Análise de Algoritmos

Trabalho I - Sistema de Votação

Marcelo Amaral
Luís Laguardia
Eduardo Adame
Tiago Barradas

Setembro de 2022

Sumário

Modelagem Geral do Programa

Nesse trabalho, tivemos que implementar um sistema de manejamento de votação com uma variedade de funcionalidades, tendo a oportunidade de aplicar na prática as estruturas e algoritmos vistos em aula. Para isso, tivemos que tomar decisões em grupo sobre qual seria a modelagem ideal para atacar os diversos desafios que nos foram apresentados. Baseando nossas escolhas principalmente no tempo de operação dos algoritmos e na quantidade de armazenamento utilizado, optamos por modelar o problema da seguinte maneira:

Roberto: É a classe principal, que aceita entrada, remoção e consulta de votos. É baseada em uma Hashtable de endereçamento direto, onde cada chave mapeia um objeto **Voto**. Quando um voto é removido, ele tem seu user_id definido como 0, e sua posição na Hashtable é guardada no Stack **Deleted**, uma estrutura simples feita para preencher primeiro as posições deletadas, e depois as posições vazias. Por ser uma Hashtable de endereçamento direto, inserção, consulta e remoção são O(1).

Além disso, essa classe também possui dois métodos de consulta de resultados: o **relatório de votos** e o **relatório de candidatos**.

O **relatório de votos** ordena os todos os votos registrados na Hashtable por ordem de submissão, usando uma implementação simples do Quicksort; logo, O(n log n). Como o momento de submissão é registrado na classe **Data** (de autoria do grupo), optamos por implementar um operator overload na classe e manter o algoritmo de ordenação simples e conciso. Essa função retorna um **Vetor** (também autoral), que pode ser filtrado por região, como especificado.

O **relatório de candidatos** computa os 10 candidatos mais votados segundo os registros atuais. Para isso, ele registra uma contagem de votos por candidato em tempo O(n) e seleciona os dez mais votados a partir de uma implementação do algoritmo de Quickselect - também O(n). Opcionalmente, o consultor também pode passar o período temporal a ser analisado, onde a etapa de contagem do algoritmo também verificará se o voto pertence ou não ao período temporal designado.

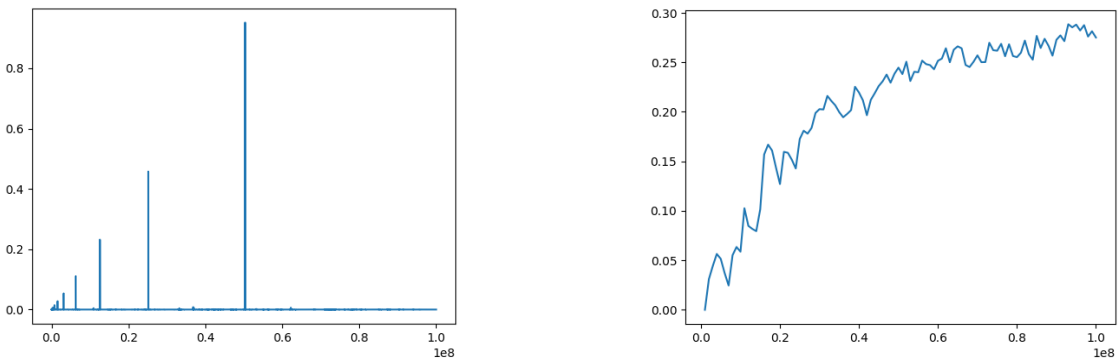
Voto: É a classe que armazena os dados do voto: ID do voto, ID do usuário, ID do candidato, data e região. Como essa é a classe mais utilizada nas aplicações do projeto (espera-se ser capaz de armazenar até 100 milhões de instâncias desse tipo), dedicamos esforços dobrados para otimizar o seu custo de memória. Tomando como base o livro de conteúdo aberto Optimizing C++/Introduction, conseguimos reduzir o tamanho de uma instância de voto para apenas 20 bytes.

