Filas

Aula 20

Fábio Henrique Viduani Martinez Diego Padilha Rubert

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul

Algoritmos e Programação II, Análise de Sistemas, 2010

Conteúdo da aula

Introdução

- lista linear especial
- política de inserções e remoções bem definida
- inserção e remoção são as únicas operações
- inserção é realizada em um dos extremos da fila e remoção no outro extremo

Introdução

- lista linear especial
- política de inserções e remoções bem definida
- inserção e remoção são as únicas operações
- inserção é realizada em um dos extremos da fila e remoção no outro extremo

Introdução

- lista linear especial
- política de inserções e remoções bem definida
- inserção e remoção são as únicas operações
- inserção é realizada em um dos extremos da fila e remoção no outro extremo

Introdução

- lista linear especial
- política de inserções e remoções bem definida
- inserção e remoção são as únicas operações
- inserção é realizada em um dos extremos da fila e remoção no outro extremo

Definição

- **fila** é uma lista linear com dois extremos destacados e tal que as operações de inserção são realizadas em um dos extremos da lista e a remoção é realizada no outro extremo
- funcionamento dessa estrutura pode ser comparado a qualquer fila que usamos com freqüência como, por exemplo, uma fila de um banco

Definição

- **fila** é uma lista linear com dois extremos destacados e tal que as operações de inserção são realizadas em um dos extremos da lista e a remoção é realizada no outro extremo
- funcionamento dessa estrutura pode ser comparado a qualquer fila que usamos com freqüência como, por exemplo, uma fila de um banco

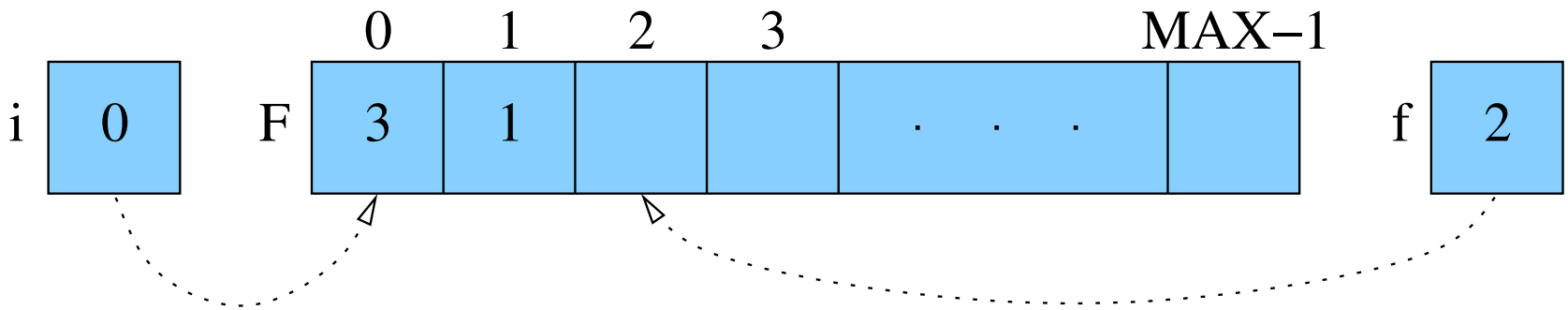
Operações básicas em alocação seqüencial

- fila armazenada em um segmento `F[i..f-1]` de um vetor `F[0..MAX-1]`, com $0 \leqslant$ `i` $\leqslant$ `f` $\leqslant$ `MAX`
- primeiro elemento da fila está na posição `i` e o último na posição `f-1`

Operações básicas em alocação seqüencial

- fila armazenada em um segmento `F[i..f-1]` de um vetor `F[0..MAX-1]`, com $0 \leqslant$ `i` $\leqslant$ `f` $\leqslant$ `MAX`
- primeiro elemento da fila está na posição `i` e o último na posição `f-1`

