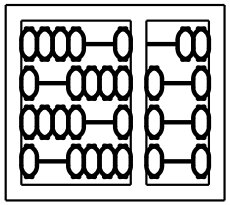
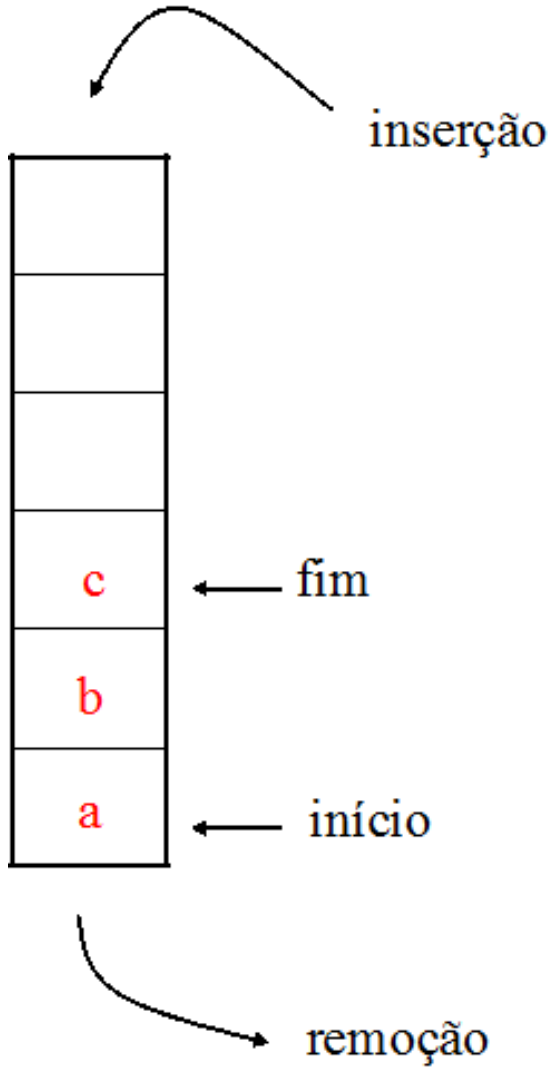


Universidade Estadual de Campinas - UNICAMP
Instituto de Computação - IC

Filas

• Definição: Uma fila F é uma lista linear em que as **inserções** são feitas em um extremo (fim) e as **remoções** em outro (início).



- Operações sobre o TAD fila:
  - Criar uma fila vazia
  - Inserir um elemento
  - remover um elemento
  - verificar se a fila está vazia
  - verificar se a fila está cheia
  - liberar a fila
  - ...

- Operações sobre o TAD fila:

– Criar uma fila vazia

– Inserir um elemento

– remover um elemento

– verificar se a fila está vazia

– verificar se a fila está cheia

– liberar a fila

– ...

- Operações sobre o TAD fila:

– Criar uma fila vazia

– Inserir um elemento

– remover um elemento

– verificar se a fila está vazia

– verificar se a fila está cheia

– liberar a fila

– ...

- Operações sobre o TAD fila:

– Criar uma fila vazia

– Inserir um elemento

– remover um elemento

– verificar se a fila está vazia

– verificar se a fila está cheia

– liberar a fila

– ...

- Operações sobre o TAD fila:

– Criar uma fila vazia

– Inserir um elemento

– remover um elemento

– verificar se a fila está vazia

– verificar se a fila está cheia

– liberar a fila

– ...

- Operações sobre o TAD fila:

– Criar uma fila vazia

– Inserir um elemento

– remover um elemento

– verificar se a fila está vazia

– verificar se a fila está cheia

– liberar a fila

– ...

- Operações sobre o TAD fila:

– Criar uma fila vazia

– Inserir um elemento

– remover um elemento

– verificar se a fila está vazia

– verificar se a fila está cheia

– liberar a fila

– ...

