Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Curso de Graduação em Ciência da Computação

Trabalho Prático 1 - Algoritmos 1

Lucas Affonso Pires

Matrícula: 2023028420

Belo Horizonte, Novembro de 2024

1 Introdução

Como proposta do trabalho, foi selecionado a seguinte situação: Ajudar o imperador de Archadia, Gramis Gana Solidor, a resolver uma situação de ameaça dos reinos vizinhos, identificando qual a melhor capital para que o reino fique seguro, os locais ondem devem ser posicionados batalhões secundários vizando defender todas as cidades por conta de estradas que não retornam à capital, e identificar possíveis patrulhamentos à partir dos batalhões para evitar invasões no reino.

2 Modelagem

Para solucionar o problema, foram pensadas quais seriam as melhores maneiras de resolver cada parte do problema. Para encontrar a capital, é necessário um algoritmo capaz de identificar no grafo o local com o caminhamento mínimo, para encontrar os locais dos batalhões, temos que ser capazes de identificar as componentes conexas deste grafo, e, por fim, temos que ser capazes de identificar nos batalhões possíveis ciclos para que o patrulhamento seja feito.

A forma pela qual foi encontrada as melhores soluções possíveis para o problema foram por meio de 3 algoritmos vistos em aula. O algoritmo BFS, para encontrar a capital, o algoritmo de Kosaraju para encontrar as componentes conexas (cidades onde estarão os batalhões secundários) e o algoritmo DFS, utilizado como suporte anteriormente, e usado para identificar as possíveis rotas de patrulhamento. Cada um será especificado abaixo na parte de soluções.

3 Solução

- Algoritmo BFS:

  A busca em largura é utilizada para explorar o grafo a partir de diferentes níveis, onde visitamos todos os nós no nível atual antes de mover para o próximo nível. O algoritmo começa em um nó inicial e visita todos os nós diretamente conectados a ele, repetindo o processo para os nós descobertos, até que todos os nós alcançáveis sejam visitados. Com isso, podemos descobrir no grafo que temos como entrada, qual será a cidade com melhor localização possível para ser a capital.

  Segue o pseudocódigo: BFS(G, s): Crie uma fila F. Marque todos os nós como não visitados. Marque o nó s como visitado e adicione s à fila. Enquanto a fila não estiver vazia: u = Remover da fila. Para cada vizinho v de u em G: Se v não foi visitado: Marque v como visitado. Adicione v à fila.

- Algoritmo DFS:

O algoritmo DFS é utilizado para explorar o grafo mais profundamente a partir de um nó inicial. Ele segue o caminho até o final antes de voltar para explorar outros caminhos. A DFS é útil para detectar componentes conexos e analisar a estrutura de dependências do grafo (como as possíveis rotas de patrulhamento das componentes conexas, após essas serem identificadas pelo algoritmo de Kosaraju).

Segue o pseudocódigo: DFS(G, u): Marque o nó u como visitado. Para cada vizinho v de u em G: Se v não foi visitado: Chame DFS(G, v).

- Algoritmo de Kosaraju:

  Kosaraju é usado para encontrar componentes fortemente conectados no grafo (no caso, cada batalhão secundário necessário). Ele realiza dois passes sobre o grafo: o primeiro para ordenar os nós conforme a ordem de finalização de uma DFS, e o segundo para realizar uma DFS no grafo transposto, encontrando os componentes fortemente conectados.

  Segue o pseudocódigo: Kosaraju(G): 1. Realize uma DFS em G e armazene a ordem de término dos nós. 2. Transponha o grafo G (inverta todas as arestas). 3. Realize uma DFS em G transposto, seguindo a ordem de término dos nós. 4. Cada DFS em G transposto identifica um componente fortemente conectado.

4 Análise de Complexidade

A análise de complexidade do aplicativo foi relizada baseando-se no tempo e espaço utilizado por ele para cada tipo de implementação diferente (tanto a original quanto a modernizada). Adiante, serão analizadas as complexidades de tempo e espaço para ambas as versões do aplicativo:

- Algoritmo BSF:

  Analisaremos primeiro a complexidade de espaço. O algoritmo BFS necessita armazenar os nós que foram visitados e possíveis estruturas auxiliares (como pilhas ou filas), logo sua complexidade espacial é $O(V)$.

  Analisando agora a complexidade de tempo. Verificamos que O algoritmo BFS visita cada vértice e aresta uma única vez, portanto, sua complexidade temporal é $O(V+E)$, sendo V o número de vértices e E o número de arestas.

- Algoritmo DFS:

  Analisando a complexidade de espaço, o algoritmo DFS, assim como o BFS, necessita armazenar os nós que foram visitados e possíveis estruturas auxiliares (como pilhas ou filas), logo sua complexidade espacial é a mesma da BFS, sendo $O(V)$.

  Analisando agora a complexidade de tempo, o DFS também visita cada vértice e aresta uma única vez, logo, sua complexidade temporal é a mesma da BFS, sendo

$O(V + E)$.

- Algoritmo de Kosaraju:

  Analisaremos primeiro a complexidade de espaço. Perceba que para o algoritmo de Kosaraju, a complexidade é $O(V + E)$, pois precisamos armazenar o grafo original, o grafo transposto e as informações de visitação durante as DFS.

  Analisando agora a complexidade de tempo, Kosaraju realiza duas passagens de DFS e a transposição do grafo. Como cada DFS tem complexidade temporal de $O(V+E)$, o Kosaraju apresenta a mesma complexidade, já que multiplicar a complexidade por uma constante (no caso 2) a mantém igual.

5 Considerações finais

Quanto à experiência de realizar esse trabalho prático, as maiores dificuldades que eu encontrei ocorreram no desenvolvimento da parte-III, especialmente quando os grafos gerados apresentam arestas bidirecionais (que vão e voltam entre 2 vértices). Ao testar casos onde isso ocorria, o patrulhamento, mesmo quando devia existir, acabava por não ser identificado corretamente, fazendo com que diversas tentivas de correção surgissem, especialmente pois o patrulhamento funcionava de maneira perfeita no restante dos casos e a lógica parecia estar correta. No fim, após analisar as ajudas fornecidas no fórum e consultar alguns colegas, a solução para o problema finalmente surgiu.

Quanto às partes I e II, a utilização da BFS e do algoritmo de Kosaraju fizeram com que a implementação ocorresse mais facilmente, visto que ao utilizar como base os slides da matéria, achar a capital e os batalhões secundários foi relativamente fácil em comparação aos patrulhamentos.

Acredito que no desenvolvimento desse Trabalho Prático, fui capaz de entender bastante em relação ao uso de cada algoritmo utilizado, percebendo como cada um é útil para resolver certos casos, tornando a resolução mais simples e eficaz. No geral, acredito que foi um excelente exercício para entender o que foi ensinado em sala, e, aplicar de maneira prática.

6 Referências

`https://drive.google.com/file/d/1XKw1F9Tp-2p4K8jGOs7JGOQ155FWag9v/view?usp=sharing` (Livro do Cormen)

`https://drive.google.com/file/d/1voIWA4wS5uVhqavKBbau-71C0cVtuomi/view?usp=sharing` (Livro do Kleinberg)

Slides da matéria de Algoritmos I