Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

Projeto e Análise de Algoritmos

Trabalho Prático - Grafos

Leandro Augusto Lacerda Campos
Professores: Sebastián A. Urrutia e Vinícius F. dos Santos

Belo Horizonte
23 de maio de 2019

Sumário

1. Introdução

Como membros de uma das duas raças que estão em guerra por toda a galáxia, nós recebemos a missão de desenvolver um programa de computador capaz de quantificar e classificar todas as naves da frota inimiga e estimar o nosso tempo de vantagem para a realização de um ataque surpresa. Este programa terá como entrada uma massa de dados relacionados à estrutura do sistema de teleportes e aos estados de ocupação inicial e planejado de cada embarcação inimiga.

A solução que iremos apresentar neste relatório está baseada na teoria dos grafos, adotando [Diestel 2017] e [Cormen et al. 2009] como referências neste assunto. Com efeito, podemos modelar a frota inimiga como um grafo $G = (V, E)$ simples e não direcionado, no qual cada vértice $v \in V$ representa um posto de combate de alguma de suas naves e cada aresta $(u, v) \in E$ simboliza um teleporte possível entre os postos de combate retratados por $u, v \in V$.

Para implementar G, optamos pelas listas de adjacências. De fato, se $S \subset V$ representa os postos de combate de uma nave de reconhecimento, uma fragata ou um transportador, então o subgrafo $G' = G[S]$ induzido por S em G será esparso, pois teremos $|S|-1 \leq E(G') \leq |S|$. Além disso, considerando os limites $10 \leq |V| \leq 10^5$ e $8 \leq |E| \leq 10^6$ informados pelo quartel-general, o grafo G pode assumir um tamanho razoavelmente grande. Logo, as listas de adjacência são uma escolha apropriada. E esta escolha implica que a quantidade de memória que a presente solução requer é $\Theta(V + E)$.

2. Classificação e contagem das naves

Nossa primeira tarefa consiste em classificar e contar as naves da frota inimiga segundo o seu tipo. Dado um grafo $G = (V, E)$ representando esta frota, queremos encontrar uma partição $R_1, ..., R_{n_r}, F_1, ..., F_{n_f}, B_1, ..., B_{n_b}, T_1, ..., T_{n_t}$ do conjunto V de vértices tal que $R_1, ..., R_{n_r}$ representam suas n_r naves de reconhecimento; $F_1, ..., F_{n_f}$ suas n_f fragatas; $B_1, ..., B_{n_b}$ seus n_b bombardeiros; $T_1, ..., T_{n_t}$ seus n_t transportadores; e a soma $n_r + n_f + n_b + n_t$ resulta na quantidade total de suas naves.

Para identificar e classificar uma nave em G, precisamos antes estabelecer o que caracteriza uma nave e o que diferencia um tipo de nave do outro.

Lema 2.1. *Se um subconjunto não-vazio $S \subset V$ representa uma nave de reconhecimento então S induz um subgrafo linear em G.*

Demonstração. Definiremos, indutivamente, um subgrafo linear G' em G. Tome como v_1 um vértice de S que representa um posto de combate situado em uma das extremidades da nave. Suponhamos escolhidos $v_1, ..., v_k$ em S, com $k < |S|$, de modo que $\{v_1, ..., v_k\}$ induz um subgrafo linear em G. Em seguida, considerando as características principais de uma nave de reconhecimento e notando que v_k não está na outra de suas extremidades, pomos como v_{k+1} o único vértice de $S - \{v_1, ..., v_k\}$ que é adjacente a v_k. Isto completa a definição de G'. □

Lema 2.2. *Se um subconjunto não-vazio $S \subset V$ representa uma fragata então $T = G[S]$ é uma árvore e $\triangle(T) > 2$.*