Implementação com vetor

- Exemplo de estrutura:

```
#define MAX  1000

typedef struct {
 int inicio, fim ;
 int elemento[MAX] ;
} Fila ;
```

Exemplo de funções

- Criar fila:

```
Fila *fila_cria(void) {

 Fila *p = (Fila *)malloc(sizeof(Fila)) ;
 p->inicio = 0 ;
 p->fim = 0 ;

 return p ;
}
```

Exemplo de funções

- Inserir na fila:

```
void fila_insere(Fila *p, int x) {

 if (p->inicio == ((p->fim + 1) % MAX)) {   /* Fila cheia */
   printf(''Fila cheia\n'') ;
   exit(1)                                  /* aborta programa */
 }

 else {
   p->elemento[p->fim] = x ;          /* insere no fim */
   p->fim = (p->fim + 1) % MAX ;      /* incrementa fim */
 }
}
```

Exemplo de funções

- Verificar se a fila está vazia:

```
boolean fila_vazia(Fila *p) {

 return (p->inicio == p->fim) ;

}
```

Exemplo de funções

- Remover da fila:

```
int fila_remove(Fila *p) {
 int x ;

 if (fila_vazia(p)) {
  printf(''Fila vazia \n'') ;
  exit(1) ;          /* aborta programa */
 }
 else {
  x = p->elemento[p->inicio] ;          /* remove do incio */
  p->inicio = (p->inicio + 1)%MAX ;  /* incrementa inicio */
 }

 return (x) ;
}
```

Exemplo de funções

- Verificar se a fila está cheia:

```
boolean fila_cheia(Fila *p) {

 return(p->inicio == ((p->fim + 1) % MAX)) ;

}
```

Implementação de fila com lista

- A estrutura da fila:

```
typedef struct NoFila {
 int info ;
 struct NoFila *prox ;
} Fila, *ApRegFila, RegFila ;
```

Exemplo de funções

- Criar a fila com nó cabeça e circular.

```
Fila *fila_cria(void) {

 Fila *p = (Fila *)malloc(sizeof(RegFila)) ;
 p->prox = p ;
 p->info = -1 ;

 return p ;
}
```

Exemplo de funções

• Inserir elemento na fila (o apontador p da fila aponta para o último nó):

```
void fila_insere(Fila **p, int x) {

 ApRegFila q ;
 q = (RegFila *)malloc(sizeof(RegFila)) ;
 q->info = x ;
 q->prox = (*p)->prox ;
 (*p)->prox = q ;
 *p = q ;
}
```

Exemplo de funções

- Remover elemento da fila:

```
int  fila_retira(Fila **p) {
 ApRegFila q, r ; int x ;
 q = (*p)->prox ;
 if(*p == q)  {
  printf(''Fila vazia \n'') ;
  exit(1) ;    /* aborta programa */
 }
 else {
  r = q->prox ; x = r->info ;
  q->prox = r->prox ;
  if (r == *p)          /* ultimo elemento removido? */
   *p = q ;
  free(r) ;
  return x ;
 }
}
```

Exemplo de funções

- Verificar se a fila está vazia:

```
boolean fila_vazia(Fila *p) {

 return(p == p->prox) ;
}
```

Exemplo de funções

- Liberar fila:

```
void fila_libera(Fila *p) {
 ApRegFila q, r ;

 q = p->prox ;
 while(q != p) {
  r = q->prox ;
  free (q) ;
  q = r ;
 }

 free (p) ;
}
```

Implementação com vetor: outras considerações

• Algumas implementações de filas **não circulares** com vetor podem ser feitas diretamente considerando-se que:

Implementação com vetor: outras considerações

• Algumas implementações de filas **não circulares** com vetor podem ser feitas diretamente considerando-se que:

1. Uma fila está armazenada em um segmento $p[i..f-1]$ de um vetor $p[0..N-1]$, com $0 \leq i \leq f \leq N$.

Implementação com vetor: outras considerações

• Algumas implementações de filas **não circulares** com vetor podem ser feitas diretamente considerando-se que:

1. Uma fila está armazenada em um segmento $p[i..f-1]$ de um vetor $p[0..N-1]$, com $0 \leq i \leq f \leq N$.
2. O primeiro elemento da fila está na posição i e o último na posição $f-1$

Implementação com vetor: outras considerações

• Algumas implementações de filas **não circulares** com vetor podem ser feitas diretamente considerando-se que:

1. Uma fila está armazenada em um segmento $p[i..f-1]$ de um vetor $p[0..N-1]$, com $0 \leq i \leq f \leq N$.
2. O primeiro elemento da fila está na posição i e o último na posição $f-1$
3. A fila está vazia se i é igual a f e cheia se f é igual a N.