Data: É a classe que armazena o momento em que o voto foi inserido. Possui os campos ano, mês, dia, hora, minuto, segundo e dia da semana, onde todo campo (com exceção do ano), é do tipo int8 (signed char), possibilitando que uma instância dessa classe ocupasse apenas 8 bytes.

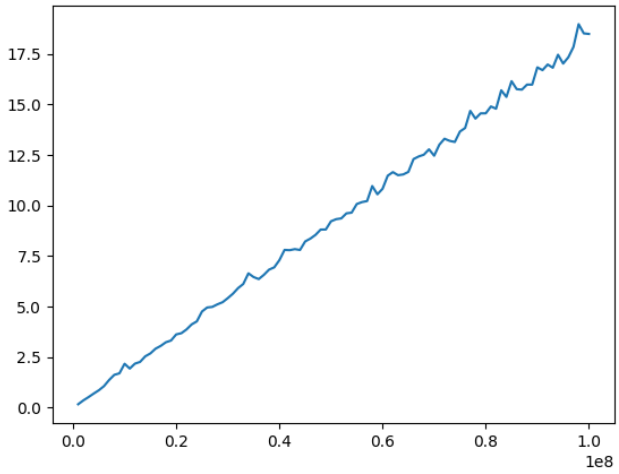
Além disso, como pré-mencionado, a data também possui overload dos operadores ‘<’ e ‘>’. Para tal, ela compara os seus atributos com outra classe de data na ordem do mais relevante (ano) para o menos relevante (segundo). Se alguma comparação for verdadeira, ela retorna true e interrompe as comparações. Se a comparação for falsa, e os atributos forem iguais, avança-se na granularidade da comparação. Se os atributos não forem iguais ou se não houver mais atributos para comparar, retorna false.

Discussão dos Resultados

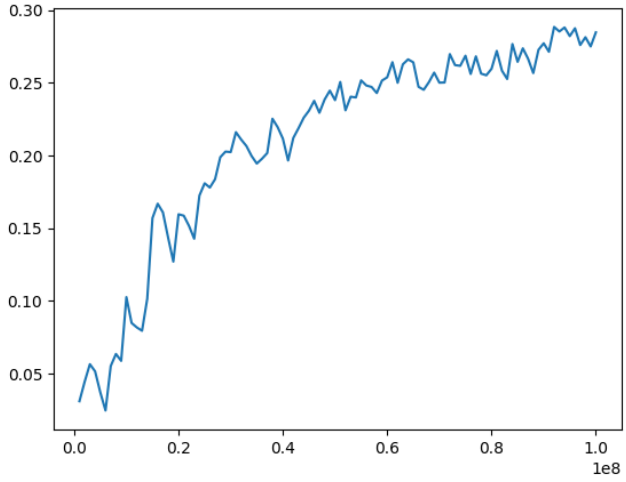
Acima, comentamos sobre o tempo de operação de três ações distintas; inserção / remoção / consulta, ordenação e seleção; e sobre seus tempos de operação esperados; O(1), O(n log n) e O(n). Abaixo, é possível conferir um gráfico da análise do tempo de execução real para cada uma das operações listadas e compará-las com uma função da família do limite descrito.



Visualização do tempo de inserção sobre duas óticas distintas: a primeira compara inserções individuais e torna bem perceptível o custo adicional de alocação e cópia dos dados para um novo espaço de memória contígua; a segunda mostra o tempo gasto para “inserção em lotes, reforçando a ideia de que, quando o tamanho do array é estável, as operações são feitas em tempo constante.



Tempo de ordenação em que as ordenações passadas não mudam a ordem dos elementos inplace, portanto não afetam ordenações subsequentes; não fica tão óbvio que a complexidade é $O(n \log n)$ devido ao modo como a biblioteca de plot organiza os eixos, mas uma comparação numérica deixa isso mais aparente.



Tempo para selecionar os 10 candidatos mais votados, em tese $O(n)$.

Referências

- Thiago Pinheiro de Araújo, Slides de aula, 2022
- Optimizing C++/Introduction