Demonstração. Segue diretamente das características principais de uma fragata que T é conexo e tem $|S|-1$ arestas. Portanto, T é uma árvore em G. Agora suponha, por absurdo, que não existe $v \in S$ tal que $d(v) > 2$. Podemos concluir que T é, por definição, um subgrafo linear em G. Mas isto contradiz a observação feita no enunciado do problema de que uma fragata nunca poderá ter a mesma estrutura interna que uma nave de reconhecimento. Logo T contém pelo menos um vértice com grau maior do que 2, donde $\triangle(T) > 2$. □

Lema 2.3. *Se um subconjunto não-vazio $S \subset V$ representa um bombardeiro então $G' = G[S]$ é bipartido completo e $|E(G')| > |S|$.*

Demonstração. Sejam $v_0 \in S$ um vértice qualquer escolhido arbitrariamente, $S_1 \subset S$ o subconjunto de vértices que representam os postos de combate que estão na mesma fileira daquele representado por v_0 e $S_2 = S - S_1$. Segue diretamente das características principais de um bombardeiro que $u, v \in S_1$ ou $u, v \in S_2$ implica $(u, v) \notin E(G')$. Logo G' é bipartido. E como $(u, v) \in S_1 \times S_2$ implica $(u, v) \in E(G')$, temos que G' é bipartido completo. Por fim, o enunciado do problema nos assegura que $|S_1|, |S_2| \geq 2$ e $|S_1|+|S_2| \geq 5$. Sem perda de generalidade, podemos supor $|S_1| \geq |S_2|$. Então, quaisquer que sejam $u \in S_1$ e $v \in S_2$, temos $d(u) \geq 2$ e $d(v) \geq 3$, donde

$$|E(G')| = \frac{1}{2}\sum_{v \in S} d(v) = \frac{1}{2}\sum_{v \in S_1} d(v) + \frac{1}{2}\sum_{v \in S_2} d(v) \geq |S_1| + \frac{3}{2}|S_2| > |S|. \quad (1)$$

□

Lema 2.4. *Se um subconjunto não-vazio $S \subset V$ representa um transportador em G então S induz um grafo circular em G.*

Demonstração. Tome como v_1 um vértice de S que representa um posto de combate situado em uma das extremidades da nave. Considerando as características principais de um transportador e utilizando um procedimento análogo ao da demonstração do Lema 2.1, podemos definir por indução um conjunto $Q = \{v_1, ..., v_{|S|-1}\} \subset S$ que induz um subgrafo linear em G. Note que $S - Q = \{v\}$, onde v é o vértice que representa o posto de combate mais próximo daquele representado por v_1, mas que está localizado na outra fileira. Portanto, v é adjacente a v_1 e a $v_{|S|-1}$. Segue daí que o conjunto $S = Q \cup \{v\}$ induz um grafo circular em G. □

Lema 2.5. *O subconjunto não-vazio $S \subset V$ representa uma nave em G se, e somente se, $G[S]$ é uma componente de G.*

Demonstração. Se S representa uma nave em G, então os lemas 2.1 a 2.4 nos asseguram que $G[S]$ é um subgrafo conexo. Como não existe teleporte possível entre postos de combate dentro e fora de uma dada nave da frota inimiga, temos que $G[S]$ é um subgrafo conexo maximal. Logo, por definição, $G[S]$ é uma componente de G.

Por outro lado, sejam G' uma componente de G e $S = V(G')$. Por definição, temos que $S \neq \emptyset$. Suponha, por absurdo, que S não representa uma nave. Então existe $S' \subset V$ tal que S' representa uma nave, $S \subsetneq S'$ e $G[S']$ tem duas ou mais componentes. Ora, isto contradiz os lemas 2.1 a 2.4. Portanto, S é uma nave. □

O algoritmo NAVES-DA-FROTA, proposto para identificar e classificar as naves da frota inimiga, decorre naturalmente dos lemas desta seção.

```
NAVES-DA-FROTA(G)
 1  chama DFS(G) para identificar as componentes de G
 2  quando identificada, cada componente é inserida em L
 3  para cada componente C em L
 4      se C.nARetorno == 0
 5          se C.grauMax == 2
 6              C.tipo = RECONHECIMENTO
 7          senão C.tipo = FRAGATA
 8      senão se C.nARetorno == 1
 9          C.tipo = TRANSPORTADOR
10      senão C.tipo = BOMBARDEIRO
11  retorna L
```

