempos de execução e colocando esses valores em gráficos para melhor visualização. É **necessário** executar mais de uma vez o algoritmo para entradas com o mesmo tamanho de cidades (no mínimo 10 vezes), guardando o tempo de execução e reportando a **média** e o **desvio padrão** deste tempo. Faça isso para vários números de cidades (exemplo: 4, 8, 12, 16, 20, ... etc). Isso é necessário para observar como o tempo está se comportando dado diferentes execuções e diferentes configurações do grafo.

6 O que deve ser entregue

Você deve submeter um arquivo compacto (**zip**) no formato **seu_nome_sua_matrícula** via Moodle contendo:

- todos os arquivos do código *.c* e *.h* que foram implementados,
- um arquivo *makefile*[2] **que crie um executável tp2**,
  - **ATENÇÃO:** O makefile é para garantir que o código está sendo compilado corretamente, de acordo com o modo que vocês modelaram o programa. É **essencial** que ao digitar "make" na linha de comando dentro da pasta onde reside o arquivo makefile, o mesmo compile o programa e gere um executável chamado **tp2**.
- sua documentação.

Sua documentação deve ter até 10 páginas contendo:

- uma breve descrição do problema,
- explicações das estruturas de dados e dos algoritmos utilizados para resolver o problema. Para tal, artifícios como pseudo-códigos, exemplos ou diagramas podem ser úteis. Note que documentar uma solução não é o mesmo que documentar seu código. Não inclua textos de códigos em sua documentação.
- análise de complexidade da solução proposta (espaço e tempo). Cada complexidade apresentada deverá ser devidamente justificada para que seja aceita.
- prova de corretude do algoritmo.

[2] https://pt.wikibooks.org/wiki/Programar_em_C/Makefiles

- avaliação experimental.

O seu TP deverá ser entregue de acordo com a data especificada no moodle. A penalidade em porcentagem para os TPs atrasados é dada pela fórmula $2^{d-1}/0.16$.

7 Implementação

7.1 Linguagem, Ambiente e Parâmetros

O seu programa deverá ser implementado na linguagem **C** e poderá fazer uso de funções da biblioteca padrão da linguagem. Trabalhos que utilizem qualquer outra linguagem de programação e/ou que façam uso de bibliotecas que não a padrão não serão aceitos. Além disso, certifique-se que seu código compile e execute corretamente nas máquinas **Linux** dos laboratórios do **DCC**.

ATENÇÃO: O arquivo da entrada deve ser passado como parâmetros para o programa através da linha de comando (e.g., *$ ./main cidades1.txt*) e imprimir a saída no **stdout** (com printf), não em um arquivo.

ATENÇÃO: certifique-se que o programa execute com os três arquivos passados como exemplos junto com a documentação no Moodle (dataset_tp2.zip), sem alterar os arquivos passados. Isso vai garantir que seu programa vai conseguir ler corretamente os arquivos do corretor automático.

7.2 Testes

A sua implementação passará por um processo de correção automática, o formato da saída de seu programa deverá ser idêntico aquele descrito nas seções anteriores. Saídas diferentes serão consideradas erro para o programa. Para auxiliar na depuração do seu trabalho, será fornecido um pequeno conjunto de entradas e suas saídas. É seu dever certificar que seu programa atenda corretamente para qualquer entrada válida.

7.3 Qualidade do código

É importante prestar atenção para a qualidade do código, mantendo-o organizado e comentado para não surgir dúvidas na hora da correção. Qualquer decisão que não estiver clara dada a documentação e a organização do código será descontada na nota final.

8 Consideração Final

Assim como em todos os trabalhos dessa disciplina é estritamente proibido a cópia parcial ou integra de códigos, seja da internet ou de colegas. Seja honesto! Você não aprende nada copiando código de terceiros. Se a cópia for detectada, sua nota será zerada e o professor será informado para que as devidas providências sejam tomadas.