Laboratórios

Conteúdo

1 Maior prefixo comum 2

2 Computando vetores 2

3 Jogo de beisebol 3

4 Diamantes e areia 3

5 Listas de nomes 4

6 Jogando cartas fora 4

7 A noiva do trevo 5

8 Recontagem 6

9 Registro de conversa 7

10 Análise de redes complexas 7

11 Maximizando vendas 8

1 Maior prefixo comum

— LeetCode: Longest Common Prefix (# 14, adaptado)
https://leetcode.com/problems/longest-common-prefix

Dado um vetor de strings, encontre o maior prefixo comum entre as strings. Se `s` é um prefixo comum de um conjunto de strings, então todas elas iniciam com `s`. Caso não exista prefixo comum, retorne uma string vazia.

Exemplos:

1. Strings: `["flower", "flow", "flight"]`. Maior prefixo comum: `"fl"`.
2. Strings: `["cadete", "cadeira", "academia"]`. Maior prefixo comum: `""`.

Abordagem 1: percorre cada caractere da primeira string e verifica se o mesmo caractere é encontrado na posição correspondente de todas as demais strings. Em caso positivo, essa posição faz parte do prefixo comum e a verificação prossegue ao próximo caractere. Caso contrário, o caractere não faz parte do prefixo comum; interrompe a execução e retorna o prefixo comum encontrado.

Abordagem 2: ordena a coleção de strings em ordem alfabética e compara somente a primeira e última strings. O prefixo comum encontrado será comum a todas as strings da coleção.

Etapas:

1. Projete os dois algoritmos apresentados (pseudocódigo).
2. Qual a complexidade assintótica deles?
3. Implemente ambos os algoritmos.
4. Implemente um método gerador de instâncias.
   - Gera um vetor com N strings compostas pelos caracteres {A, B, C, D}.
   - Para isso, gera uma string base com 5 a 10 caracteres. Para construir a coleção, cria uma cópia da string base e aplica modificações em alguns caracteres, dando probabilidade de modificação proporcional à posição do caractere na string.
5. Gere 20 instâncias com $N \in \{2, 2^2, 2^3, \ldots, 2^{20}\}$.
6. Resolva as 20 instâncias com os dois algoritmos e compare o tempo de execução deles.

---

2 Computando vetores

Considere um vetor não ordenado $\mathcal{A}$ de números inteiros com n elementos. Queremos extrair duas informações:

1. A soma de todos os valores de $\mathcal{A}$ em um dado intervalo $[a, b]$.
2. O k-ésimo menor elemento.

Vamos implementar essas duas operações considerando dois cenários. No primeiro cenário, não vamos alterar o conteúdo de $\mathcal{A}$. No segundo cenário, vamos inicialmente ordenar os elementos de $\mathcal{A}$, e então

aproveitar a estrutura ordenada para reimplementar as operações de maneira mais eficiente. Para comparar o custo computacional nos dois cenários, vamos gerar vetores com números inteiros aleatórios no intervalo $[0, 1000]$, $n \in \{10^2, 10^4, 10^6, 10^8\}$, $a = 580$, $b = 788$ e $k = 73$.

Monte uma tabela comparando o tempo de processamento de ambos os cenários para executar o conjunto das duas operações. No caso do segundo cenário, o tempo necessário para ordenação do vetor deve ser incluído no tempo de processamento.

---

3 Jogo de beisebol

[PILHAS]

— LeetCode: Baseball Game (# 682)
https://leetcode.com/problems/baseball-game

Você está anotando o placar de um jogo de beisebol com regras estranhas. No início do jogo, você começa com um registro vazio.

Você recebe uma sequência de strings S, onde S_i é a i-ésima operação que você deve aplicar ao registro e é uma das seguintes:

- Um inteiro "`x`": registra a nova pontuação `x`.
- "`+`": registra a nova pontuação como a soma das duas pontuações anteriores.
- "`D`": registra a nova pontuação como o dobro da última pontuação.
- "`C`": invalida a última pontuação, removendo-a do registro.

Retorne a soma de todas as pontuações do registro após aplicar todas as operações. Para a operação "`+`", sempre haverá ao menos duas pontuações prévias no registro. Para as operações "`C`" e "`D`", sempre haverá ao menos uma pontuação prévia no registro.