Considerando que o grafo G é representado por listas de adjacências e L é uma lista encadeada, afirmamos que a complexidade de tempo do algoritmo NAVES-DA-FROTA é $O(V + E)$. Com efeito, executar uma busca em profundidade para identificar as componentes de G e obter, para cada uma delas, a contagem das arestas de retorno e o grau máximo consome $O(V + E)$ de tempo. Fazer inserções no início de L e depois percorrê-la para acessar cada componente têm custo total $O(V)$. Por fim, como o corpo do laço é executado $O(V)$ vezes e cada execução custa $O(1)$, a classificação das componentes tem custo $O(V)$.

Teorema 2.6. NAVES-DA-FROTA *identifica e classifica corretamente todas as naves da frota inimiga.*

Demonstração. Segue diretamente do Lema 2.5 que existe uma bijeção entre L e o conjunto de naves da frota inimiga, o que demonstra a corretude da identificação.

Seja C uma componente qualquer em L. Sabemos que o conjunto não-vazio $V(C)$ representa uma nave que pertence a um, e a somente um, dos seguintes grupos: (a) naves de reconhecimento; (b) fragatas; (c) bombardeiros; e (d) transportadores.

Para começar, suponha que $V(C)$ representa uma nave de reconhecimento. Pelo Lema 2.1, a componente C é um grafo linear e, portanto, $C.nARetorno = 0$ e $\triangle(C) = 2$. Se, por outro lado, $V(C)$ representa uma fragata, então, de acordo com o Lema 2.2, a componente C é uma árvore com $C.nARetorno = 0$ e $\triangle(C) > 2$.

Agora suponha que $V(C)$ representa um bombardeiro. Pelo Lema 2.3, a componente C é um grafo bipartido completo com $|E(C)| > |V(C)|$, donde $C.nARetorno = |E(C)| - |V(C)| + 1 > |V(C)| - |V(C)| + 1 = 1$. Por fim, resta o caso em que $V(C)$ representa um transportador. Ora, pelo Lema 2.4, sabemos que C é um grafo circular, com $C.nARetorno = 1$.

Em quaisquer dos quatro casos, a nave representada por $V(C)$ é corretamente classificada. Assim, demonstramos também a corretude da classificação. □

Para concluir nossa primeira tarefa, ainda falta contar as naves por tipo. Mas isto é bastante simples: basta percorrer a lista L retornada por NAVES-DA-FROTA(G), ao custo $O(V)$ de tempo, verificando o tipo de cada nave e incrementando o contador correspondente.

3. Cálculo do tempo de vantagem

Nossa segunda tarefa compreende calcular o tempo que a nossa frota dispõe até que alguma nave inimiga esteja pronta para combate, chamado de tempo de vantagem.

Sejam $G = (V, E)$ o grafo simples e não direcionado que modela a frota inimiga e $S \subset V$ o conjunto não-vazio que representa alguma de suas naves. Definamos agora, em função de S, outro grafo simples e não direcionado, denotado por H_S, que tem como vértices o conjunto $T_S \leftrightarrow S$ das funções bijetivas entre o conjunto T_S dos tripulantes da nave e o conjunto S dos seus postos de combate. Desta forma, cada vértice de H_S é uma bijeção $\Sigma_S \in T_S \leftrightarrow S$ que caracteriza um estado de ocupação válido da nave. Note que $|V(H_S)| = |S|!$ e, por isso, podemos enumerar os vértices de H_S.

