Lista 3 - MAC0122 Princípios de Desenvolvimento de Algoritmos - POLI

Prof. Ronaldo Fumio Hashimoto

1 Exercícios

Os exercícios de 24 a 33 correspondem, respectivamente, aos exercícios de 3.24 a 3.33 do livro de Robert Sedgewick [1].

24. Escreva uma função que retorna o número de nós de uma lista circular, dado um ponteiro para um dos nós desta lista.

25. Escreva um fragmento de código que determina o número de nós que estão entre dois nós referênciados por dois ponteiros de nós `x` e `t` em uma lista circular.

26. Escreva um fragmento de código que, dados dois ponteiros `x` e `t` para duas listas circulares disjuntas, insire a lista apontada por `t` na lista apontada por `x`, no ponto seguinte a `x`.

27. Dados dois ponteiros para nós `x` e `t` em uma lista circular, escreva um fragmento de código que move o nó seguinte a `t` para a posição do nó seguinte ao nó seguinte de `x`.

28. No programa abaixo (Programa 3.9 no livro), quando a lista está sendo construida, cada *link* é estabelecido duas vezes porque a lista circular é mantida depois que cada nó é inserido. Modifique o programa para construir a lista circular sem este trabalho extra.

Programa 3.9 (Problema de Josephus: é formada uma roda com N pessoas. No primeiro passo, a M-ésima pessoa é removida da roda. A cada passo subsequente, a M-ésima pessoa, contada a partir do ponto de parada do passo anterior, é removida da roda, até que sobre apenas uma pessoa).

```
#include <stdio.h>
typedef struct node* link;
struct node { int item; link next; };
main(int argc, char* argv[])
{ int i, N = atoi(argv[1]), M = atoi(argv[2]);
  link t = malloc(sizeof(*t)), x = t;
  t->item = 1; t->next = t;
  for (i = 2; i <= N; i++)
    {
      x = (x->next = malloc(sizeof(*x)));
      x->item = i; x->next = t;
    }
  while (x != x->next)
    {
      for (i = 1; i < M; i++) x = x->next;
      x->next = x->next->next; N--;
    }
  printf("%d\n", x->item);
}
```

29. Dê o tempo de execução do programa do exercicio 28 (Programa 3.9 do livro), com o fator constante, como uma função de M e N.

30. Use o programa do exercicio 28 (Programa 3.9 do livro) para determinar o valor da função de Josephus para $M = 2, 3, 5, 10$ e $N = 10^3$, 10^4, 10^5, e 10^6.

31. Use o programa do exercicio 28 (Programa 3.9 do livro) para plotar a função Josephus versus N para $M = 10$ e N de 10 a 1000.

32. Refaça a tabela da Fig 3.6, começando com o item i inicialmente na posição $N - i$ no vetor (*array*).

33. Desenvolva uma versão do programa do exercicio 28 (Programa 3.9) que usa um vetor (*array*) de índices para implementar a lista ligada (veja Figura abaixo, Figura 3.6 do livro).

	0	1	2	3	4	5	6	7	8
item	1	2	3	4	5	6	7	8	9
next	1	2	3	4	5	6	7	8	0
5	1	2	3	4	5	6	7	8	9
	1	2	3	5	5	6	7	8	0
1	1	2	3	4	5	6	7	8	9
	1	2	3	5	5	6	7	8	1
7	1	2	3	4	5	6	7	8	9
	1	2	3	5	5	7	7	8	1
4	1	2	3	4	5	6	7	8	9
	1	2	5	5	5	7	7	8	1
3	1	2	3	4	5	6	7	8	9
	1	5	5	5	5	7	7	8	1
6	1	2	3	4	5	6	7	8	9
	1	7	5	5	5	7	7	8	1
9	1	2	3	4	5	6	7	8	9
	1	7	5	5	5	7	7	1	1
2	1	2	3	4	5	6	7	8	9
	1	7	5	5	5	7	7	7	1

Figura 1: Figura adaptada de [2].
Representação de lista ligada como vetor.
Essa sequência exibe a lista ligada para o problema de Josephus, implementada como índices de vetores em vez de ponteiros. O índice do item seguinte ao item com índice 0 na lista é `next[0]`, e assim por diante. Inicialmente (as três linhas do topo), o item da pessoa i tem índice $i-1$, e nós formamos uma lista circular estabelecendo `next[i]` para $i+1$ para i de 0 a 8 e `next[8]` para 0. Para simular o processo de remoção, nós trocamos os *links* (entrada de vetor `next`) mas não movemos os itens. Cada par de linhas exibe o resultado de andar pela lista quatro vezes através de `x = next[x]`. Então, apaga-se o quinto item (exibida ao lado esquerdo) ao modificarmos `next[x]` para `next[next[x]]`.

Referências

[1] R. Sedgewick, "Algorithms in c—third edition," p. 82, 1998.

[2] R. Sedgewick, "Algorithms in c—third edition," 1998.