Lista 5 - MAC0122 Princípios de Desenvolvimento de Algoritmos - POLI

Prof. Ronaldo Fumio Hashimoto

1 Exercícios

Os exercícios de 47 a 55 correspondem, respectivamente, aos exercícios de 3.47 a 3.55 do livro de Robert Sedgewick [1].

47. Escreva um programa que libera (chama a função `free` para um ponteiro) todos os nós de uma dada lista.

48. Escreva um programa que libere os nós que estão posicionados em posições divisíveis por 5 em um lista ligada (o quinto, décimo, décimo quinto, e assim por diante).

49. Escreva um programa que libere os nós localizados em posições pares de uma lista ligada (o segundo, quarto, sexto, e assim por diante).

50. Implemente a interface abaixo (Programa 3.12 do livro) usando `malloc` e `free` diretamente em `newNode` e `freeNode` respectivamente.

    Programa 3.12

```
typedef struct node* link;
typedef int itemType;
struct node { itemType item; link next; };
typedef link Node;
void initNodes(int);
link newNode(int);
void freeNode(link);
void insertNext(link, link);
link deleteNext(link);
link Next(link);
itemType Item(link);
```

51. Execute estudos empíricos comparando o tempo de execução das funções de alocação de memória do Programa 3.14 com o programa desenvolvido no exercício 50 (que usa `malloc` e `free`). Use o Programa 3.13 com $M = 2$ e $N = 10^3$, 10^4, 10^5, e 10^6 para rodar seus estudos.

Programa 3.13

```
#include "list.h"
main(int argc, char *argv[])
  { int i, N = atoi(argv[1]), M = atoi(argv[2]);
    Node t, x;
    initNodes(N);
    for (i = 2, x = newNodes(1); i <= N; i++)
      { t = newNode(i); insertNext(x, t); x = t; }
    while (x != Next(x))
       {
          for (i = 1; i < M; i++) x = Next(x);
          freeNode(deleteNext(x));
       }
    printf("%d\n", Item(x));
}
```

Programa 3.14

```
#include <stdlib.h>
#include "list.h"
link freelist;
void initNodes(int N)
  { int i;
    freelist = malloc((N+1)*(sizeof *freelist));
    for (i = 0; i < N+1; i++)
      freelist[i].next = &freelist[i+1];
    freelist[N].next = NULL;
   }
link newNode(int i)
   { link x = deleteNext(freelist);
     x->item = i; x->next = x;
     return x;
   }
void freeNode(link x)
   { insertNext(freelist, x); }
void insertNext(link x, link t)
   {t->next = x->next; x->next = t;}
link deleteNext(link x)
   { link t = x->next; x->next = t->next; return t; }
link Next(link x)
   { return x->next; }
itemType Item(link x)
   { return x->item; }
```

52. Implemente a interface do Programa do exercicio 50 (Programa 3.12 do livro) utilizando índices de vetor (*array*) em vez de ponteiros (não utilize nó cabeça (*head*)), de maneira que a figura abaixo (Fig. 3.11 do livro) represente as operações do seu programa.

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

Figura 1: Figura adaptada de [2].

Representação de lista ligada como vetor, com *free list*.

Essa versão da Figura 3.6 do livro exibe o resultado de manter uma *free list* com nós excluídos da lista circular, com o índice do primeiro nó da *free list* dado à esquerda. No final do processo, a *free list* é uma lista ligada contendo todos os nós que foram excluídos. Seguindo os *links* de baixo para cima, começando do 1, vemos que os itens estão na ordem 2 9 6 3 4 7 1 5, que é a ordem inversa de suas remoções.

53. Suponha que você tenha um conjunto de nós sem ponteiros nulos (cada nó aponta para si mesmo ou algum outro nó do conjunto). Prove que você obtém um ciclo se você começar a seguir os links a partir de um dado nó.

54. Sob as condições do exercício 53, escreva um fragmento de código que, dado um ponteiro para nó, encontra o número de diferentes nós que são alcançados a partir do nó dado apenas seguindo os *links* dos nós. Não modifique os nós. Não use mais do que uma quantidade constante de memória extra.

55. Sob as condições do exercício 54, escreva uma função que determina se dados dois *links*, se seguidos, terminam no mesmo ciclo ou não.

Referências

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

[2] R. Sedgewick, "Algorithms in c—third edition," p. 106, 1998.