As arestas de H_S, por sua vez, são determinadas pela seguinte regra: o par $\left(\Sigma_S^{(i)}, \Sigma_S^{(j)}\right) \in (T_S \leftrightarrow S)^2$ é uma aresta de H_S se, e somente, existe um único par $(u, v) \in E(G[S])$ tal que $\Sigma_S^{(i)}(u) = \Sigma_S^{(j)}(v)$, $\Sigma_S^{(i)}(v) = \Sigma_S^{(j)}(u)$ e $\Sigma_S^{(i)}(w) = \Sigma_S^{(j)}(w)$ para todo $w \in S - \{u, v\}$. Ou seja, para que os estados $\Sigma_S^{(i)}$ e $\Sigma_S^{(j)}$ sejam adjacentes em H_S, é necessário e suficiente que seja possível alternar entre estes estados utilizando um único teleporte.

Assim, o tempo de vantagem é obtido por $\min_S d_{H_S}\left(\Sigma_S^{(0)}, \Sigma_S^{(P)}\right)$, onde $\Sigma_S^{(0)}$ e $\Sigma_S^{(P)}$ são os estados inicial e planejado da nave representada pelo subconjunto não-vazio $S \subset V$ e $d_{H_S}\left(\Sigma_S^{(0)}, \Sigma_S^{(P)}\right)$ é o comprimento de um caminho mínimo entre estes estados no grafo H_S. Note que $d_{H_S}\left(\Sigma_S^{(0)}, \Sigma_S^{(P)}\right)$ mede o tempo mínimo necessário para que a nave representada por S esteja com todos os seus tripulantes em suas posições de combate corretas. Com efeito, cada aresta de um caminho mínimo entre $\Sigma_S^{(0)}$ e $\Sigma_S^{(P)}$ representa um único teleporte, que demora exatamente uma unidade de tempo para ser realizado. Além disso, uma nave pode realizar apenas um teleporte por vez.

Também veja que $\Sigma_S^{(P)}$ é informado implicitamente por meio do conjunto $\mathcal{M}_S$ de movimentações que é parâmetro do problema. De fato, qualquer transição entre estados de ocupação da nave representada por S pode ser completamente definida por um conjunto $\mathcal{M}_S$ de movimentações que satisfaça as seguintes condições: (a) para todo $u \in S$, existe um único par $(x, y) \in \mathcal{M}_S$ tal que $x = u$; e (b) para todo $v \in S$, existe um único par $(x, y) \in \mathcal{M}_S$ tal que $y = v$.

Por ser um problema de difícil solução, não é necessário calcular precisamente o tempo de vantagem. A missão em tela admite como solução uma cota inferior não-trivial para este valor. O Teorema 3.1 viabiliza a obtenção desta cota sem que seja obrigatória a construção de outro grafo além de G.

Teorema 3.1. *Sejam $S \subset V$ o subconjunto não-vazio que representa uma nave inimiga e $\mathcal{M}_S$ o conjunto de movimentações que descreve o estado de ocupação planejado $\Sigma_S^{(P)}$*

desta nave a partir do seu estado inicial $\Sigma_S^{(0)}$. Então

$$0 \leq \frac{1}{2} d_G(\mathcal{M}_S) \leq d_{H_S}\left(\Sigma_S^{(0)}, \Sigma_S^{(P)}\right), \tag{2}$$

onde $d_G : V^2 \to \mathbb{R}$ é a função que associa, a cada par $(u, v) \in V^2$, o comprimento $d_G(u, v)$ de um caminho mínimo entre os vértices u e v no grafo G.

Demonstração. A primeira desigualdade é óbvia. Por definição, a função d_G não pode assumir valores negativos. Além disso, se existe $(u, v) \in \mathcal{M}_S$ tal que $u \neq v$, então $\frac{1}{2} d_G(u, v) > 0$ e, portanto, $\frac{1}{2} d_G(\mathcal{M}_S) = \frac{1}{2} \sum_{e \in \mathcal{M}_S} d_G(e) > 0$.