Implementação com vetor: outras considerações

• Algumas implementações de filas **não circulares** com vetor podem ser feitas diretamente considerando-se que:

1. Uma fila está armazenada em um segmento $p[i..f-1]$ de um vetor $p[0..N-1]$, com $0 \leq i \leq f \leq N$.
2. O primeiro elemento da fila está na posição i e o último na posição $f-1$
3. A fila está vazia se i é igual a f e cheia se f é igual a N.
4. Remover um elemento da fila não vazia corresponde a: $x = p[i++]$;

Implementação com vetor: outras considerações

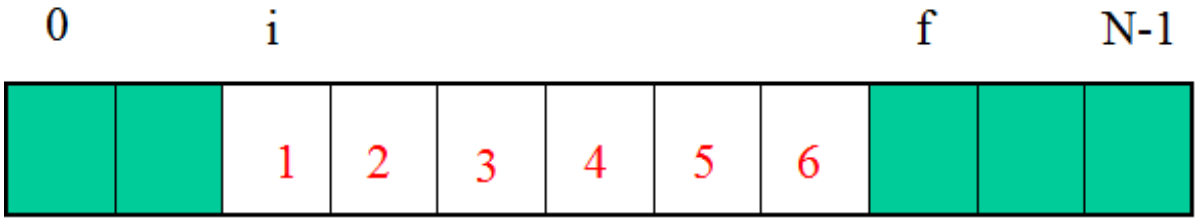
• Algumas implementações de filas **não circulares** com vetor podem ser feitas diretamente considerando-se que:

1. Uma fila está armazenada em um segmento $p[i..f-1]$ de um vetor $p[0..N-1]$, com $0 \leq i \leq f \leq N$.
2. O primeiro elemento da fila está na posição i e o último na posição $f-1$
3. A fila está vazia se i é igual a f e cheia se f é igual a N.
4. Remover um elemento da fila não vazia corresponde a: $x = p[i++]$;
5. Inserir um elemento na fila não cheia corresponde a: $p[f++] = y$;

Implementação com vetor: outras considerações

• Algumas implementações de filas **não circulares** com vetor podem ser feitas diretamente considerando-se que:

1. Uma fila está armazenada em um segmento $p[i..f-1]$ de um vetor $p[0..N-1]$, com $0 \leq i \leq f \leq N$.
2. O primeiro elemento da fila está na posição i e o último na posição $f-1$
3. A fila está vazia se i é igual a f e cheia se f é igual a N.
4. Remover um elemento da fila não vazia corresponde a: $x = p[i++]$;
5. Inserir um elemento na fila não cheia corresponde a: $p[f++] = y$;



Implementação com vetor: outras considerações

• Algumas implementações de filas **circulares** com vetor podem ser feitas diretamente considerando-se que:

Implementação com vetor: outras considerações

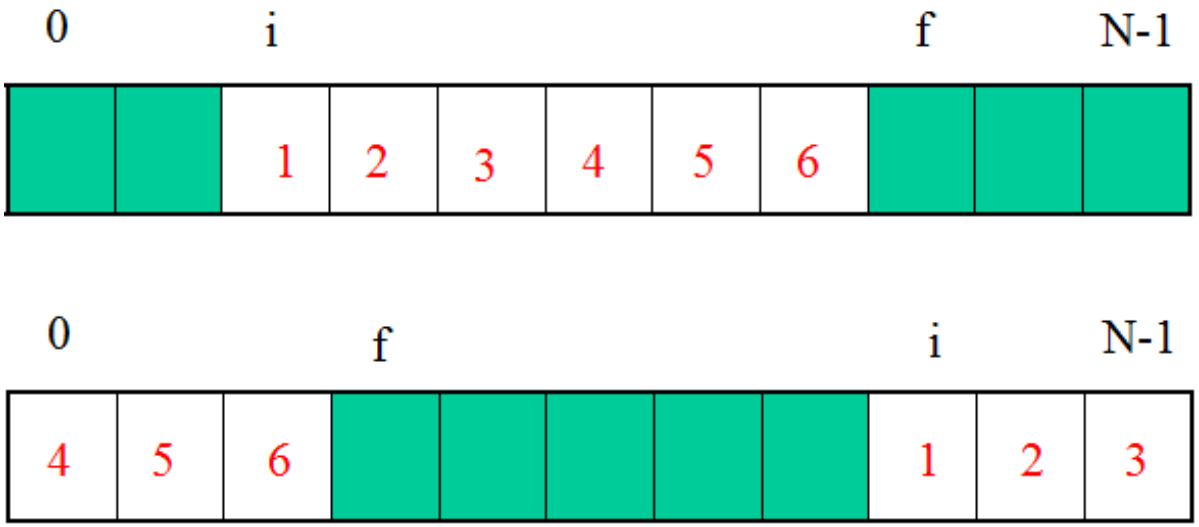
• Algumas implementações de filas **circulares** com vetor podem ser feitas diretamente considerando-se que:

