AULA 21

LISTAS LINEARES CIRCULARES

Algumas operações básicas em listas lineares, em especial aquelas implementadas em alocação encadeada, não são muito eficientes. Um exemplo emblemático desse tipo de operação é a busca, mais eficiente e mais econômica em listas lineares em alocação seqüencial como pudemos perceber em aulas anteriores. As listas lineares circulares realizam algumas operações ainda mais eficientemente, já que possuem uma informação a mais que as listas lineares ordinárias.

Uma lista linear circular é, em geral, implementada em alocação encadeada e difere de uma lista linear por não conter um ponteiro para nulo em qualquer de suas células, o que não permite que se identifique um fim, ou mesmo um começo, da lista. Caso exista uma célula cabeça da lista, então é possível identificar facilmente um início e um fim da lista. Nesta aula, baseada nas referências [7, 13], veremos a implementação das operações básicas de busca, inserção e remoção em listas lineares circulares com cabeça em alocação encadeada.

21.1 Operações básicas em alocação encadeada

A adição de uma informação na estrutura de uma lista linear em alocação encadeada permite que operações sejam realizadas de forma mais eficiente. Quando a última célula da lista linear encadeada aponta para a primeira célula, temos uma **lista linear circular**. Dessa forma, não podemos identificar um fim da lista linear. No entanto, o que aparenta uma dificuldade é, na verdade, uma virtude desse tipo de estrutura, como veremos nas funções que implementam suas operações básicas de busca, inserção e remoção. Veja a figura 21.1 para um exemplo de uma lista linear circular com cabeça em alocação encadeada.

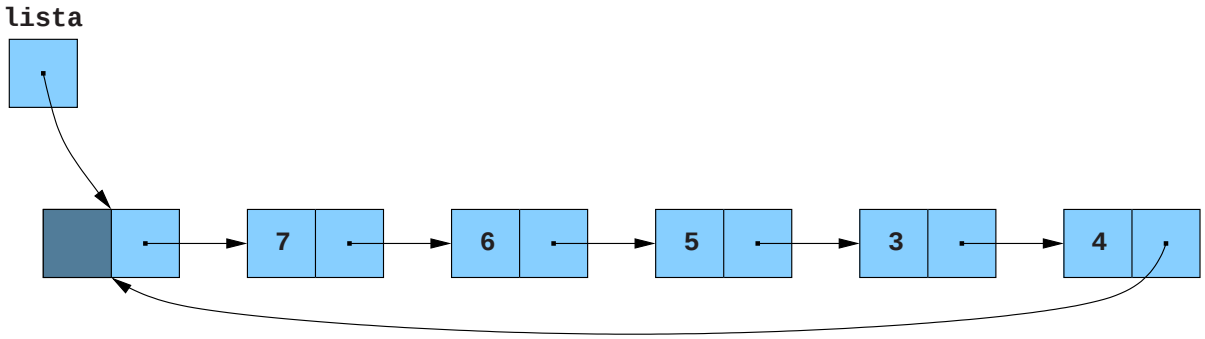


Figura 21.1: Representação de uma lista linear circular sem cabeça em alocação encadeada.

A definição de um tipo célula da lista linear circular permanece a mesma:

```
typedef struct cel {
   int chave;
   struct cel *prox;
} celula;
```

A declaração e inicialização de uma lista linear circular com cabeça em alocação encadeada é realizada da seguinte forma:

```
   celula *lista;
   lista = (celula *) malloc(sizeof (celula));
   lista->prox = lista;
```

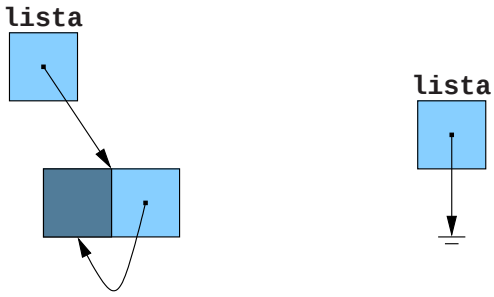
ou ainda:

```
   celula c, *lista;
   c.prox = &c;
   lista = &c;
```

e sem cabeça:

```
   celula *lista;
   lista = NULL;
```

A figura 21.2 mostra uma lista linear circular vazia (a) com cabeça e (b) sem cabeça.



(a) (b)

Figura 21.2: Representação de uma lista linear circular vazia (a) com cabeça e (b) sem cabeça.

A seguir vamos implementar as operações básicas de busca, inserção e remoção sobre listas lineares circulares com cabeça em alocação encadeada. A implementação das operações básicas em uma lista linear circular sem cabeça em alocação encadeada é deixada como exercício.