Seja $P = \left(v_0 = \Sigma_S^{(0)}, v_1, ..., v_k = \Sigma_S^{(P)}\right)$ um caminho mínimo de comprimento k entre os estados $\Sigma_S^{(0)}$ e $\Sigma_S^{(P)}$ no grafo H_S. Usaremos indução em k para provar a segunda desigualdade. Para $k = 0$, temos $\Sigma_S^{(0)} = \Sigma_S^{(P)}$, o que implica $\mathcal{M}_S = \{(u, u); u \in S\}$. Segue daí que $0 = \frac{1}{2} d_G(\mathcal{M}_S) \leq d_{H_S}\left(\Sigma_S^{(0)}, \Sigma_S^{(P)}\right) = 0$. Para $k = 1$, temos $\left(\Sigma_S^{(0)}, \Sigma_S^{(P)}\right) \in E(H_S)$, donde $\mathcal{M}_S = \{(u, v), (v, u)\} \cup \{(w, w); w \in S - \{u, v\}\}$. Logo,

$$1 = \frac{2}{2} = \frac{1}{2} d_G(\mathcal{M}_S) \leq d_{H_S}\left(\Sigma_S^{(0)}, \Sigma_S^{(P)}\right) = 1 \tag{3}$$

Agora, para um certo $k > 0$, suponhamos que a segunda desigualdade seja válida para $k-1$. Seja $\mathcal{M}_S^*$ o conjunto de movimentações que descreve v_{k-1} a partir de $\Sigma_S^{(0)}$. Por serem adjacentes, v_{k-1} e $\Sigma_S^{(P)}$ se diferenciam por apenas um teleporte. Seja $(u, v) \in E(G)$ este teleporte. Logo, $\mathcal{M}_S - \mathcal{M}_S^* = \{(u', u), (v', v)\}$ e $\mathcal{M}_S^* - \mathcal{M}_S = \{(u', v), (v', u)\}$. Segue daí e da desigualdade triangular que

$$\begin{aligned} d_G(\mathcal{M}_S) &= d_G(\mathcal{M}_S^*) + d_G(u', u) + d_G(v', v) - d_G(u', v) - d_G(v', u) \\ &\leq d_G(\mathcal{M}_S^*) + 2 d_G(u, v) \\ &= d_G(\mathcal{M}_S^*) + 2. \end{aligned} \tag{4}$$

Por fim, aplicando a hipótese de indução e o fato de que $\left(v_{k-1}, \Sigma_S^{(P)}\right) \in E(H_S)$ implica $d_{H_S}\left(\Sigma_S^{(0)}, v_{k-1}\right) = d_{H_S}\left(\Sigma_S^{(0)}, \Sigma_S^{(P)}\right) - 1$, temos $d_G(\mathcal{M}_S) \leq d_G(\mathcal{M}_S^*) + 2 \leq 2 d_{H_S}\left(\Sigma_S^{(0)}, v_{k-1}\right) + 2 = 2 d_{H_S}\left(\Sigma_S^{(0)}, \Sigma_S^{(P)}\right)$, donde $\frac{1}{2} d_G(\mathcal{M}_S) \leq d_{H_S}\left(\Sigma_S^{(0)}, \Sigma_S^{(P)}\right)$. Isto conclui a demonstração. □

Corolário 3.1.1. *O valor $\min_S \frac{1}{2} d_G(\mathcal{M}_S)$ é uma cota inferior não-trivial para o tempo de vantagem.*

Tendo em vista o Corolário 3.1.1, a tarefa em discussão se resume a computar distâncias no grafo G. Dados $u, v \in V$, chamaremos de distância entre u e v, e denotaremos por $d_G(u, v)$, o comprimento de um caminho mínimo em G que tem origem em u e término em v (ou o contrário, uma vez que G é não direcionado).