Definição



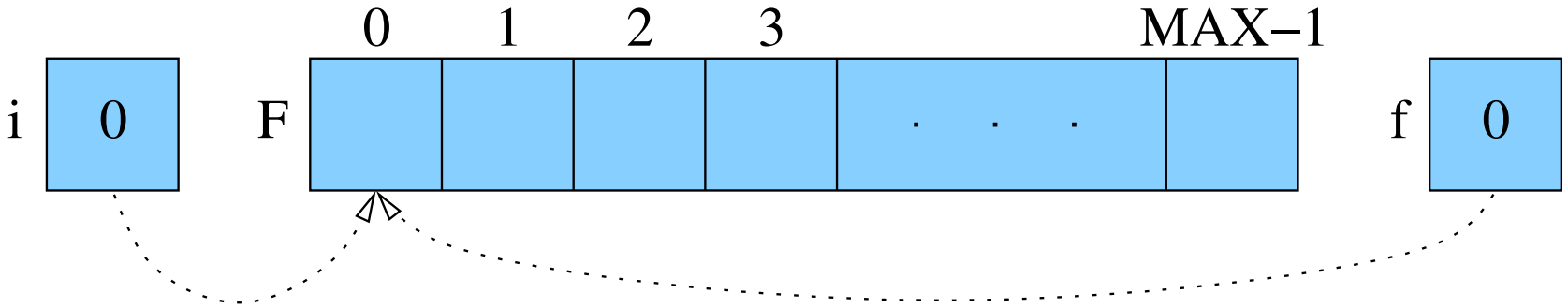
Operações básicas em alocação seqüencial

- fila está **vazia** se `i` = `f`
- fila está **cheia** se `f` = `MAX`

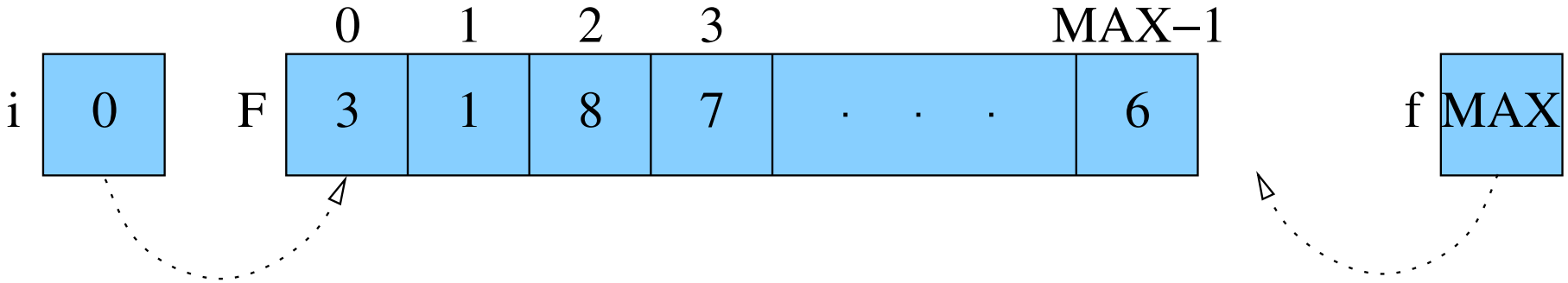
Operações básicas em alocação seqüencial

- fila está **vazia** se `i` = `f`
- fila está **cheia** se `f` = `MAX`

Operações básicas em alocação seqüencial



fila vazia



fila cheia

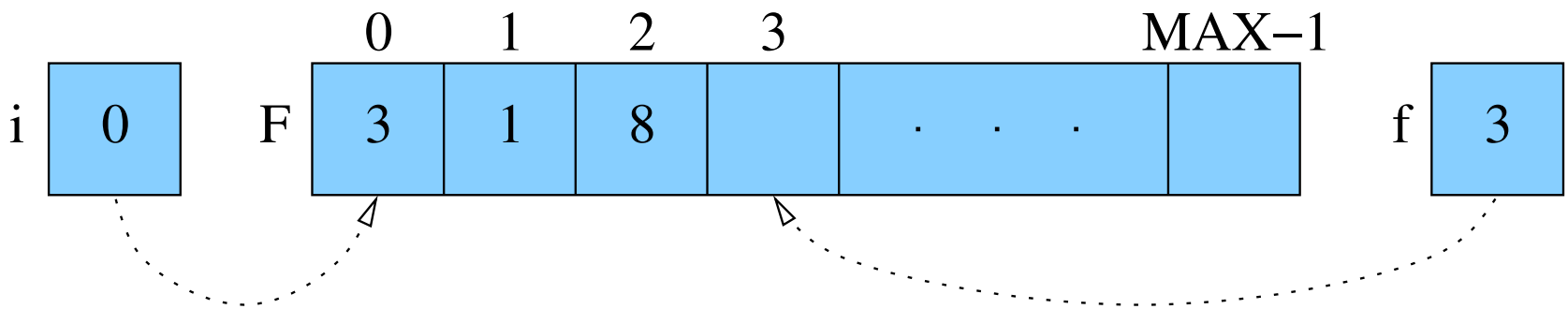
Operações básicas em alocação seqüencial