Entrada: `5 2 C D +` **Saída:** `30`

Entrada: `5 -2 4 C D 9 + +` **Saída:** `27`

Entrada: `1 C` **Saída:** `0`

---

4 Diamantes e areia

[PILHAS]

— Beecrowd: Diamonds and Sand (# 1069)
https://judge.beecrowd.com/en/problems/view/1069

John está trabalhando em uma mina de diamantes, tentando extrair o maior número possível de diamantes "`<>`". Ele deve remover todas as partículas de areia encontradas "`.`" nesse processo, e depois de extrair um diamante, novos diamantes podem ser formados. Se ele tiver como entrada "`.<...<<..>>....>....>>>.`" três diamantes são formados. O primeiro é retirado do "`<..>`", resultando em "`.<...<>....>....>>>.`". O segundo diamante é então removido, deixando "`.<.......>....>>>`". O terceiro diamante é então retirado, deixando no final "`.....>>>.`" sem possibilidade de extração de novos diamantes.

Entrada: `<..><.<..>>` **Saída:** `3`

Entrada: `<<<..<.....<<<<....>` **Saída:** `1`

---

5 Listas de nomes [FILAS]

— Beecrowd: Name Lists (# 3135)
https://judge.beecrowd.com/en/problems/view/3135

Marta quer escolher alguns nomes para seu futuro filho ou filha. Ela encontrou uma lista de nomes, mas não gosta da sua forma de apresentação. Ela gostaria de ter uma lista de nomes, onde cada linhas fosse ordenada conforme o tamanho (comprimento) do nome, do menor para o maior. Em cada linha, somente um nome de cada tamanho deve aparecer. Por exemplo, dada a lista de nomes Eva, Ana e Lucas, Eva e Lucas devem aparecer na primeira linha, enquanto Ana deve aparecer na segunda linha.

Que tal escrevermos um algoritmo para produzir essa lista de nomes?

A entrada consiste na quantidade N de nomes, seguida de N linhas contendo um nome em cada, cujo comprimento será de 2 a 10. A saída consiste em uma ou mais linhas com as listas de nomes, ordenados pelo tamanho. Em cada linha, deve haver somente um nome de cada tamanho. Outros nomes com o mesmo tamanho aparecerão nas linhas seguintes, conforme a ordem em que aparecem na entrada.

Entrada
```
12
sergio
ana
maria
carlos
eva
joaquim
jo
mara
laura
lucas
ari
paulo
```

Saída
```
jo, ana, mara, maria, sergio, joaquim
eva, laura, carlos
ari, lucas
paulo
```

---

6 Jogando cartas fora [LISTAS]

— Beecrowd: Throwing Cards Away (# 1110)
https://judge.beecrowd.com/en/problems/view/1110

Considere um baralho ordenado de n cartas numeradas de 1 a n, com a carta 1 no topo do baralho e a carta n no final. A seguinte operação é realizada enquanto houver pelo menos duas cartas no baralho:

- Joga fora (descarta) a carta do topo do baralho, e mova a nova carta do topo para o final do baralho.

Sua tarefa é encontrar a sequência de cartas descartadas e a última carta restante. Para cada entrada (número), o algoritmo deve produzir duas linhas de saída. A primeira linha apresenta a sequência de cartas descartas, enquanto a segunda linha mostra a última carta restante.

Entrada
```
7
```
Saída
```
Discarded:  1, 3, 5, 7, 4, 2
Remaining:  6
```

Entrada
```
19
```
Saída
```
Discarded:  1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 4, 8, 12, 16, 2, 10, 18, 14
Remaining:  6
```

Entrada
```
10
```
Saída
```
Discarded:  1, 3, 5, 7, 9, 2, 6, 10, 8
Remaining:  4
```

---

7 A noiva do trevo

[FILAS DE PRIORIDADE]

— Beecrowd: The Clover Bride (# 3149)
https://judge.beecrowd.com/en/problems/view/3149

Existe uma história famosa de uma noiva que aparece no trevo da entrada da cidade, sempre próximo da meia noite. Muitos moradores da cidade afirmam já ter visto a noite, mas não há um consenso sobre a hora exata da sua aparição, principalmente porque a maioria das pessoas fala "próximo da meia noite".

Um estudioso de eventos sobrenaturais está tentando organizar os depoimentos dos moradores, para verificar se algum deles é verdadeiro. Como "próximo da meia noite" pode ser alguns minutos antes ou depois da meia noite, o estudioso precisa da sua ajuda para criar um algoritmo que, dado um limite de M minutos a serem considerados antes ou depois da meia noite, o nome do morador e a hora da aparição, mostre os depoimentos de forma ordenada.