- Para o vetor $p[0..N-1]$ da fila circular, os seus elementos estão dispostos da seguinte maneira:
  - $p[i..f-1]$, como anteriormente,
  - ou $p[i..N-1]p[0..f-1]$

Implementação com vetor: outras considerações

• Algumas implementações de filas **circulares** com vetor podem ser feitas diretamente considerando-se que:

- Para o vetor $p[0..N-1]$ da fila circular, os seus elementos estão dispostos da seguinte maneira:
  - $p[i..f-1]$, como anteriormente,
  - ou $p[i..N-1]p[0..f-1]$



- A fila está vazia se $i = f$
- A fila está cheia se $i = f + 1$ ou $f + 1 = N$ e $i = 0$, ou seja, se $(f + 1)\ \%\ N = i$
- Remover um elemento da fila não vazia corresponde a: $x = p[i++];\ \ if(i == N)\ i = 0;$
- Inserir um elemento na fila não cheia corresponde a: $p[f++] = y;\ \ if(f == N)\ f = 0;$
- A posição f ficará sempre desocupada para que se possa distinguir uma fila vazia de uma fila cheia

Exemplo de aplicação

- Distâncias entre pontos de uma rede:

– n cidades numeradas de 0 a $n-1$ e interligadas por estradas de mão única.

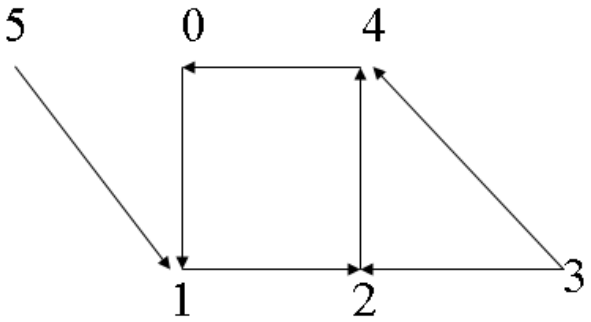
– A distância de uma cidade o a uma cidade x é o menor número de estradas que é preciso para ir de o a x.

– Determinar a distância de uma dada cidade o a cada uma das outras cidades.

Exemplo de aplicação

A

	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	-1

origem $o = 3$

Exemplo de aplicação

- Programa:

```
int *Distancias(int **A, int n, int o){
  int *d, x, y ;
  int *f, s, t ;
  d = (int *)malloc(n*sizeof(int)) ;
  for (x = 0; x < n; x++) d[x] = -1 ;
  d[o] = 0 ;
  f =  (int *)malloc(n*sizeof(int)) ;
  s = 0 ; t = 1 ; f[s] = o ;
```

Exemplo de aplicação

```
  while (s < t) {
    /* f[s..t-1] eh uma fila de cidades */
    x = f[s++] ;
    for (y = 0 ; y < n ; y++)
       if (A[x][y] == 1 && d[y] == -1)  {
         d[y] = d[x] + 1 ;
         f[t++] = y ;
       }
   }
  free (f)
  return d ;    /* retorna vetor de distancias */
}
```

- Exemplo de aplicação

– Ordenação por distribuição

- Bibliografia:

– Paulo Feofiloff. Algoritmos em linguagem C. Elsevier 2009

– Apostila dos profs. Tomasz e Lucchesi disponível na página do curso