- declaração e inicialização de uma fila em alocação seqüencial

```
int i, f, F[MAX];
i = 0;
f = 0;
```

Operações básicas em alocação seqüencial

- inserção de uma chave na fila

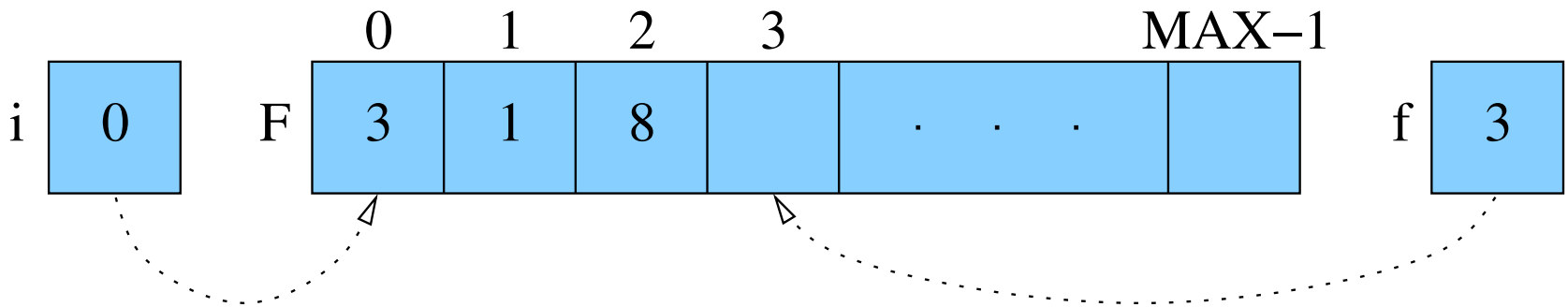


Operações básicas em alocação seqüencial

```
void enfileira_seq(int *f, int F[MAX], int y)
{
   if (*f != MAX) {
      F[f] = y;
      (*f)++;
   }
   else
      printf("Fila cheia!\n");
}
```

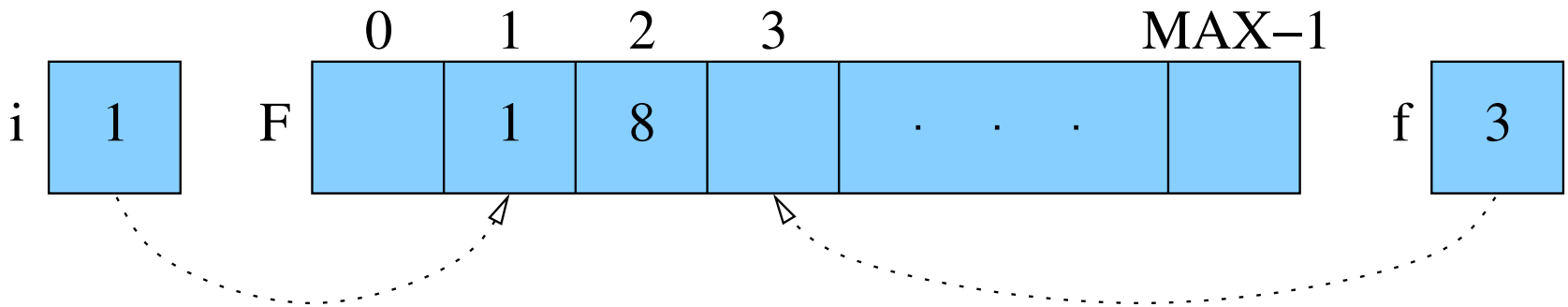
Operações básicas em alocação seqüencial

