BC1424
Algoritmos e Estruturas de Dados I

Aula 09:
Exercícios de Filas e Listas

Prof. Jesús P. Mena-Chalco
jesus.mena@ufabc.edu.br

1Q-2015

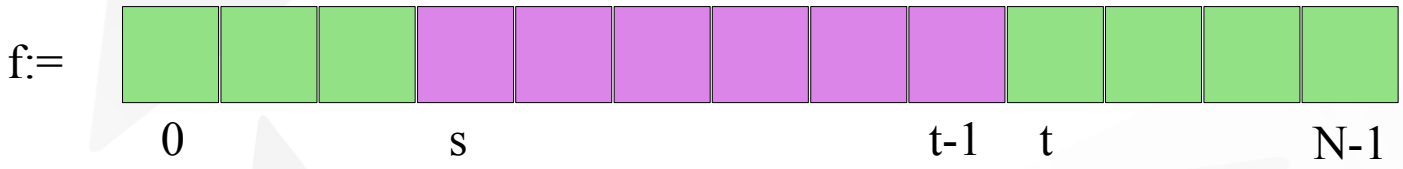
Fila: Implementação em vetor

Uma fila (sequência dinâmica) pode ser armazenada em um segmento

`f[s..t-1]`

de um vetor

`f[0..N-1]`



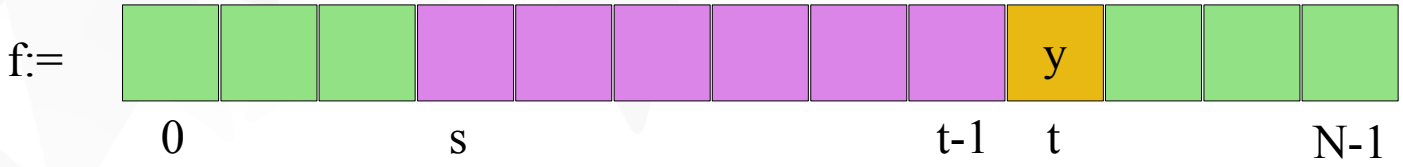
Fila: Implementação em vetor

Para **remover** um elemento:

```
x = f[s];
s = s+1;
```

```
x = f[s++];
```

Fila: Implementação em vetor



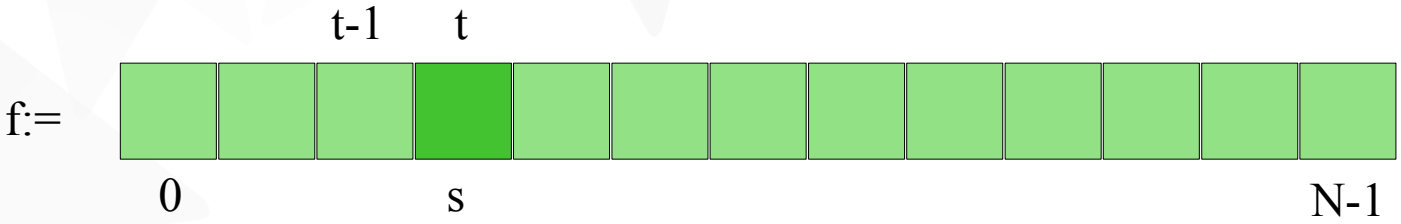
Para **inserir** o elemento **y** na fila **f**:

```
f[t] = y;
t = t+1
```

```
f[t++] = y;
```

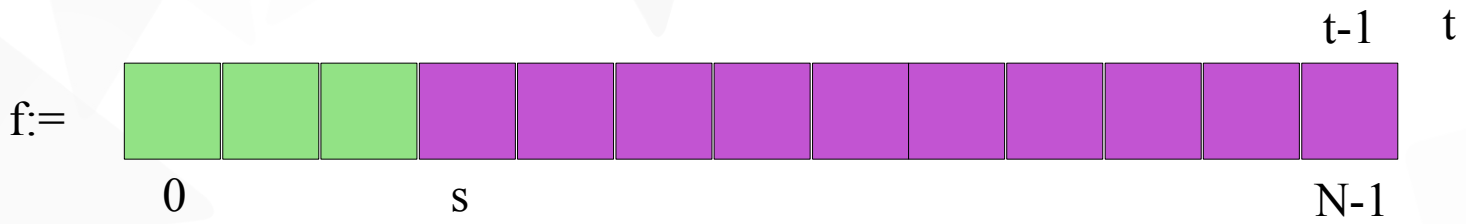
Fila: Alguns cuidados

- **Remover um elemento** em uma **fila vazia** (quando `s==t`)



