Seminário: Aha! Insights em Algoritmos

Baseado em *Programming Pearls*, Coluna 2

Guilherme Altoé Tomazini Kaiky Viglioni Tavares Moura
Marco Antônio Faustini Pessoa

1 Introdução: O Poder do Insight

O estudo de algoritmos muitas vezes é associado a estruturas complexas e cálculos avançados. No entanto, o "Aha! Insight"refere-se a soluções inesperadamente simples para problemas aparentemente difíceis. Este documento explora três problemas clássicos discutidos por Jon Bentley, onde a redefinição da perspectiva do problema permite soluções eficientes em tempo e espaço.

2 Problema A: A Busca pelo Inteiro Faltante

O Problema: Dado um arquivo sequencial contendo no máximo 4 bilhões de inteiros de 32 bits em ordem aleatória, encontre um inteiro de 32 bits que *não* esteja no arquivo.

- **Restrição 1 (Memória Ampla):** Se houver memória suficiente, a solução ideal é um **Bitmap** (Vetor de Bits), onde cada bit representa a presença de um número. São necessários 2^{32} bits ≈ 536 MB.
- **Restrição 2 (Memória Limitada):** Como resolver usando apenas alguns centenas de bytes e arquivos temporários?

O Aha! Insight (Busca Binária em Arquivo): O insight é aplicar a lógica da busca binária não sobre o índice, mas sobre a *contagem* de elementos em um intervalo.

1. Dividimos o intervalo de 0 a $2^{32} - 1$ em dois: $[0, 2^{31} - 1]$ e $[2^{31}, 2^{32} - 1]$.
2. Lemos o arquivo original e contamos quantos números caem em cada metade.
3. Como o total de números é menor que o espaço disponível (2^{32}), pelo menos uma metade terá menos números do que sua capacidade máxima.
4. Aplicamos a mesma lógica recursivamente na metade "vazia"até encontrarmos o número faltante.

Eficiência: O algoritmo lê o arquivo aproximadamente $\log_2(n)$ vezes, o que, para 32 bits, é cerca de 32 passagens.

3 Problema B: Rotação de Vetores

O Problema: Rotacionar um vetor de n elementos à esquerda por i posições. Exemplo: $abcdefgh$ com $i = 3$ torna-se $defghabc$.

Abordagens Comuns:

- Usar um vetor auxiliar (gasta $O(n)$ de espaço).
- Rotacionar um por um, i vezes (gasta $O(n \cdot i)$ de tempo).

O Aha! Insight (Algoritmo de Reversão): Podemos ver o vetor como a concatenação de duas partes ab, onde a tem os primeiros i elementos e b o restante. Queremos transformar ab em ba. A solução elegante utiliza a propriedade da reversão:

1. Reverta $a \to a^r b$ (ex: `cba|defgh`)
2. Reverta $b \to a^r b^r$ (ex: `cba|hgfed`)
3. Reverta o vetor inteiro $(a^r b^r)^r = ba$ (ex: `defghabc`)

Vantagem: O código é extremamente simples, opera *in-place* (espaço constante) e leva um tempo proporcional a n.

4 Problema C: Localização de Anagramas

O Problema: Dado um dicionário de palavras inglesas, encontre todos os conjuntos de anagramas (ex: "pots", "stop", "tops").

O Aha! Insight (Assinaturas e Ordenação): Comparar cada palavra com todas as outras seria $O(n^2)$, o que é inviável para dicionários grandes. O segredo é criar uma **assinatura canônica** para cada classe de anagramas.

1. **Assinatura:** Para cada palavra, ordene suas letras alfabeticamente. Tanto "pots"quanto "stop"recebem a assinatura "opst".
2. **Processamento:** Gere um arquivo onde cada linha contém a *assinatura* e a *palavra original*.
3. **Ordenação:** Ordene o arquivo pelas assinaturas. Palavras que são anagramas aparecerão em linhas adjacentes
4. **Saída:** Percorra o arquivo ordenado e agrupe as palavras com a mesma assinatura.