21.1.1 Busca

Imagine que queremos verificar se uma chave está presente em uma lista linear circular em alocação emcadeada. A função `busca_circular` recebe um ponteiro `c` para a lista linear circular e um valor `x` e devolve uma célula contendo a chave procurada `x`, caso `x` ocorra na lista. Caso contrário, a função devolve `NULL`.

```
/* Recebe uma chave x e uma lista linear circular c e verifica se x consta em
   c, devolvendo a célula em que x se encontra ou NULL em caso contrário */
celula *busca_circular(int x, celula *c)
{
   celula *p, *q;

   p = c;
   p->chave = x;
   q = c->prox;
   while (q->chave != x)
      q = q->prox;
   if (q != c)
      return q;
   else
      return NULL;
}
```

Observe que a busca em uma lista linear circular em alocação encadeada é muito semelhante à busca em uma lista linear ordinária da aula 18. Em particular, seu tempo de execução de pior caso é proporcional ao número de células da lista, assim como é o tempo de execução de pior caso da busca em uma lista linear ordinária. Observe, no entanto, que as constantes são diferentes: enquanto na busca em uma lista linear ordinária a estrutura de repetição sempre realiza duas comparações, a estrutura de repetição da busca em uma lista circular realiza apenas uma comparação.

21.1.2 Inserção

A inserção em uma lista linear circular com cabeça em alocação encadeada é simples e semelhante à inserção em uma lista linear ordinária. Temos apenas de prestar atenção para que a inserção sempre se mantenha circular, o que se dá de modo natural se inserimos a célula que contém a nova chave sempre como a célula seguinte à cabeça.

```
/* Recebe uma chave y e uma lista circular c e insere y no início da lista c */
void insere_circular(int y, celula *c)
{
   celula *nova;

   nova = (celula *) malloc(sizeof (celula));
   nova->chave = y;
   nova->prox = c->prox;
   c->prox = nova;
}
```

A figura 21.3 mostra a execução da função `insere_circular` para uma chave 2 e uma lista circular `c`.