A primeira linha da entrada apresenta a quantidade de minutos M (antes e depois da meia noite) a considerar depoimentos, e o total de depoimentos N. As próximas N linhas apresentam a hora H da aparição e o nome X do morador. A saída consiste no nome dos moradores que relatam aparições dentro do intervalo de tempo considerado. A ordem de exibição dos nomes é conforme o horário da aparição (mais cedo aparece antes). No caso de relatos de aparições no mesmo horário, é apresentado o nome do morador que registrou o depoimento antes (i.e., apresentado antes na entrada).

Entrada
```
15 5
23:44 Marcelo
00:11 Carlos
00:09 Ana
00:30 Nicolas
00:10 Bernardo
```
Saída
```
Ana
Bernardo
Carlos
```

Entrada
```
10 5
23:44 Marcelo
00:10 Carlos
00:09 Bernardo
00:30 Nicolas
00:10 Ana
```
Saída
```
Bernardo
Carlos
Ana
```

---

8 Recontagem

[MAPAS]

— Kattis: Recount
https://open.kattis.com/problems/recount

As eleições para o conselho escolar foram fortemente contestadas: uma proposta para trocar os horários de início das aulas para alunos do ensino fundamental e médio, uma controversa proposta de novo código de vestimenta que proíbe roupas esportivas na escola e uma proposta para aumentar os impostos imobiliários para pagar por um novo espaço de treinamento de futebol. E a lista continua! Já se passaram horas após o encerramento das urnas e ainda não surgiu um vencedor.

Em seu desespero, a comissão eleitoral recorre a você para que escreva um programa para contar os votos.

A entrada consiste na lista de votos. Cada linha contém o nome de um candidato em quem foi votado. Um nome pode consistir em várias palavras, separadas por espaços. As palavras contêm letras ou hifens, mas nenhum outro caractere de pontuação. Haverá pelo menos 2 votos na lista, e ela termina com uma única linha contendo os caracteres "***"'. Esta linha não deve ser contada. Pode haver até 100 000 votos válidos.

Se um candidato obteve a maioria simples dos votos expressos (ou seja, mais do que qualquer outro candidato), apresente o nome desse candidato. Se nenhum candidato obtiver maioria simples, produza "Segundo turno!".

Entrada

```
Penny Franklin
Marti Graham
Connie Froggatt
Joseph Ivers
Connie Froggatt
Penny Franklin
Connie Froggatt
Bruce Stanger
Connie Froggatt
Barbara Skinner
Barbara Skinner
***
```

Saída

```
Connie Froggatt
```

Entrada

```
Penny Franklin
Connie Froggatt
Barbara Skinner
Connie Froggatt
Jose Antonio Gomez-Iglesias
Connie Froggatt
Bruce Stanger
Barbara Skinner
Barbara Skinner
***
```

Saída

```
Segundo turno!
```

9 Registro de conversa

[TABELAS HASH]

— Kattis: Conversation log
https://open.kattis.com/problems/conversationlog

A popular rede social *My+Din* está lutando para gerenciar seus muitos fóruns. A regulamentação recente exige que o site denuncie usuários envolvidos em conversas sobre determinados tópicos. O grande número de usuários implica que o monitoramento manual é muito caro e, por isso, o site pediu a seus muitos estagiários que encontrassem uma solução.

Um estagiário teorizou que as conversas sobre qualquer tópico terão as mesmas palavras-chave usadas continuamente. Se as palavras mais utilizadas puderem ser identificadas, talvez a investigação manual possa ser direcionada para fóruns apropriados.

A **entrada** consiste no número de mensagens M, seguido de M linhas contendo o nome de usuário e o conteúdo da mensagem em letras minúsculas.

A **saída** deve conter as palavras usadas por todos os usuários no fórum, uma por linha, ordenadas da mais usada para a menos usada. Em caso de empates, apresentar as palavras em ordem alfabética. Se não houver palavra, a saída deve ser "ALL CLEAR".