- remoção de uma chave na fila



Operações básicas em alocação seqüencial

- remoção de uma chave na fila



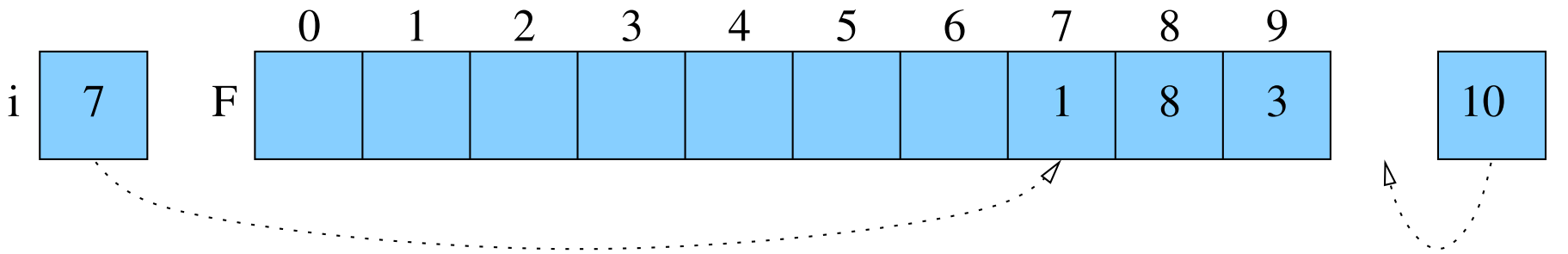
Operações básicas em alocação seqüencial

```
int desenfileira_seq(int *i, int f, int F[MAX])
{
   int x;

   if (*i != f) {
      x = F[*i];
      (*i)++;
   }
   else {
      x = INT_MAX;
      printf("Fila vazia!\n");
   }
   return x;
}
```

Operações básicas em alocação seqüencial

- Problema!



Operações básicas em alocação seqüencial

- Considere que as células são alocadas seqüencialmente como se estivessem em um círculo
- o compartimento `F[MAX-1]` é seguido pelo compartimento `F[0]`
- os elementos da fila estão dispostos no vetor `F[0..MAX-1]` em `F[i..f-1]` ou em `F[i..MAX-1]F[0..f-1]`
- fila está **vazia** se `i` $=$ `f` $= -1$
- fila está **cheia** se `f` $=$ `i`

Operações básicas em alocação seqüencial

- Considere que as células são alocadas seqüencialmente como se estivessem em um círculo
- o compartimento `F[MAX-1]` é seguido pelo compartimento `F[0]`
- os elementos da fila estão dispostos no vetor `F[0..MAX-1]` em `F[i..f-1]` ou em `F[i..MAX-1]F[0..f-1]`
- fila está **vazia** se `i` $=$ `f` $= -1$
- fila está **cheia** se `f` $=$ `i`

Operações básicas em alocação seqüencial

- Considere que as células são alocadas seqüencialmente como se estivessem em um círculo
- o compartimento `F[MAX-1]` é seguido pelo compartimento `F[0]`
- os elementos da fila estão dispostos no vetor `F[0..MAX-1]` em `F[i..f-1]` ou em `F[i..MAX-1]F[0..f-1]`
- fila está **vazia** se `i` $=$ `f` $= -1$
- fila está **cheia** se `f` $=$ `i`

Operações básicas em alocação seqüencial

- Considere que as células são alocadas seqüencialmente como se estivessem em um círculo
- o compartimento `F[MAX-1]` é seguido pelo compartimento `F[0]`
- os elementos da fila estão dispostos no vetor `F[0..MAX-1]` em `F[i..f-1]` ou em `F[i..MAX-1]F[0..f-1]`
- fila está **vazia** se `i` $=$ `f` $= -1$
- fila está **cheia** se `f` $=$ `i`