Seja $S \subset V$ o subconjunto não-vazio que representa alguma nave inimiga. Se ela for uma nave de reconhecimento ou uma fragata, os lemas 2.1 e 2.2 nos asseguram que o subgrafo $G' = G[S]$ induzido por S é uma árvore. Sendo assim, podemos escolher um vértice $r \in S$ qualquer para enraizar G' e utilizar os algoritmos de decomposição SQRT e de ancestral comum mais baixo (LCA) para computar distâncias da seguinte forma: dados $u, v \in S$, temos $d_G(u, v) = d_G(r, u) + d_G(r, v) - 2d_G(r, w)$, onde w é o ancentral comum mais baixo de u e v. Para maiores informações sobre estes dois algoritmos e sobre como usar LCA no cálculo de distâncias, consulte [Laaksonen 2017].

Calcular $d_G(\mathcal{M}_S)$ usando decomposição SQRT e ancestral comum mais baixo possui complexidade de tempo igual a $O\left(S\sqrt{S}\right)$. Com efeito, executamos uma versão modificada do algoritmo DFS, com custo $O(S)$, para pré-computar a distância de cada vértice de S em relação à raiz r e para fazer a decomposição da árvore. Depois, para cada par $(u, v) \in \mathcal{M}_S$, encontramos seu LCA e calculamos $d_G(\mathcal{M}_S)$ com custo $O\left(\sqrt{S}\right)$ e $O(1)$, respectivamente. Como $|\mathcal{M}_S| = |S|$, as operações com os elementos de $\mathcal{M}_S$ têm custo total $O\left(S\sqrt{S}\right)$. Logo, considerando este último custo e o da etapa de preparação, obtemos a complexidade de tempo informada.

Se, por outro lado, a nave representada por S for um bombardeiro, então podemos utilizar o Teorema 3.2 para computar distâncias entre postos de combate desta embarcação.

Teorema 3.2. *Seja $S \subset V$ o subconjunto não-vazio que representa um bombardeiro. Então*

$$d_G(u, v) = \begin{cases} 0 & \text{se } u = v \\ 1 & \text{se } (u, v) \in E \\ 2 & \text{se } (u, v) \notin E \end{cases} \tag{5}$$

quaisquer que sejam $u, v \in S$.

Demonstração. O resultado é óbvio quando $u = v$. Então vamos considerar apenas o caso em que $u \neq v$. Tendo em vista o Lema 2.3, seja $G' = G[S_1 + S_2]$ o subgrafo bipartido completo que é induzido por S em G. Para fixar ideias, suponha $u \in S_1$. Se $v \in S_2$, então $(u, v) \in E$ e, portanto, $d_G(u, v) = 1$. Do contrário, se $v \in S_1$, basta notar que existe $w \in S_2$ tal que $(u, w), (w, v) \in E$. Logo, $d_G(u, v) = 2$. □

Obter $d_G(\mathcal{M}_S)$ usando o Teorema 3.2 possui complexidade de tempo igual a $O(S)$. De fato, na chamada DFS(G) do algoritmo NAVES-DA-FROTA, obtemos e armazenamos, para cada um dos vértices de S, a informação sobre a qual conjunto da partição (S_1, S_2) ele pertence. Assim, para cada par $(u, v) \in \mathcal{M}_S$, o custo de verificar se u e v estão no mesmo conjunto da partição é $O(1)$. Como $|\mathcal{M}_S| = |S|$, as verificações têm custo total $O(S)$, conforme afirmado.

Por fim, se a nave representada por S for um transportador, então podemos utilizar o Teorema 3.3 para computar distâncias entre seus postos de combate.

Teorema 3.3. *Seja $S \subset V$ o subconjunto não-vazio que representa um transportador com k postos de combate. Então existe um caminho simples $P = (v_1, v_2, ..., v_i, ..., v_j, ..., v_k)$*

em G, com $v_i \in S$ para todo $i = 1, ..., k$, tal que $d_G(v_i, v_j) = \min(j - i, k - j + i)$ quaisquer que sejam $1 \leq i \leq j \leq k$.

