

## MAC0338 - Análise de Algoritmos

Departamento de Ciência da Computação

Segundo semestre de 2023

### Lista 9

**Entregar:** Exercício 5.

**Instruções:** Leia as instruções postadas no e-disciplinas.

1. (**CLRS 17.1-2**) Mostre que se uma operação **Decrementa** for incluída nas operações de manipulação de um contador binário com  $k$  bits,  $n$  operações podem custar tempo  $\Theta(nk)$ .
2. (**CLRS 17.1-3**) Uma sequência de  $n$  operações é executada em uma estrutura de dados. A  $i$ -ésima operação custa  $i$  se  $i$  é uma potência de 2, e 1 caso contrário. Determine o tempo amortizado por operação.
3. (**CLRS 17.2-1**) Uma sequência de operações sobre uma pilha é executada numa pilha cujo tamanho nunca excede  $k$ . Depois de cada  $k$  operações, uma cópia da pilha toda é feita para propósito de *back-up*. Mostre que o custo de  $n$  operações sobre a pilha, incluindo a operação de cópia para *back-up*, é  $O(n)$ , atribuindo valores adequados de créditos a cada operação.
4. (**CLRS 17.2-3**) Suponha que desejamos não apenas incrementar um contador mas também algumas vezes reinicializá-lo com zero. Mostre como implementar um contador com um vetor binário de maneira que qualquer sequência de  $n$  operações **incrementa1** e **zera\_contador** consuma tempo  $O(n)$ , desde que o contador esteja inicialmente com zero. (**Dica:** Mantenha um apontador para o 1 mais significativo do contador.)
5. Suponha que desejemos que nossa tabela dinâmica também seja diminuída se sua ocupação diminui significativamente. Ou seja, queremos que, em uma remoção, caso a tabela fique “muito vazia”, seja alocado um novo vetor menor, e os elementos que estão atualmente na tabela grande sejam copiados para o vetor menor e o vetor grande seja desalocado. Considere o seguinte esquema. Seja  $n$  o número de elementos na tabela antes da operação. Quando adicionamos um elemento, se a tabela estiver com espaço vazio o custo é 1, se a tabela estiver cheia relaocaremos para uma tabela com o dobro do tamanho, o custo é  $n + 1$ . Quando removemos um elemento, se a tabela estiver com pelo menos um quarto da sua capacidade então só removemos o elemento com custo 1. Se a tabela estiver com menos de um quarto da sua capacidade, então realocaremos para uma tabela com metade do tamanho e o custo da operação é  $1 + (n - 1)$ , custo 1 para deletar e  $n - 1$  para copiar os elementos restantes. Mostre que essa tabela tem custo amortizado constante por operação.
6. Exercício 1.3 de <http://cs.nyu.edu/~yap/classes/funAlgo/05f/lect/16.pdf>.
7. Considere a implementação de lista ligada para representar conjuntos disjuntos. Sugira uma mudança simples da rotina UNION que não necessite do apontador **fim** para o último da lista de cada conjunto. Sua sugestão deve ser tal que, independente de estarmos ou não usando a heurística dos tamanhos (anexe no final a lista menor), o consumo assintótico de tempo de pior caso deve se manter igual.
8. Considere a implementação do union-find por árvores enraizadas. Escreva uma versão não recursiva do FINDSET com compressão de caminhos.
9. Considere a implementação do union-find por árvores enraizadas com compressão de caminhos e heurística dos ranks (a árvore de menor rank é pendurada na de menor rank no union). Considere uma sequência qualquer (válida) de  $m$  operações MAKESET, FINDSET e LINK em que todas as operações LINK aparecem antes das operações FINDSET. Mostre que tal sequência consome, no pior caso, tempo  $O(m)$ . O que acontece com o tempo consumido por uma sequência deste tipo se apenas compressão de caminhos estiver implementada?