Operações básicas em alocação seqüencial

- Considere que as células são alocadas seqüencialmente como se estivessem em um círculo
- o compartimento `F[MAX-1]` é seguido pelo compartimento `F[0]`
- os elementos da fila estão dispostos no vetor `F[0..MAX-1]` em `F[i..f-1]` ou em `F[i..MAX-1]F[0..f-1]`
- fila está **vazia** se `i` $=$ `f` $= -1$
- fila está **cheia** se `f` $=$ `i`

Operações básicas em alocação seqüencial

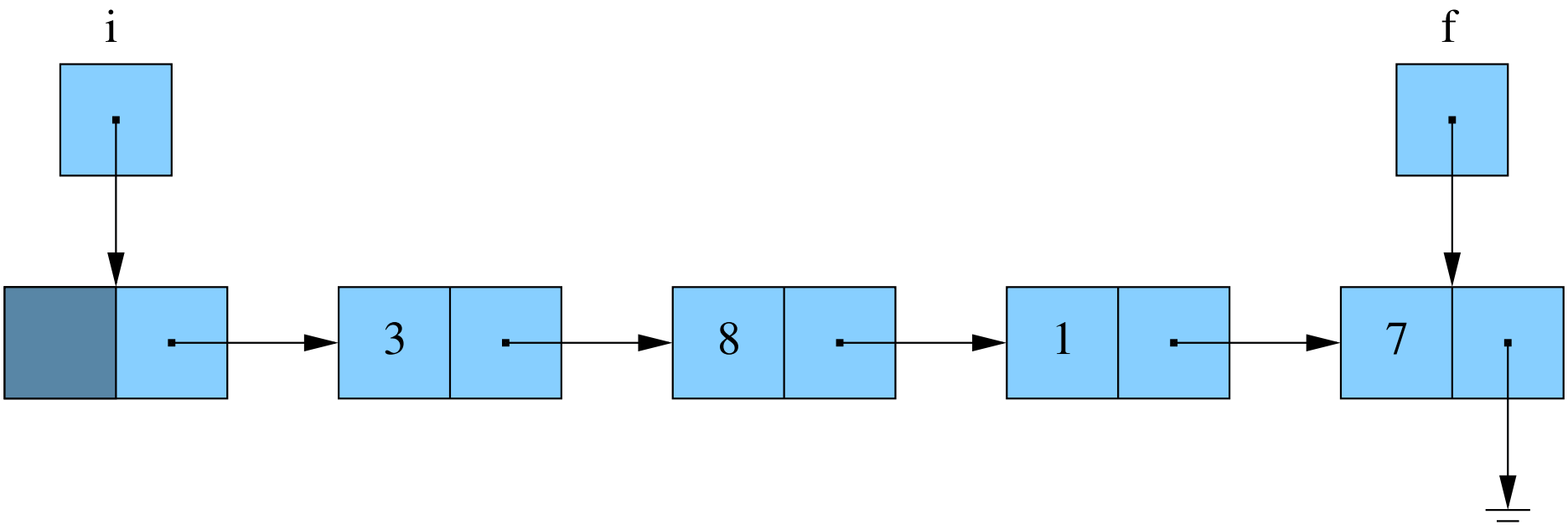
```
void enfileira_seq_2(int *i, int *f, int F[MAX], int y)
{
   if (*f != *i || *f == -1) {
      if (*f == -1) {
         *i = 0;
         *f = 0;
      }
      F[*f] = x;
      (*f)++;
      if (*f == MAX)
         *f = 0;
   }
   else
      printf("Fila cheia!\n");
}
```

Operações básicas em alocação seqüencial

```
int desenfileira_seq_2(int *i, int *f, int F[MAX])
{
   int r;

   r = INT_MIN;
   if (*i != -1) {
      r = F[*i];
      if (*i != *f)
         *i = (*i + 1) % MAX;
      else {
         *i = -1;
         *f = -1;
      }
   }
   else
      printf("Fila vazia!\n");
   return r;
}
```

Operações básicas em alocação encadeada



Operações básicas em alocação encadeada

- tipo célula:

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

- declaração e inicialização de uma fila vazia em alocação encadeada com cabeça:

```
celula *i, *f;
i = (celula *) malloc(sizeof (celula));
f = i;
```