Fila: Alguns cuidados

- **Inserir um elemento** em uma **fila cheia** (`t==N`)



Transbordamento em fila: *evento excepcional (mau planejamento lógico)*

Em outras linguagens de programação

- Inserir (colocar): **enqueue**
- Remover (retirar): **de-queue**

testeFila01.c (Tidia)

```c
void Inserir(int y, int fila[], int *s, int *t, int n) {
    if (*t==n)
        printf("lista cheia\n");
    else
        fila[(*t)++] = y;
}
```

```c
int Remover(int fila[], int *s, int *t) {
    if (*s==*t)
        printf("lista vazia\n");
    else
        return fila[(*s)++];
}
```

testeFila01.c (Tidia)

```c
int main() {
    int MAX=5;
    int fila[MAX];
    int s=0;
    int t=0;

    Inserir(100, fila, &s, &t, MAX);
    Inserir(200, fila, &s, &t, MAX);
    Inserir(300, fila, &s, &t, MAX);
    Inserir(400, fila, &s, &t, MAX);

    printf("remover: %d\n", Remover(fila, &s, &t));
    printf("remover: %d\n", Remover(fila, &s, &t));
    printf("remover: %d\n", Remover(fila, &s, &t));

    Inserir(500, fila, &s, &t, MAX);
    Inserir(900, fila, &s, &t, MAX);

    imprimirFila(fila, MAX, &s, &t);
}
```

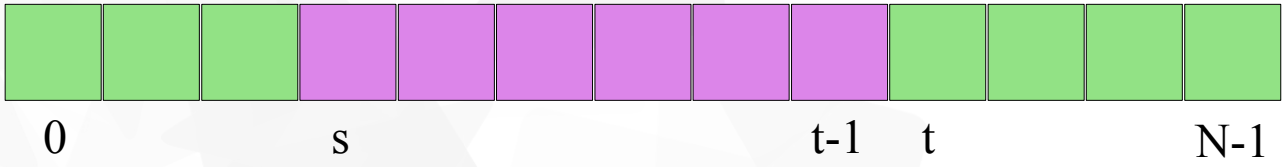
testeFila01.c (Tidia)

```
remover: 100
remover: 200
remover: 300
lista cheia
s=3 t=5
0: 100  1: 200  2: 300  3: 400  4: 500
```

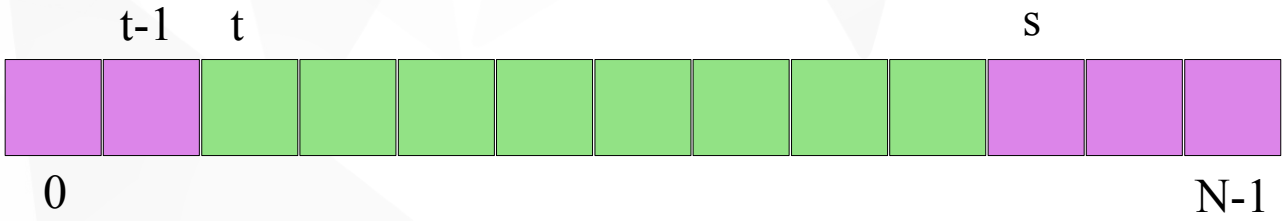
```c
int main() {
    int MAX=5;
    int fila[MAX];
    int s=0;
    int t=0;

    Inserir(100, fila, &s, &t, MAX);
    Inserir(200, fila, &s, &t, MAX);
    Inserir(300, fila, &s, &t, MAX);
    Inserir(400, fila, &s, &t, MAX);

    printf("remover: %d\n", Remover(fila, &s, &t));
    printf("remover: %d\n", Remover(fila, &s, &t));
    printf("remover: %d\n", Remover(fila, &s, &t));

    Inserir(500, fila, &s, &t, MAX);
    Inserir(900, fila, &s, &t, MAX);

    imprimirFila(fila, MAX, &s, &t);
}
```

