MAC0338 - ANÁLISE DE ALGORITMOS

LISTA 9

FIZ: 4 / 9

CHECKLIST

entregue fiz incompleto não entendi CRLS

- análise amortizada

1

- tabela dinâmica

2

- análise amortizada

3 4

- tabela dinâmica

5

- análise amortizada

6

- union - find

7 8 9

Sabrina Araújo da Silva nUSP 12566182

MAC0338 - ANÁLISE DE ALGORITMOS

LISTA 9

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 $\Theta(nk)$.

uma operação Decrementa tem a seguinte estrutura

```
Decrementa (A, K)
    i ← 0
    enquanto i < K e A[i] = 0 faça
        A[i] ← 1
        i ← i + 1
    se i < n
        então A[i] ← 0
```

Considere que as operações de manipulação de um contador binário com K bits sejam como abaixo:

```
para i ← 1 até n faça
    Incrementa (A, K)
    Decrementa (A, K)
```

No pior caso, a chamada de Incrementa faz K alterações. Quando isso ocorre, a chamada de Decrementa reverte a operação, também produzindo K alterações.

Por exemplo:

- estado inicial do vetor em um determinado i

0	1	1	1

- após Incrementa

1	0	0	0

- após Decrementa (volta ao estado inicial)

0	1	1	1

É possível observar que se Incrementa produzir K alterações, Decrementa produzirá o mesmo e reverterá o estado do vetor para o estado anterior ao Incrementa, gerando um loop para as próximas operações. No pior caso, cada execução de Incrementa e Decrementa produz K alterações e, portanto, as n operações têm custo $\Theta(nK)$

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.

Seja a seguinte sequência de n operações

para $i \leftarrow 1$ até n faça
	1 operação (E)

Similar às tabelas dinâmicas temos

c_i = custo da i-ésima operação

$$c_i = \begin{cases} 1, & \text{se } i \text{ não é potência de } 2 \\ i, & \text{se } i \text{ é potência de } 2 \end{cases}$$

$$\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\lg n} 2^j = n + (2n - 1) < 3n$$

O tempo amortizado por operação é $3 \in O(1)$

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.

Da descrição, tem-se que as operações possíveis são empilha e desempilha, além do back-up. Para cada operação empilha e desempilha usamos 2 créditos, um para pagar pela própria operação, e o segundo fica armazenado em cada elemento para pagar por sua cópia ao realizar o backup.

Uma vez que cada elemento da pilha tem 1 crédito armazenado, e a pilha sempre tem um número de elementos não negativo, garantimos que o valor do crédito seja não negativo e suficiente para pagar por uma operação de back-up.

Portanto, o custo amortizado por operação é 2, ou seja, $O(1)$ (constante). E o custo de n operações sobre a pilha é $O(n)$.

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.)

Considere as seguintes implementações de incrementa1 e zera-contador

```
incrementa1 (A, K)
1   i ← 0
2   bit ← -1  (bit mais significativo)
3   enquanto i < K e A[i] = 1 faça
4       A[i] ← 0
5       se i == bit
6           então bit ← -1
7       i ← i + 1
8   se i < K
9       então A[i] ← 1
10      se i > bit
11          então bit ← i
```

```
zera_contador (A, bit)
1   se bit ≥ 0
2       então para i ← bit decrescendo até 0 faça
3           se A[i] == 1
4               então A[i] ← 0
```

Para calcular o custo das operações podemos usar a análise por créditos

- atribuímos 3 créditos por incrementa
  - o primeiro é usado para pagar a alteração para 1 na linha 9
  - o segundo é armazenado sobre o bit alterado na linha 9
  - o terceiro é armazenado quando o bit mais significativo muda para

maior na linha 11.

As alterações na linha 4 são pagas por créditos armazenados na linha 9.

As alterações na função zera-contador são pagas por créditos que foram armazenados na linha 11. Para resetar o contador, somente é necessário que o bit mais significativo e os anteriores sejam alterados. Assim, armazenar o número de vezes que o bit mais significativo aumentou de posição corresponde ao número de bits que precisam ser alterados em zera-contador.

Portanto, o custo amortizado por operação é $3 \in O(1)$ e o custo total da sequência de operações é $\leq 3n \in O(n)$.

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. Sugira um esquema para isso que resulte em um custo amortizado constante para operações de inserção e remoção. Faça a análise do esquema proposto justificando a sua resposta.

resolvida no resumo, mas refazer aqui

6. Exercício 1.3 de `http://cs.nyu.edu/~yap/classes/funAlgo/05f/lect/l6.pdf`.

Exercise 1.3: Let us generalize the example of incrementing binary counters. Suppose we have a collection of binary counters, all initialized to 0. We want to perform a sequence of operations, each of the type

$$\texttt{inc}(C), \quad \texttt{double}(C), \quad \texttt{add}(C, C')$$

where C, C' are names of counters. The operation $\texttt{inc}(C)$ increments the counter C by 1; $\texttt{double}(C)$ doubles the counter C; finally, $\texttt{add}(C, C')$ adds the contents of C' to C while simultaneously set the counter C' to zero. Show that this problem has amortized constant cost per operation.

We must define the cost model. The length of a counter is the number of bits used to store its value. The cost to double a counter C is just 1 (you only need to prepend a single bit to C). The cost of $\texttt{add}(C, C')$ is the number of bits that the standard algorithm needs to look at (and possibly modify) when adding C and C'. E.g., if $C = 11, 1001, 1101$ and $C' = 110$, then $C + C' = 11, 1010, 0011$ and the cost is 9. This is because the algorithm only has to look at 6 bits of C and 3 bits of C'. Note that the 4 high-order bits of C are not looked at (think of them as simply being "copied" to the output). After this operation, C has the value $11, 1010, 0011$ and C' has the value 0.
HINT: The potential of a counter C should take into account the number of 1's as well as the bit-length of the counter.

operações:

- inc (C)
- double (C)
- add (C, C")

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.

disjuntos: conjuntos com nenhum elemento em comum.

Seja L_1 e L_2 duas listas e L_1.head o representante da lista L_1.

A ideia é

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 $\mathrm{O}(m)$. O que acontece com o tempo consumido por uma sequência deste tipo se apenas compressão de caminhos estiver implementada?