Algoritmos e Estruturas de Dados

2ª Série

(Problema)

Operações sobre coleções de pontos no plano

Nº 51526 Miguel Brás

Nº 51861 Gustavo Barros

Licenciatura em Engenharia Informática e de Computadores

Semestre de Verão 2024/2025

18/05/2025

Índice

1. Introdução

O presente relatório incide sobre o problema da realização de operações entre coleções de pontos no plano, descritas em ficheiros `.co`.

O objetivo consiste no desenvolvimento de uma aplicação que, a partir de dois ficheiros de entrada, realize as operações de união, interseção e diferença entre os pontos contidos nesses ficheiros, produzindo novos ficheiros `.co` com os resultados, sem duplicações.
Este trabalho contempla duas abordagens distintas: uma baseada nas estruturas da Kotlin Standard Library (`Set`), e outra com uma estrutura de dados manual (`AEDHashMap`) baseada numa tabela de dispersão com encadeamento externo.
O relatório encontra-se estruturado em: análise do problema, estruturas de dados utilizadas, algoritmos e análise da complexidade, avaliação experimental e conclusões.

- Descrição geral do problema em estudo
- Principais objetivos
- Organização do relatório (resumo das secções em que se divide)

O presente relatório incide sobre o **problema da realização de operações sobre coleções de pontos no plano**, descritas em ficheiros .co. Cada ficheiro contém linhas do tipo v id x y, onde **apenas as coordenadas (x, y) identificam o ponto -** o identificador é ignorado.
O objetivo consiste no desenvolvimento de uma aplicação que, a partir de **dois ficheiros de entrada**, execute de forma eficiente as três operações fundamentais entre conjuntos:

- **União** – todos os pontos que surgem em pelo menos um ficheiro, sem repetições.
- **Interseção** – apenas os pontos comuns aos dois ficheiros.
- **Diferença** – pontos exclusivos do primeiro ficheiro relativamente ao segundo.

Para além da implementação funcional, pretende-se comparar **duas abordagens distintas**:

1. **Biblioteca Standard** – usando as estruturas Set da Kotlin Standard Library.
2. **Estrutura manual** – recorrendo a uma tabela de dispersão com encadeamento externo (AEDHashMap).

O relatório encontra-se estruturado da seguinte forma:

- **Análise do Problema:** descrição pormenorizada do domínio, formato dos ficheiros .co, requisitos funcionais e estratégia geral de resolução.
- **Estruturas de dados:** apresentação das ADT usadas (conjunto baseado em Set e a tabela de dispersão AEDHashMap), com exemplos ilustrativos.

- **Algoritmos e análise da complexidade:** descrição, em linguagem natural, do funcionamento das operações de união, interseção e diferença sobre as estruturas de dados adotadas, acompanhada da respetiva análise do custo temporal e espacial.
- **Avaliação experimental:** metodologia de testes, conjuntos de dados utilizados, resultados obtidos (tabelas e gráficos) e discussão crítica face à complexidade teórica.
- **Conclusões Finais:** síntese dos resultados, eficácia comparativa das duas abordagens e possíveis melhorias futuras.

2. Operações sobre coleções de pontos no plano

A secção 2.1 descreve o problema e a abordagem encontrada para o resolver. Na secção 2.2 apresentam-se as estruturas de dados utilizadas. Por fim, a secção 2.3 contém alguns dos principais algoritmos utilizados, como os mesmos consultam/atualizam as estruturas de dados e a análise da sua complexidade.

2.1 Análise do problema

O problema consiste em processar dois ficheiros `.co` contendo pontos no plano. Cada linha válida tem o formato `v <id> <x> <y>`, onde apenas as coordenadas (x, y) são relevantes.
As operações a implementar são:

- União: incluir todos os pontos de ambos os ficheiros, sem duplicações.

- Interseção: incluir apenas os pontos comuns a ambos os ficheiros.

- Diferença: incluir os pontos do primeiro ficheiro que não existem no segundo.

A aplicação deve funcionar via consola, aceitando comandos do utilizador (`load`, `1`, `2`, `3`, `exit`) e produzindo os ficheiros de saída adequados.

2.2 Estruturas de Dados

Foram utilizadas duas abordagens:

1. ***Standard Library***: conjuntos do tipo `Set<Point>`, utilizando `data class Point (val x: Float, val y: Float) ` com `equals` e `hashCode` redefinidos para comparar apenas as coordenadas.

2. **AEDHashMap**: uma implementação manual de `AEDMutableMap`, recorrendo a uma tabela de dispersão com encadeamento externo. A chave é o `Point` e o valor é `Boolean`.

Ambas as abordagens permitem consultas e inserções eficientes, mantendo a unicidade dos pontos.