Fila Circular

Fila circular

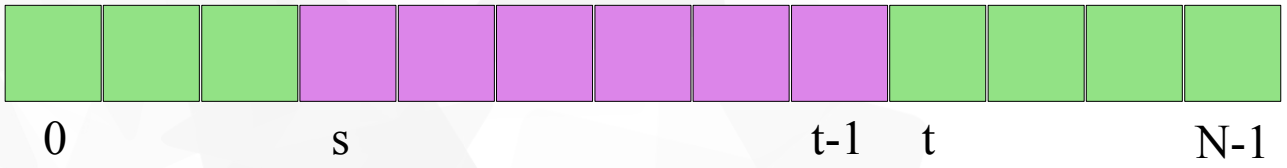


`f[s..t-1]`

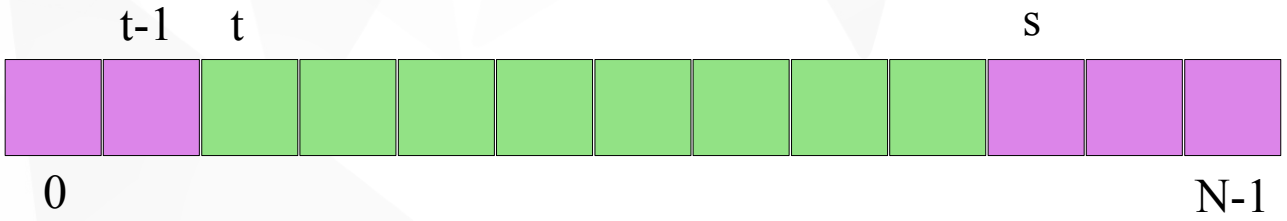


`f[s..N-1]f[0..t-1]`

Fila circular



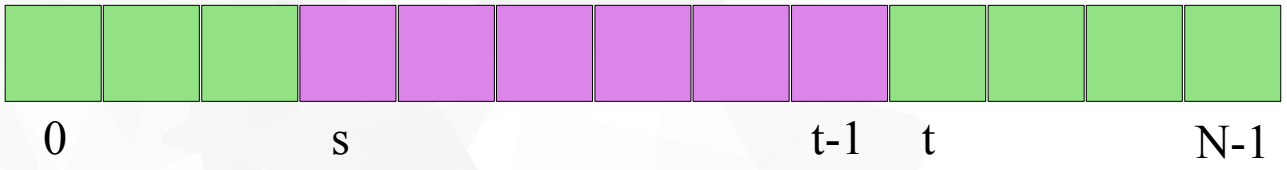
`f[s..t-1]`



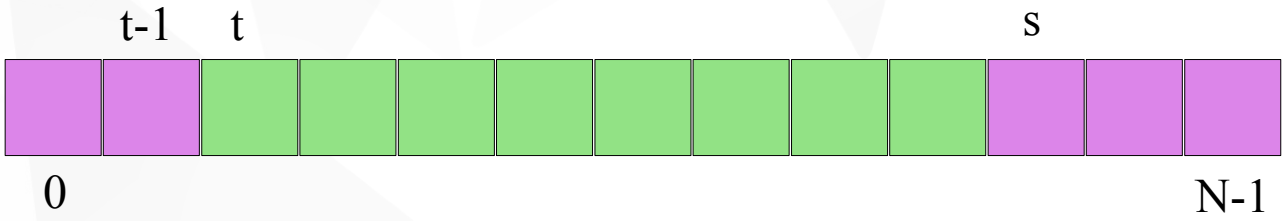
`f[s..N-1]f[0..t-1]`

- **Fila vazia se:** `t==s`
- **Fila cheia se:** `t+1=s` **ou** `(t+1=N && s==0)`