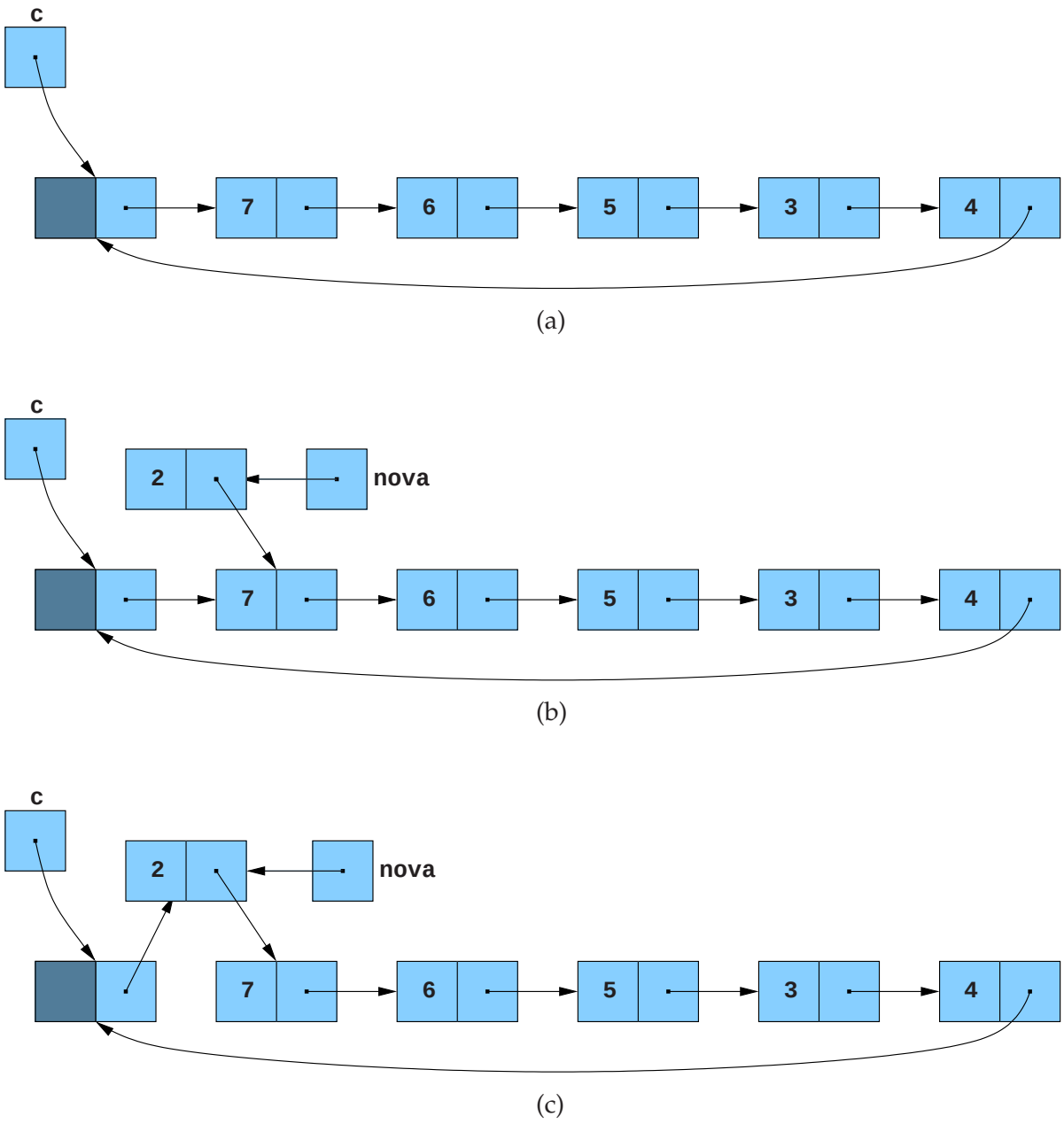


Figura 21.3: Inserção da chave 2 em uma lista circular `c`.

21.1.3 Remoção

A função de remoção de um elemento em uma lista linear circular em alocação encadeada é também muito semelhante à remoção de um elemento realizada em uma lista linear ordinária. Buscamos a célula a ser removida como sendo aquela que possui a chave `x`, fornecida como parâmetro para a função.

```
/* Recebe uma chave x e uma lista linear circular c e
   remove a célula com chave x da lista c, caso exista */
void remove_circular(int x, celula *c)
{
   celula *p, *q;

   p = c;
   p->chave = x;
   q = p->prox;
   while (q->chave != x) {
      p = q;
      q = q->prox;
   }
   if (q != c) {
      p->prox = q->prox;
      free(q);
   }
}
```

A figura 21.4 mostra a execução de `remove_circular` para a chave 4 e a lista circular `c`.

Exercícios

21.1 Escreva funções que implementem as operações básicas de busca, inserção e remoção sobre uma lista linear circular sem cabeça em alocação encadeada.

21.2 Escreva uma função para liberar todas as células de uma lista linear circular em alocação encadeada para a lista de espaços disponíveis da memória. Escreva duas versões dessa função, uma para uma lista linear circular com cabeça e outra para uma lista linear circular sem cabeça.

21.3 Suponha que queremos implementar uma lista linear circular em alocação encadeada de tal forma que as chaves de suas células sempre permaneçam em ordem crescente. Escreva as funções para as operações básicas de busca, inserção e remoção para listas lineares circulares em alocação encadeada com as chaves em ordem crescente. Escreva dois conjuntos de funções, considerando listas lineares circulares com cabeça e listas lineares circulares sem cabeça.

21.4 O **problema de Josephus** foi assim descrito através do relato de Flavius Josephus, um historiador judeu que viveu no primeiro século, sobre o cerco de Yodfat. Ele e mais 40 soldados aliados estavam encurralados em uma caverna rodeada por soldados romanos e, não havendo saída, optaram então por suas próprias mortes antes da captura. Decidiram que formariam um círculo e, a cada contagem de 3, um soldado seria morto pelo grupo. Josephus foi o soldado restante e decidiu então não se suicidar, deixando essa estória como legado. Podemos descrever um problema mais geral como segue. Imagine n pessoas dispostas em círculo. Suponha que as pessoas são numeradas de 1 a n no sentido horário e que um número inteiro m, com $1 \leqslant m < n$, seja fornecido. Começando com a pessoa de número 1, percorra o círculo no sentido horário e elimine cada m-ésima pessoa enquanto o círculo tiver duas ou mais pessoas. Escreva e teste uma função que resolva o problema, imprimindo na saída o número do sobrevivente.