Demonstração. Pelo Lema 2.4, $G[S]$ é um subgrafo circular em G. Portanto, podemos enumerar $S = \{v_1, v_2, ..., v_i, ..., v_j, ..., v_k\}$ de modo a garantir que as sequências $P = (v_1, v_2, ..., v_i, ..., v_j, ..., v_k)$ e $Q = (u_1 = v_i, v_{i-1}, ..., v_1, ..., u_r = v_j, ..., u_k = v_{i+1})$ sejam caminhos simples em G. Segue daí e do Lema 2.5 que, para todo par de inteiros (i, j) tal que $1 \leq i \leq j \leq k$, existem apenas dois caminhos possíveis entre v_i e v_j em G: o subcaminho P_{ij} e o subcaminho Q_{1r}.

O comprimento de P_{ij} é dado por $j - i$. E o comprimento de Q_{1r} é dado por $(i-1)+(k-j+1) = k-j+i$. Logo, por definição, $d_G(v_i, v_j) = \min(j-i, k-j+i)$. □

Avaliar $d_G(\mathcal{M}_S)$ usando o Teorema 3.3 possui complexidade de tempo igual a $O(S)$. Na verdade, em decorrência do Lema 2.4, nós podemos enumerar S usando a árvore de busca em profundidade construída na chamada $\text{DFS}(G)$ do algoritmo NAVES-DA-FROTA. Desta forma, calcular $d_G(v_i, v_j)$ custa $O(1)$. Como precisamos fazer esse cálculo $|\mathcal{M}_S| = |S|$ vezes, obtemos o custo total $O(S)$, conforme foi dito.

Uma vez que sabemos determinar $d_G(\mathcal{M}_S)$ qualquer que seja o tipo da nave representada por S, torna-se fácil computar $\min_S \frac{1}{2} d_G(\mathcal{M}_S)$. Desta forma, optamos por não apresentá-lo nem provar a sua corretude neste relatório. Finalmente, tendo em vista as análises de complexidade de tempo que nós fizemos acima, podemos afirmar que, no pior caso, o custo de obter esta cota inferior não-trivial para o tempo de vantagem é $O\left(V\sqrt{V}\right)$.

4. Testes e resultados empíricos

Para comparar resultados práticos com os téoricos apresentados, bem como medir o tempo de execução e o consumo de memória em casos de teste que respeitem os limites passados pelo quartel-general, foi realizada uma bateria de testes com o programa de computador desenvolvido para implementar a solução proposta. A máquina na qual os testes foram realizados tem a seguinte configuração:

- Processador: Intel Core i7-4720 2.6 GHz
- Memória: 16 GB DDR3L SDRAM 800 MHz
- Sistema operacional: Ubuntu 18.10
- Compilador: GCC 8.3.0

As tabelas 1(a) e 1(b) apresentam o tempo de execução e o consumo de memória primária para os casos de teste fornecidos pelo quartel-general. O caso de teste 9 foi o que apresentou os maiores valores nos dois indicadores. Em relação ao seu tempo de execução, constamos que 41,1% está relacionado às operações de leitura dos dados de entrada; 6,9% à construção do grafo; 51,6% aos algoritmos de identificação, classificação e contagem das naves; e apenas 0,3% ao cálculo da cota inferior não-trival do tempo de vantagem.

Também geramos alguns casos de teste relacionados a naves de reconhecimento e a bombardeiros. Note que as naves de reconhecimento, as fragatas e os transportadores

Tabela 1. Resultados obtidos nos casos de teste oficiais

(a)

	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Caso 6
Tempo (ms)	3	2	4	11	13	32
Memória (MB)	4,08	4,08	4,08	4,39	5,44	7,50

(b)

	Caso 7	Caso 8	Caso 9	Caso 10	PDF1	PDF2
Tempo (ms)	38	37	323	60	2	1
Memória (MB)	8,77	7,99	35,99	11,59	4,08	4,08

têm complexidades de tempo e espaciais similares. Por sua vez, os bombardeiros estão relacionados a subgrafos densos em G, o que gera impacto negativo no tempo de execução das operações envolvidas na leitura dos dados de entrada, na construção do grafo G e na identificação e classificação das naves. Segue daí o motivo de selecionar apenas estes dois tipos de nave para uma análise experimental mais detalhada.