- declaração e inicialização de uma fila sem cabeça:

```
celula *i, *f;
i = NULL;
f = NULL;
```

Operações básicas em alocação encadeada

- tipo célula:

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

- declaração e inicialização de uma fila vazia em alocação encadeada com cabeça:

```
celula *i, *f;
i = (celula *) malloc(sizeof (celula));
f = i;
```

- declaração e inicialização de uma fila sem cabeça:

```
celula *i, *f;
i = NULL;
f = NULL;
```

Operações básicas em alocação encadeada

- tipo célula:

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

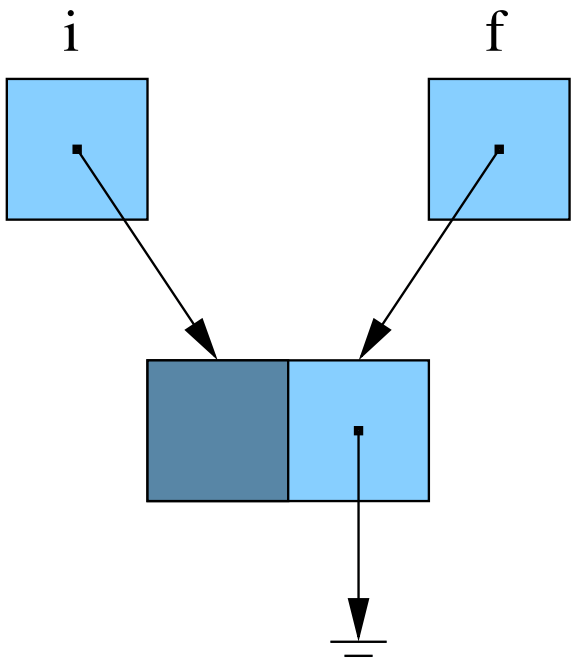
- declaração e inicialização de uma fila vazia em alocação encadeada com cabeça:

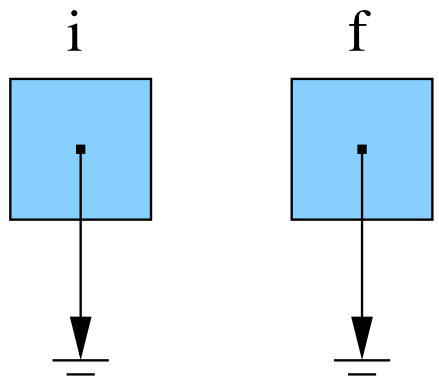
```
celula *i, *f;
i = (celula *) malloc(sizeof (celula));
f = i;
```

- declaração e inicialização de uma fila sem cabeça:

```
celula *i, *f;
i = NULL;
f = NULL;
```

Operações básicas em alocação encadeada





Operações básicas em alocação encadeada

```
void enfileira_enc_C(celula *i, celula *f, int y)
{
   celula *nova;

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

Operações básicas em alocação encadeada

```
int desenfileira_enc_C(celula *i, celula *f)
{
   int x;
   celula *p;