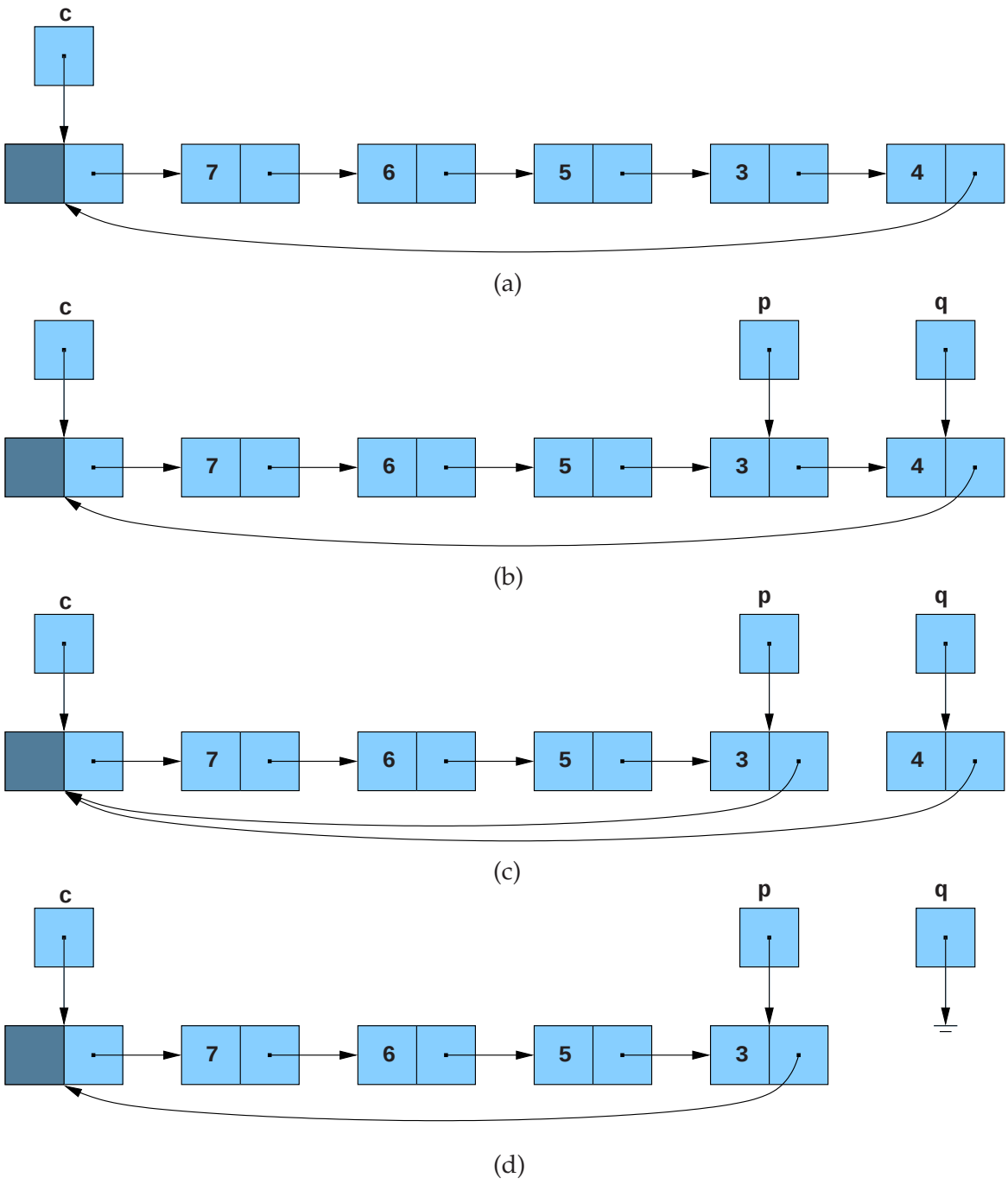


Figura 21.4: Remoção da chave 4 de uma lista circular `c`.

21.5 Um computador permite representar números inteiros até um valor máximo, que depende da sua organização e arquitetura interna. Suponha que seja necessário, em uma dada aplicação, o uso de números inteiros com precisão muito grande, digamos ilimitada. Uma forma de representar números inteiros com precisão ilimitada é mostrada na figura 21.5.

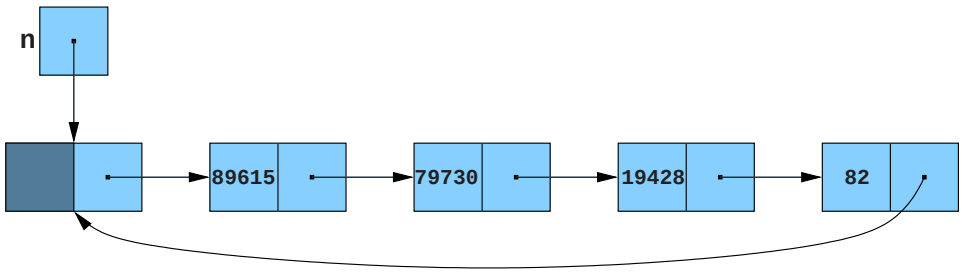


Figura 21.5: Uma lista linear circular com cabeça em alocação encadeada que armazena o número inteiro 82194287973089615.

O tipo célula, que define as células da lista linear circular, não sofre alterações na sua definição, sendo o mesmo o tipo `celula` previamente definido. Observe apenas que uma chave contém um número inteiro de até 5 algarismos.

Escreva uma função que receba duas listas lineares circulares contendo dois números inteiros de precisão ilimitada e devolva uma lista linear circular contendo o resultado da soma desses dois números.