Entrada	**Saída**

`8`
`Jepson no no no no nobody never`
`Ashley why ever not`
`Marcus no not never nobody`
`Bazza no never know nobody`
`Hatty why no nobody`
`Hatty nobody never know why nobody`
`Jepson never no nobody`
`Ashley never never nobody no`
	`no`
	`nobody`
	`never`

Entrada	**Saída**

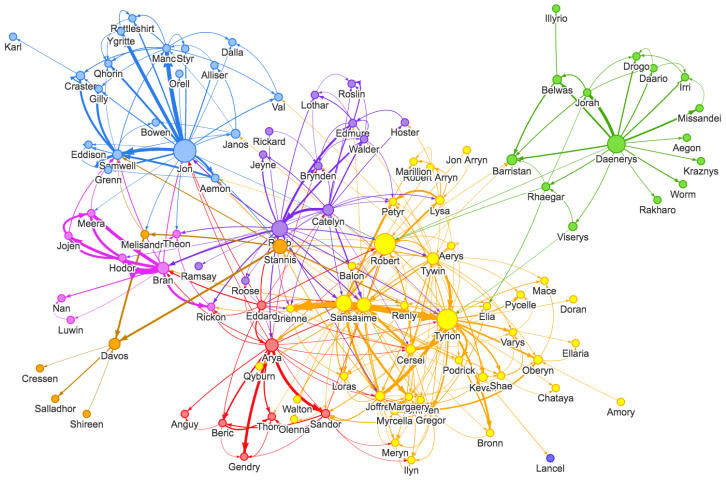
`2`
`Villain avast`
`Scoundrel ahoy`
	`ALL CLEAR`

10 Análise de redes complexas

[GRAFOS]

Grafos são frequentemente usado para modelar e analisar cenários do mundo real, como redes de tráfego [1], dinâmicas de propagação de doenças [2] ou redes de colaboração [3]. Uma aplicação interessante é na análise da interação entre personagens em enredos, como obras literárias, filmes ou séries [4, 5, 6]. As métricas estudadas em sala de aula fornecem ferramentas interessantes para analisar redes de interação, como os graus dos vértices, a densidade do grafo, os tamanhos dos subgrafos completos e o comprimento dos caminhos entre vértices. Além disso, existem outras métricas (algumas específicas para esse tipo de rede) interessantes, como medidas de centralidade (e.g. *betweenness*) e *PageRank*. Também estão disponíveis muitas ferramentas para tratar e analisar grafos. Exemplos incluem NetworkX[1], Gephi[2], graph-tool[3], Neo4J[4] e igraph[5]. Finalmente, há muito material (principalmente livros e artigos científicos

[1] https://networkx.org
[2] https://gephi.org
[3] https://graph-tool.skewed.de
[4] https://neo4j.com
[5] https://python.igraph.org



de qualidade) sobre análise de redes sociais. Um bom exemplo é encontrado em https://ericmjl.github.io/Network-Analysis-Made-Simple.

Faça um estudo das redes de interação na série *Game of Thrones*. Os dados (personagens e interações) podem ser obtidos em https://github.com/mathbeveridge/gameofthrones. Construa os grafos de interações das diferentes temporadas da série e defina o conjunto de métricas (e algoritmos) para analisar essas redes. Além de computar os valores para essas métricas, gere visualizações das redes para sua inspeção visual. Uma sugestão é seguir as abordagens adotadas em outros trabalhos de análise de redes sociais.

Ao final, produza um relatório com os resultados (métricas e visualizações). Discuta esses resultados e apresente as conclusões obtidas. Compare os achados com outros trabalhos da literatura.

---

11 Maximizando vendas [GRAFOS]

A *Tenises* comercializa tênis no atacado. Com grande frequência, a empresa participa de feiras de calçados por todo o país, onde expõe seus produtos a varejistas e fecha contratos para fornecimento de tênis dos mais diversos modelos. A experiência mostra que quanto maior a variedade de produtos (i.e. tênis com formatos, cores e estilos diversos), maior a probabilidade de venda e sucesso nos contratos. Diante dessa informação, a empresa fez um esforço para classificar cada modelo de tênis conforme uma série de características. Com isso, a empresa consegue determinar, quantitativamente, quão diferentes são dois tênis um do outro. Mesmo com todo esse trabalho realizado, decidir quais modelos levar para a feira é uma tarefa difícil, pois os estandes comportam um número limitado de unidades e a empresa possui muitos modelos no seu catálogo. Vamos implementar um algoritmo para decidir quais modelos de tênis levar à feira, de modo que esse grupo seja o mais diverso (i.e. diferente) possível!

Seja n o número de modelos de tênis disponíveis em estoque para serem levados à feira, e m o número de modelos de tênis que serão levados. Um arquivo fornece, para cada par de modelo, a diferença (diversidade) entre eles. O cenário é modelado usando um grafo completo e ponderado, onde cada modelo de tênis é um vértice, e o peso da aresta que conecta um par de vértices define a diferença entre os modelos correspondentes. Esse grafo possui n vértices, e devemos selecionar um subconjunto com m deles, de modo que o somatório de pesos das arestas desse subconjunto seja o maior possível.