   p = i->prox;
   if (p != NULL) {
      x = p->chave;
      i->prox = p->prox;
      free(p);
      if (i->prox == NULL)
         f = NULL;
      return x;
   }
   else {
      printf("Fila vazia!\n");
      return INT_MIN;
   }
}
```

Exercícios

20.1 Implemente as funções de enfileiramento e desenfileiramento em uma fila em alocação encadaeda sem cabeça.

20.2 Um estacionamento possui um único corredor que permite dispor 10 carros. Os carros chegam pelo sul do estacionamento e saem pelo norte. Se um cliente quer retirar um carro que não está próximo do extremo norte, todos os carros impedindo sua passagem são retirados, o cliente retira seu carro e os outros carros são recolocados na mesma ordem que estavam originalmente. Sempre que um carro sai, todos os carros do sul são movidos para frente, de modo que as vagas fiquem disponíveis sempre no extremo sul do estacionamento. Escreva um algoritmo que processa o fluxo de chegada/saída deste estacionamento. Cada entrada para o algoritmo contém uma letra 'E' para entrada ou 'S' para saída, e o número da placa do carro. (*continua*)

Exercícios

20.1 Implemente as funções de enfileiramento e desenfileiramento em uma fila em alocação encadaeda sem cabeça.

20.2 Um estacionamento possui um único corredor que permite dispor 10 carros. Os carros chegam pelo sul do estacionamento e saem pelo norte. Se um cliente quer retirar um carro que não está próximo do extremo norte, todos os carros impedindo sua passagem são retirados, o cliente retira seu carro e os outros carros são recolocados na mesma ordem que estavam originalmente. Sempre que um carro sai, todos os carros do sul são movidos para frente, de modo que as vagas fiquem disponíveis sempre no extremo sul do estacionamento. Escreva um algoritmo que processa o fluxo de chegada/saída deste estacionamento. Cada entrada para o algoritmo contém uma letra 'E' para entrada ou 'S' para saída, e o número da placa do carro. (*continua*)

Exercícios

20.2 (*continuação*) Considere que os carros chegam e saem pela ordem especificada na entrada. O algoritmo deve imprimir uma mensagem sempre que um carro chega ou sai. Quando um carro chega, a mensagem deve especificar se existe ou não vaga para o carro no estacionamento. Se não existe vaga, o carro deve esperar até que exista uma vaga, ou até que uma instrução fornecida pelo usuário indique que o carro deve partir sem que entre no estacionamento. Quando uma vaga torna-se disponível, outra mensagem deve ser impressa. Quando um carro sai do estacionamento, a mensagem deve incluir o número de vezes que o carro foi movimentado dentro da garagem, incluindo a saída mas não a chegada. Este número é 0 se o carro partiu da linha de espera, isto é, se o carro esperava uma vaga, mas partiu sem entrar no estacionamento.

Exercícios

20.3 Implemente uma fila usando duas pilhas.

20.4 Implemente uma pilha usando duas filas.

20.5 Suponha que temos n cidades numeradas de 0 a $n-1$ e interligadas por estradas de mão única. As ligações entre as cidades são representadas por uma matriz A definida da seguinte forma: $A[x][y]$ vale 1 se existe estrada da cidade x para a cidade y e vale 0 em caso contrário. A figura abaixo ilustra um exemplo. A **distância** de uma cidade o a uma cidade x é o menor número de estradas que é preciso percorrer para ir de o a x. O problema que queremos resolver é o seguinte: determinar a distância de uma dada cidade o a cada uma das outras cidades da rede. As distâncias são armazenadas em um vetor d de tal modo que $d[x]$ seja a distância de o a x. Se for impossível chegar de o a x, podemos dizer que $d[x]$ vale ∞. Usamos ainda -1 para representar ∞ uma vez que nenhuma distância "real" pode ter valor -1. (*continua*)

Exercícios

20.3 Implemente uma fila usando duas pilhas.

20.4 Implemente uma pilha usando duas filas.

