MAC 338 - Análise de Algoritmos

Departamento de Ciência da Computação
Primeiro semestre de 2008

Lista 7

1. **(Exercício 17.1-1 do CLRS)** Se o conjunto de operações sobre a pilha incluisse uma operação **MultiPush**, que empilhasse k elementos na pilha, a delimitação de $O(1)$ para o custo amortizado de n operações continuaria válida?

2. **(Exercício 17.1-2 do CLRS)** Mostre que se uma operação **Decrementa** for incluída nas operações de manipulação de um contador binário com k bits, n operações poderiam custar tempo $\Theta(nk)$.

3. **(Exercício 17.1-3 do CLRS)** Uma seqüência 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.

4. **(Exercício 17.2-1 do CLRS)** Uma seqüência 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.

5. **(Exercício 17.2-3 do CLRS)** 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 seqüência 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.)

6. **(Exercício 17.3-6 do CLRS)** Mostre como implementar uma fila com duas pilhas de modo que o custo amortizado para cada **enfila** e **desenfila** seja $O(1)$.

7. **(Exercício 17.3-7 do CLRS)** Projete uma estrutura de dados que suporte as seguintes duas operações sobre um conjunto S de inteiros:

   (a) **insere**(S, x): insere x no conjunto S;

   (b) **remova_maior_metade**(S): remove os maiores $\lceil |S|/2 \rceil$ elementos de S.

   Explique como implementar essa estrutura de dados de maneira que n operações consumam tempo $O(m)$.

8. 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.

9. Mostre que $\lg(\lg^* n) = O(\lg^*(\lg n))$.

10. Considere a implementação do union-find por árvores enraizadas. Escreva uma versão não recursiva do FINDSET com compressão de caminhos.

11. 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 seqüência qualquer (válida) de m operações MAKESET, FINDSET e LINK em que todas as operações LINK aparecem antes das operações FINDSET. Mostre que tal seqüência consome, no pior caso, tempo $O(m)$. O que acontece com o tempo consumido por uma seqüência deste tipo se apenas compressão de caminhos estiver implementada?