Os resultados desta análise são apresentados na Figura 1. Veja que o tempo de execução do programa tendo como entrada uma nave de reconhecimento com 10^6 postos de combate é praticamente o mesmo daquele que tem como entrada um bombardeiro com apenas $2 \cdot 10^3$ postos.

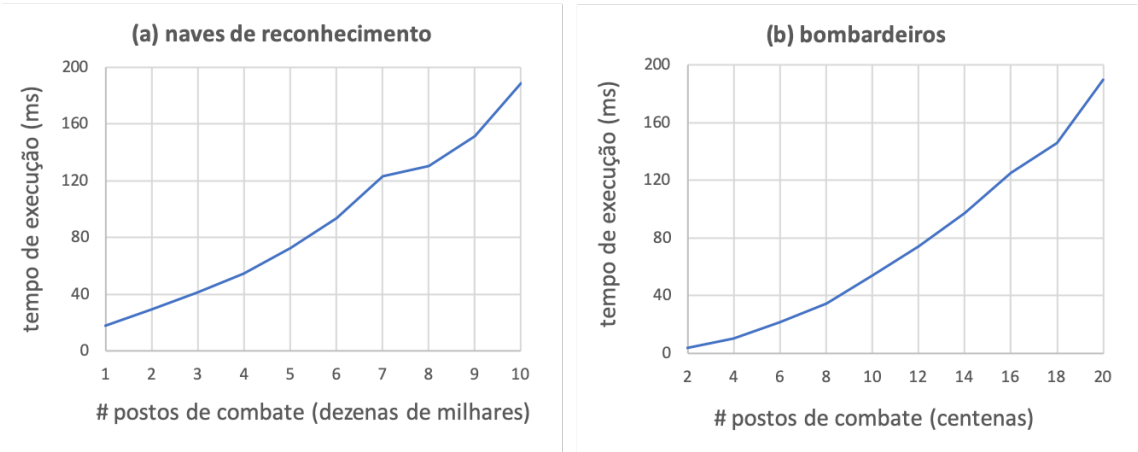

Figura 1. Tempo de execução do programa em função do tipo de nave

5. Conclusão

A partir dos testes e resultados empíricos obtidos, consideramos que a nossa missão foi concluída com sucesso. A partir da solução que nós propusemos e implementamos, a nossa frota poderá evitar novas derrotas em batalha por saber, antes do início de qualquer confronto, a quantidade e o tipo das naves inimigas e o tempo que elas precisam até estar prontas para combate. Este tempo, denominado tempo de vantagem, é uma das variáveis mais importantes na tomada de decisão sobre a realização ou não de um ataque surpresa.

Recomendamos, para futuros trabalhos, revisar as partes da implementação da solução que estão relacionadas às operações de leitura dos dados de entrada e aos algoritmos de identificação, classificação e contagem das naves. Conforme foi dito na Seção 4, estas duas partes representaram cerca de 92,7% do tempo de execução do caso de teste que, dentre aqueles fornecidos pelo quartel-general, obteve o pior desempenho. Em especial, sugerimos uma reavaliação da escolha e da implementação, até então balizadas mais pela simplicidade do que pela eficiência, das estruturas de dados utilizadas e da postura defensiva que adotamos em relação aos dados de entrada.

Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, 3rd Edition*. MIT Press.

Diestel, R. (2017). *Graph Theory, 5th Edition*, volume 173 of *Graduate texts in mathematics*. Springer.

Laaksonen, A. (2017). *Guide to Competitive Programming - Learning and Improving Algorithms Through Contests*. Undergraduate Topics in Computer Science. Springer.