Fila circular



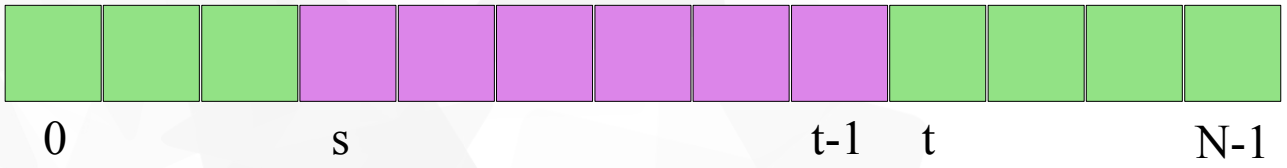
`f[s..t-1]`



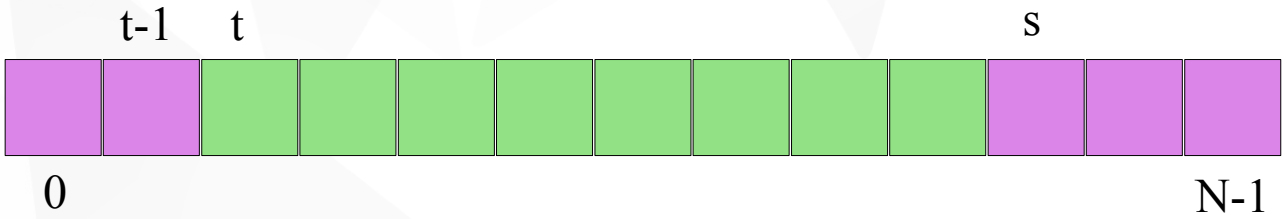
`f[s..N-1]f[0..t-1]`

- **Fila vazia se:** `t==s`
- **Fila cheia se:** `t+1=s` **ou** `(t+1=N && s==0)` → `t+1%N==s`

Fila circular



`f[s..t-1]`



`f[s..N-1]f[0..t-1]`

- **Fila vazia se:** `t==s`
- **Fila cheia se:** `t+1=s` **ou** `(t+1=N && s==0)` → `t+1%N==s`

Modifique as funções Inserir e Remover (exercício anterior) para que se considere uma lista circular.

Fila circular: testeFila02.c (Tidia)

```
void Inserir(int y, int fila[], int *s, int *t, int n) {
    if (((*t)+1)%n==*s)
        printf("lista cheia\n");
    else {
        fila[(*t)++] = y;
        if (*t==n)
            *t = 0;
    }
}
```

```
int Remover(int fila[], int *s, int *t, int n) {
    if (*s==*t)
        printf("lista vazia\n");
    else {
        int y = fila[(*s)++];
        if (*s==n)
            *s = 0;
        return y;
    }
}
```

Fila circular: testeFila02.c (Tidia)

```
Inserir(100, fila, &s, &t, MAX);
Inserir(200, fila, &s, &t, MAX);
Inserir(300, fila, &s, &t, MAX);
Inserir(400, fila, &s, &t, MAX);

printf("remover: %d\n", Remover(fila, &s, &t, MAX));
printf("remover: %d\n", Remover(fila, &s, &t, MAX));
printf("remover: %d\n", Remover(fila, &s, &t, MAX));

Inserir(500, fila, &s, &t, MAX);
Inserir(900, fila, &s, &t, MAX);

imprimirFila(fila, MAX, &s, &t);
```

```
Running /home/ubuntu/workspace/aed1-09/testeFila02.c
remover: 100
remover: 200
remover: 300
s=3 t=1
0: 900  1: 200  2: 300  3: 400  4: 500
```