Para resolver esse problema, vamos assumir que a solução seja representada por um *array* binário S, onde uma posição i desse *array* indica se o modelo i de tênis será levado (valor 1) ou não (valor 0). Portanto, S tem tamanho n e deve conter m valores iguais a 1, com o restante igual a 0. Um primeiro algoritmo para esse problema constrói a solução passo a passo, selecionando um tênis por vez. O pseudocódigo

desse algoritmo é apresentado abaixo.

```
1 CONSTRUCAO():
2   S[i] = 0, para i = {1,2,...,n}
3   PARA i = 1 ATÉ m FAÇA
4     v = seleciona um modelo ainda não escolhido (i.e. S[v] é 0)
5     S[v] = 1
6   RETORNA S
```

O passo da linha 4 é o mais importante desse algoritmo. Temos duas opções: (1) selecionar um modelo ainda não escolhido aleatoriamente (neste caso, construímos uma solução aleatória); ou (2) usar alguma estratégia de seleção mais sofisticada. Como estratégia de seleção, podemos iniciar o algoritmo escolhendo o primeiro modelo aleatoriamente (primeira iteração do laço); nas iterações subsequentes, escolhemos o modelo mais diferente daqueles já escolhidos. Com isso, esperamos construir uma uma solução melhor.

Um segundo algoritmo para esse problema, baseado em busca local (ou refinamento), consiste em fazermos pequenas modificações na solução, buscando melhorá-la. O pseudocódigo desse algoritmo é apresentado abaixo.

```
1 REFINAMENTO(S):
2   ENQUANTO há melhoria na solução FAÇA
3     M = gera o conjunto de soluções modificadas
4     S = melhor solução de M (que seja melhor que S)
5   RETORNA S
```

Perceba que esse algoritmo recebe como parâmetro uma solução, que pode ser uma solução construída aleatoriamente ou usando a estratégia de escolha descrita acima. A modificação (usada para gerar o conjunto M da linha 3) consiste em remover um modelo escolhido da solução (atribuindo 0 à sua posição em S) e inserir um modelo não escolhido (atribuindo 1 à sua posição em S). O conjunto M é composto pelas soluções geradas a partir da aplicação dessa modificação a todos os pares de modelos $\{i,j\}$, tal que $S[i] \neq S[j]$.

Implemente esses algoritmos para decidir quais modelos levar à feira. Os arquivos contendo as informações (chamados de instâncias) estão disponíveis em https://www.uv.es/rmarti/paper/mdp.html. Use as instâncias *SOM-a* e *SOM-b*. Faça um relatório contendo os resultados da aplicação desses algoritmos na solução de cada instância, e procure responder às seguintes perguntas (lembre-se: a qualidade da solução é dada pelo somatório das diferenças entre todos os pares dos m modelos de tênis escolhidos):

1. Qual algoritmo tem o melhor desempenho?
2. Qual algoritmo executa mais rápido?
3. O refinamento iniciando de uma solução aleatória é melhor ou pior que o mesmo algoritmo iniciando de uma solução construída com a estratégia de escolha predefinida?
4. Quanto o refinamento consegue melhorar na qualidade da solução previamente construída?

Referências

[1] Manish Joshi and Theyazn Hassn Hadi. A review of network traffic analysis and prediction techniques. *arXiv preprint arXiv:1507.05722*, 2015.

[2] Carlos Andre Reis Pinheiro, Matthew Galati, Natalia Summerville, and Mark Lambrecht. Using network analysis and machine learning to identify virus spread trends in COVID-19. *Big Data Research*, 25:100242, 2021.

[3] Mark EJ Newman. The structure of scientific collaboration networks. *Proceedings of the national academy of sciences*, 98(2):404–409, 2001.

[4] Vincent Labatut and Xavier Bost. Extraction and analysis of fictional character networks: A survey. *ACM Computing Surveys (CSUR)*, 52(5):1–40, 2019.

[5] Melody Yu. Decoding the popularity of TV series: A network analysis perspective. *arXiv preprint arXiv:2307.05329*, 2023.

[6] Ana Lúcia Cetertich Bazzan. Similar yet different: the structure of social networks of characters in Seinfeld, Friends, How I Met Your Mother, and The Big Bang Theory. *Revista de Informática Teórica e Aplicada*, 27(2):66–80, 2020.