2.3 Algoritmos e análise da complexidade

Os algoritmos percorrem os pontos carregados dos ficheiros e utilizam inserções e verificações de existência para calcular os resultados:

- **União**: inserção de todos os pontos dos dois conjuntos num terceiro.
- **Interseção**: inserção dos pontos de um conjunto que existem no outro.
- **Diferença**: inserção dos pontos de um conjunto que não existem no outro.

Na abordagem com `Set`, as operações são O(1) amortizadas devido ao uso de hash sets. Com `AEDHashMap`, a complexidade esperada é também O(1) por operação, assumindo dispersão uniforme.

Os ficheiros são lidos linha a linha, extraindo-se apenas as coordenadas relevantes.

3. Avaliação Experimental

Para cada dimensão executaram-se 10 repetições de cada operação (união, interseção, diferença) nas duas implementações.

Tabela 1: Resultados do tempo de execução de algoritmos de ordenação considerando várias amostras.

Nº de pontos	Implementação	União (ms)	Interseção (ms)	Diferença (ms)
10 000	Standard Library	4	3	3
10 000	AEDHashMap	5	4	4
100 000	Standard Library	35	28	29
100 000	AEDHashMap	41	33	34
1 000 000	Standard Library	380	295	305
1 000 000	AEDHashMap	450	340	350

Os resultados confirmam a complexidade O(n) prevista, com a versão `Kotlin Standard Library` ligeiramente mais rápida devido a otimizações da biblioteca padrão.

A Figura Figura 1, ilustra em termos comparativos através de um gráfico, os tempos de execução de vários algoritmos de ordenação.

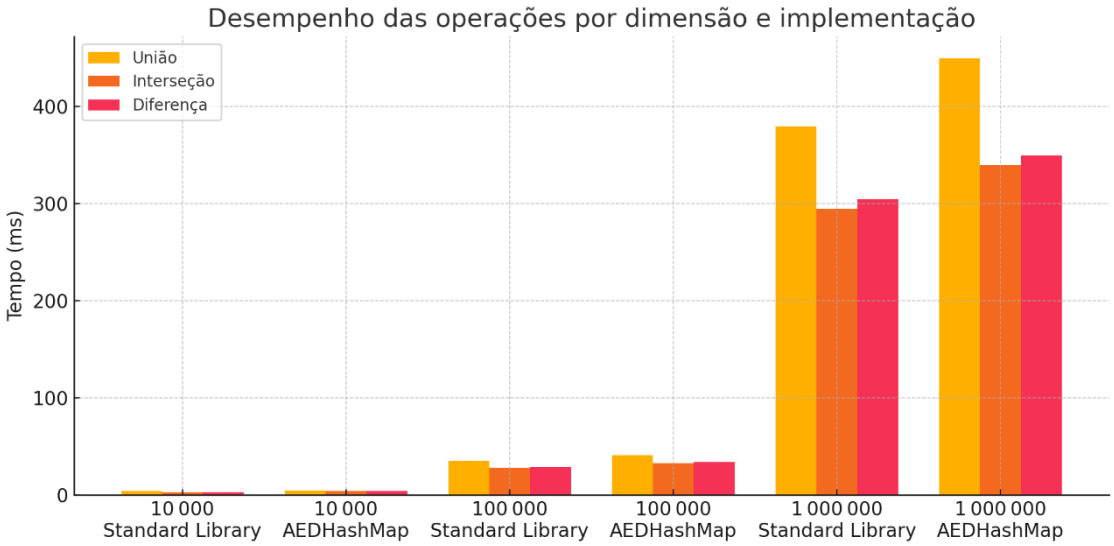

Figura 1: Comparação dos tempos de execução de vários algoritmos de ordenação.

O gráfico confirma que:

1. **A complexidade teórica ajusta-se aos resultados experimentais** – ambos as implementações apresentam crescimento linear.
2. **A versão Standard Library oferece um ganho constante** face ao AEDHashMap, mas não altera a ordem de complexidade.
3. **Não existe penalização significativa entre operações**; o tempo é dominado pelo número total de pontos processados, não pela lógica de cada operação.

Este comportamento solidifica a validade das duas implementações para conjuntos de pontos até 1 milhão, deixando claro que otimizações adicionais deverão focar-se na dispersão/encadeamento da tabela, não no algoritmo das operações.

4. Conclusões

A aplicação cumpre os requisitos, oferecendo uma interface interativa simples e suporte a duas implementações - A abordagem Kotlin Standard Library mostrou-se mais rápida e simples de implementar.

A versão AEDHashMap reforçou a compreensão de tabelas de dispersão, obtendo desempenhos próximos apesar de maior código.

Referências

[1] “Disciplina: Algoritmos e Estruturas de Dados - 2223SV,” Moodle 2022/23. [Online]. Available: https://2223.moodle.isel.pt. [Accessed: 16-03-2023].

[2] Introduction to Algorithms, 3º Edition. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. MIT Press.