Sobre a Prova 1

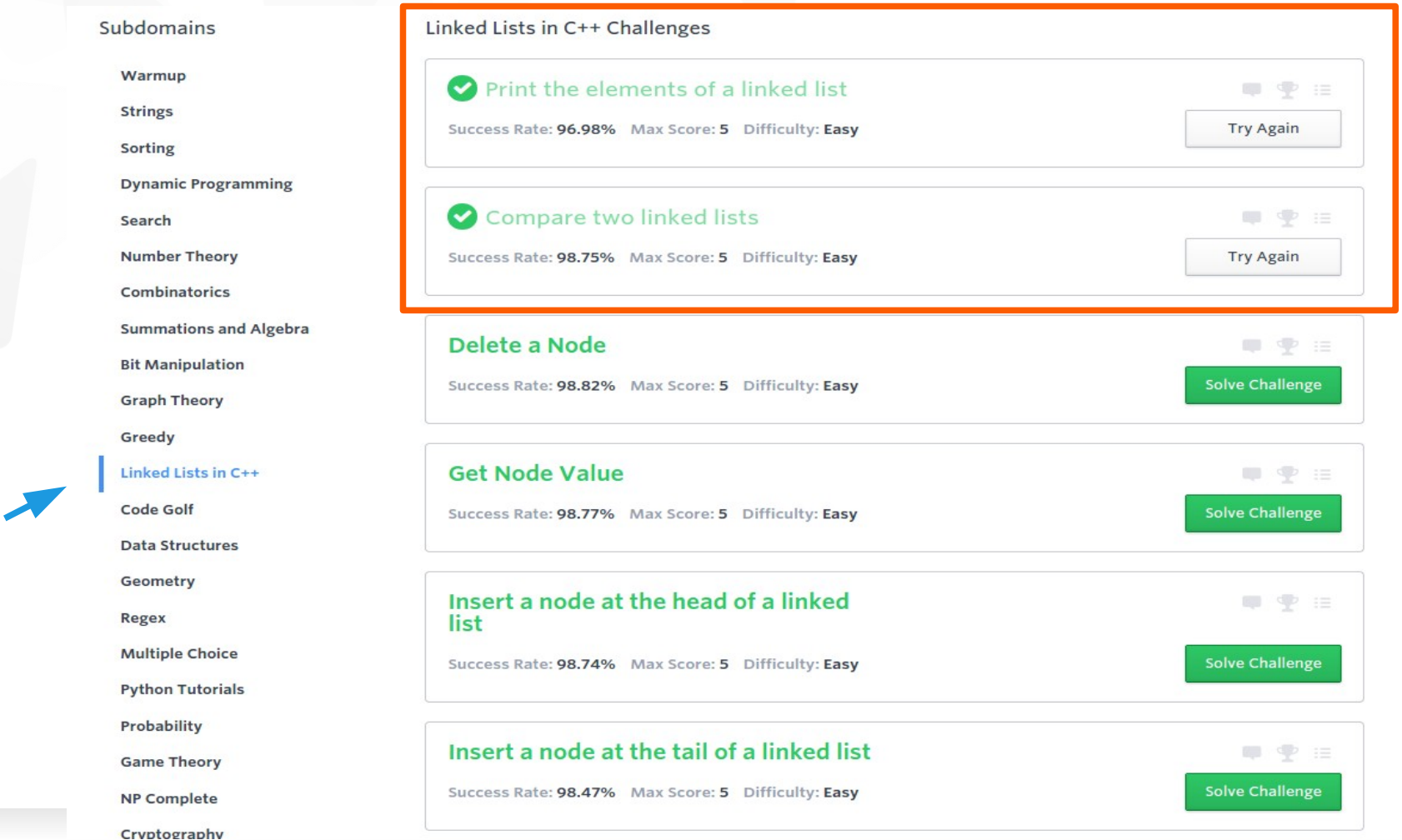
Sobre a Prova 1:
Exercícios propostos para estudo

Livro texto: **“Algoritmos em linguagem C”**, Paulo Feofiloff (2009).

- **Cap. 2:** 2.2.2, 2.2.3, 2.2.4, 2.2.5, 2.3.1, 2.3.4, 2.3.8, 2.3.10
- **Cap. 3:** 3.1.3, 3.1.4, 3.2.1, 3.3.4, 3.4.1
- **Cap. 4:** 4.3.1, 4.3.2, 4.3.5, 4.4.1, 4.5.1, 4.5.2, 4.6.4, 4.7.7, 4.7.11, 4.8.1, 4.8.2, 4.8.3
- **Cap. 5:** 5.1.1, 5.4.1
- **Ap. D:** D.3.1, D.3.3, D.4.1, D.4.2, D.4.3
- **Ap. E:** E.2.1

Atividade opcional até 15/03 23h50 (Tidia)

- **HackerRank:** Faça os 13 problemas do subdomínio “Linked lists in C++”. Cada problema vale 0.1 na nota da Prova 01.
- O aluno que completar todos os exercícios recebe bonus de +0.7.



Veremos agora! (não entram na conta)

Será utilizado um programa para detecção de plágio

(1) Print the elements of a linked list

```
struct Node
{
    int data;
    Node *next;
};
```

```
void Print(Node *head) {
    // This is a "method-only" submission.
    // You only need to complete this method.
    Node *p = head;
    while(p!=NULL) {
        printf("%d\n", p->data);
        p = p->next;
    }

}
```

(2) Compare two linked lists

```
struct Node
{
    int data;
    Node *next;
};
```

```
int CompareLists(Node *headA, Node *headB) {
    Node *pA = headA;
    Node *pB = headB;

    while(pA!=NULL && pB!=NULL){
        if (pA->data==pB->data) {
            pA = pA->next;
            pB = pB->next;
        }
        else
            return 0;
    }
    if (pA==NULL && pB==NULL)
        return 1;
    else
        return 0;
}
```