20.5 Suponha que temos n cidades numeradas de 0 a $n-1$ e interligadas por estradas de mão única. As ligações entre as cidades são representadas por uma matriz A definida da seguinte forma: $A[x][y]$ vale 1 se existe estrada da cidade x para a cidade y e vale 0 em caso contrário. A figura abaixo ilustra um exemplo. A **distância** de uma cidade o a uma cidade x é o menor número de estradas que é preciso percorrer para ir de o a x. O problema que queremos resolver é o seguinte: determinar a distância de uma dada cidade o a cada uma das outras cidades da rede. As distâncias são armazenadas em um vetor d de tal modo que $d[x]$ seja a distância de o a x. Se for impossível chegar de o a x, podemos dizer que $d[x]$ vale ∞. Usamos ainda -1 para representar ∞ uma vez que nenhuma distância "real" pode ter valor -1. (*continua*)

Exercícios

20.3 Implemente uma fila usando duas pilhas.

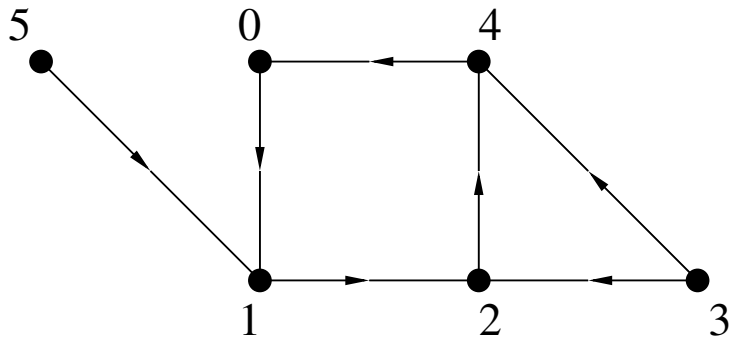
20.4 Implemente uma pilha usando duas filas.

20.5 Suponha que temos n cidades numeradas de 0 a $n-1$ e interligadas por estradas de mão única. As ligações entre as cidades são representadas por uma matriz A definida da seguinte forma: $A[x][y]$ vale 1 se existe estrada da cidade x para a cidade y e vale 0 em caso contrário. A figura abaixo ilustra um exemplo. A **distância** de uma cidade o a uma cidade x é o menor número de estradas que é preciso percorrer para ir de o a x. O problema que queremos resolver é o seguinte: determinar a distância de uma dada cidade o a cada uma das outras cidades da rede. As distâncias são armazenadas em um vetor d de tal modo que $d[x]$ seja a distância de o a x. Se for impossível chegar de o a x, podemos dizer que $d[x]$ vale ∞. Usamos ainda -1 para representar ∞ uma vez que nenhuma distância "real" pode ter valor -1. (*continua*)

Exercícios

20.5 (*continuação*)

	0	1	2	3	4	5
0	0	1	0	0	0	0
1	0	0	1	0	0	0
2	0	0	0	0	1	0
3	0	0	1	0	1	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0



	0	1	2	3	4	5
d	2	3	1	0	1	6

Solucione o problema das distâncias em uma rede usando uma fila em alocação seqüencial. Solucione o mesmo problema usando uma fila em alocação encadeada.

Exercícios

20.6 Imagine um tabuleiro quadrado 10–por–10. As casas “livres” são marcadas com 0 e as casas “bloqueadas” são marcadas com -1. As casas $(1, 1)$ e $(10, 10)$ estão livres. Ajude uma formiga que está na casa $(1, 1)$ a chegar à casa $(10, 10)$. Em cada passo, a formiga só pode se deslocar para uma casa livre que esteja à direita, à esquerda, acima ou abaixo da casa em que está.

20.7 Um **deque** é uma lista linear que permite a inserção e a remoção de elementos em ambos os seus extremos. Escreva quatro funções para manipular um deque: uma que realiza a inserção de um novo elemento no início do deque, uma que realiza a inserção de um novo elemento no fim do deque, uma que realiza a remoção de um elemento no início do deque e uma que realiza a remoção de um elemento no fim do deque.

Exercícios

20.6 Imagine um tabuleiro quadrado 10–por–10. As casas “livres” são marcadas com 0 e as casas “bloqueadas” são marcadas com -1. As casas $(1, 1)$ e $(10, 10)$ estão livres. Ajude uma formiga que está na casa $(1, 1)$ a chegar à casa $(10, 10)$. Em cada passo, a formiga só pode se deslocar para uma casa livre que esteja à direita, à esquerda, acima ou abaixo da casa em que está.

20.7 Um **deque** é uma lista linear que permite a inserção e a remoção de elementos em ambos os seus extremos. Escreva quatro funções para manipular um deque: uma que realiza a inserção de um novo elemento no início do deque, uma que realiza a inserção de um novo elemento no fim do deque, uma que realiza a remoção de um elemento no início do deque e uma que realiza a remoção de um elemento no fim do deque.

Exercícios

20.8 Suponha que exista um único vetor M de células de um tipo fila pré-definido, com um total de **MAX** posições. Este vetor fará o papel da memória do computador. Este vetor M será compartilhado por duas filas em alocação seqüencial. Implemente eficientemente as operações de enfileiramento e desenfileiramento para as duas filas de modo que nenhuma delas estoure sua capacidade de armazenamento, a menos que o total de elementos em ambas as filas seja **MAX**.