MAC 6711 - Tópicos de Análise de Algoritmos

Departamento de Ciência da Computação
Primeiro semestre de 2025

Lista 6

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

2. **(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 $\mathrm{O}(n)$, atribuindo valores adequados de créditos a cada operação.

3. **(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.)

4. 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.

5. Considere a seguinte alternativa para representar um contador binário com k bits. O contador é representado por dois vetores com k posições, $P[0\,..\,k-1]$ e $N[0\,..\,k-1]$, onde, para cada i, no máximo um entre $P[i]$ e $N[i]$ vale 1. (O valor real do contador é a diferença $P - N$.)

   Abaixo estão implementações das operações de **Incrementa** e **Decrementa** para tal representação. Para simplificar, assumimos que o número k é grande o suficiente para que os algoritmos abaixo não acessem um índice inválido.

   Algoritmo Incrementa (P, N)

   1. $i \leftarrow 0$
   2. **enquanto** $P[i] = 1$ **faça**
   3. $\quad P[i] \leftarrow 0$
   4. $\quad i \leftarrow i + 1$
   5. **se** $N[i] = 1$
   6. $\quad$ **então** $N[i] \leftarrow 0$
   7. $\quad$ **senão** $P[i] \leftarrow 1$

   Algoritmo Decrementa (P, N)

   1. $i \leftarrow 0$
   2. **enquanto** $N[i] = 1$ **faça**
   3. $\quad N[i] \leftarrow 0$
   4. $\quad i \leftarrow i + 1$
   5. **se** $P[i] = 1$
   6. $\quad$ **então** $P[i] \leftarrow 0$
   7. $\quad$ **senão** $N[i] \leftarrow 1$

   Abaixo está um exemplo destes algoritmos em ação. (Note que qualquer número exceto o zero pode ser representado de diversas maneiras.)

   $$\begin{array}{ccccccccccccc} P = 10001 & & P = 10010 & & P = 10011 & & P = 10000 & & P = 10000 & & P = 10000 & & P = 10001 \\ N = 01100 & +1 & N = 01100 & +1 & N = 01100 & +1 & N = 01000 & -1 & N = 01001 & -1 & N = 01010 & +1 & N = 01010 \\ P - N = 5 & & P - N = 6 & & P - N = 7 & & P - N = 8 & & P - N = 7 & & P - N = 6 & & P - N = 7 \end{array}$$

   Suponha que o contador começa de $(0, 0)$ (ou seja, $P = 0^k$ e $N = 0^k$), e sejam aplicadas n operações, cada uma delas sendo um **Incrementa** ou um **Decrementa**. Suponha que $n < 2^k$, de modo que de fato os algoritmos não acessem um índice inválido dos vetores. Neste caso, o custo de pior caso do **Incrementa** e do **Decrementa** é $\Theta(\lg n)$.

   Prove que o custo amortizado do **Incrementa** e do **Decrementa** é $\mathrm{O}(1)$.

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

7. Dados $n+1$ pontos $p, p_1, \ldots, p_n$, dizemos que p é uma *combinação convexa* de $p_1, \ldots, p_n$ se existem números reais não-negativos $\lambda_1, \ldots, \lambda_n$ tais que (a) $\sum_{i=1}^{n} \lambda_i = 1$ e (b) $\sum_{i=1}^{n} \lambda_i p_i = p$. O *fecho convexo* de uma coleção finita de pontos é o conjunto de todas as combinações convexas de pontos da coleção. O *casco convexo* de uma coleção finita de pontos é o polígono que delimita o fecho convexo dessa coleção. Como o casco convexo é um polígono, ele pode ser dado por uma seqüência de pontos: os vértices do polígono. Note que os pontos dessa seqüência são sempre pontos da coleção original de pontos.

   O seguinte algoritmo, de Graham, determina o casco convexo da coleção $\{p_1, \ldots, p_n\}$. Vamos supor, para simplificar, que não há na coleção três pontos colineares. No pseudo-código abaixo, para três pontos distintos p, q e w, denotamos por $\theta(p, q, w)$ o ângulo entre a reta que passa por p e q e a reta que passa por q e w.

```
GRAHAM(p, n)
 1  PRELIMINARES (p, n)
 2  p[n + 1] ← p[1]
 3  c[1] ← p[1]
 4  t ← 1
 5  para k ← 2 até n faça
 6     t ← t + 1
 7     c[t] ← p[k]
 8     enquanto t > 2 e θ(c[t − 1], c[t], p[k + 1]) ≤ 180^o faça
 9        t ← t − 1
10  devolva c
```

```
PRELIMINARES(p, n)
1  min ← 1
2  para i ← 2 até n faça
3     se y(p[i]) < y(p[min])
4        então min ← i
5  p[1] ↔ p[min]
6  seja q um ponto tal que a reta que passa por q e p[1] é paralela ao eixo x
7  ordene p[2 . . n] de modo que θ(q, p[1], p[2]) < · · · < θ(q, p[1], p[n])
```

   Demonstre que as linhas 2 a 10 do algoritmo de Graham consomem tempo $O(n)$.

8. Como na aula para o *rr*-splay step, faça a análise do *lr*-splay step e do *r*-splay step.

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. Prove que o custo de pior caso de uma chamada do FINDSET(x) é $O(\lg n)$, onde n é o número de elementos na árvore onde está o x.

12. 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?