5 Conclusão e Princípios

Os problemas apresentados demonstram três pilares fundamentais da engenharia de software:

1. **Busca Binária Ubíqua:** Pode ser usada para encontrar erros em arquivos ou raízes de funções, não apenas em arrays ordenados.
2. **Poder das Primitivas:** Operações simples (como reverter um array) podem ser combinadas para resolver tarefas complexas (como rotação).
3. **Ordenação para Agrupamento:** Muitas vezes, ordenamos dados não para vê-los em ordem, mas para colocar itens idênticos lado a lado.

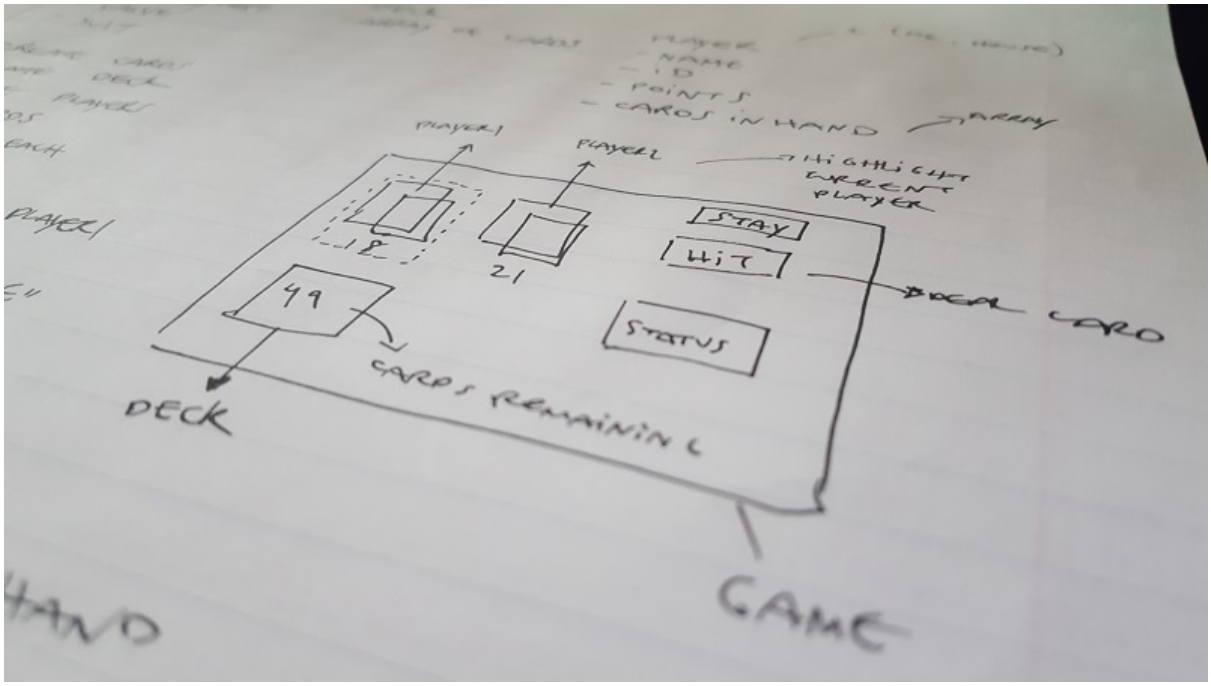

Figura 1: Representação de pensamento computacional no papel

Dessa forma, é possível compreender que, quando temos algum problema para resolver, é essencial parar e pensar antes de tomar qualquer atitude, porque, muitas vezes, a resposta pode ser mais simples do que aparenta.
O ideal é parar, pensar, se possível utilizar algum papel ou documento para organizar as ideias, conversar com outros colegas de time/trabalho (caso haja algum) e, aí sim, após organizar as ideias, partir para a prática.