صف (Queue)

قاسم مهدور (mahdevar@ibb.ut.ac.ir)

صف (Queue)

- تعریف:

ساختمان داده خطی می‌باشد که عنصر جدید فقط به انتهای آن افزودن می‌شود و حذف عناصر از ابتدای آن صورت می‌پذیرد.

مثال:

$X_0, X_1, \ldots, X_n$

- در صف فوق عنصر X_n در انتهای صف قرار دارد. می‌توان این عنصر را حذف را نمود و عنصر X_0 در ابتدای صف قرار دارد.
- اندازه صف خالی 0 است.
- به اولین عنصر صف front و آخرین آنها rear می‌گوییم.
- صف، در حقیقت یک فهرست است که اولین ورودی آن اولین خروجی آن است (به چنین سیاستی برای ورود و خروج عناصر، اولین ورودی- اولین خروجی (First Input – First Output (FIFO)) می‌گویند).

فهرست مطالب

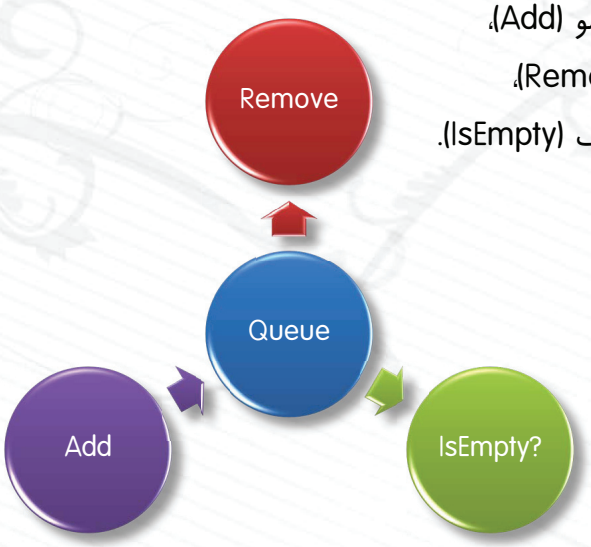
اشیا داخل صف

- می‌توان یک صف از
  - اعداد صحیح، اعشاری، دودویی، ....
  - رشته‌های حرفی، یا
  - انواع داده‌های دیگر(؟)

ساخت.

اعمالی که می‌توان روی صف انجام داد

1. ایجاد (Create)،
2. اضافه نمودن یک عضو (Add)،
3. حذف یک عضو (Remove)،
4. تعیین خالی بودن صف (IsEmpty).



5

مثال

- یک صف از اعداد صحیح

1, 2, 3, 4

- نتیجه اجرای برخی از اعمال روی این صف

Remove() ⟶ 1

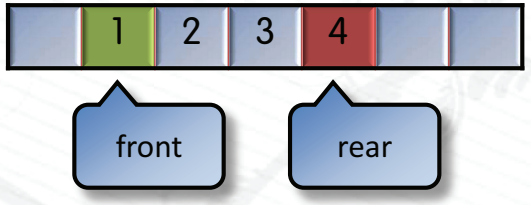
Add(5) ⟶ 2, 3, 4, 5

IsEmpty() ⟶ FALSE

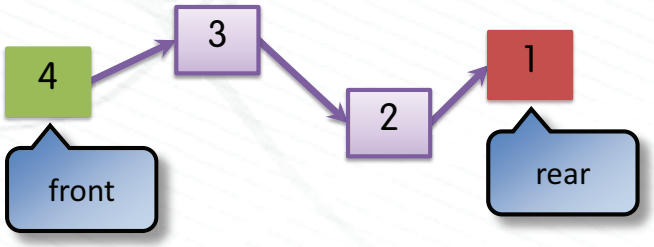
6

روش‌های پیاده سازی صف

- با استفاده از آرایه



- با استفاده از فهرست پیوندی



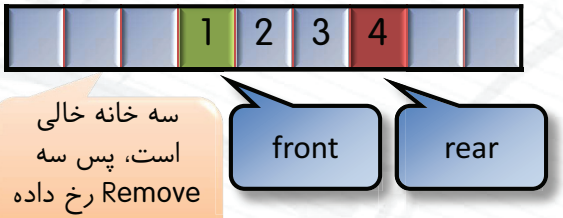
7

پیاده سازی با استفاده از آرایه

- پیاده سازی صف با استفاده از آرایه
- چگونگی اضافه نمودن به صف
- نحوه حذف از صف

پیاده سازی صف با استفاده از آرایه

- برای انجام این کار به یک آرایه، یک متغیر برای دانستن اندیس شروع صف (front)، و یک متغیر دیگر جهت دانستن اندیس انتهای صف (rear) نیاز است.



سه خانه خالی است، پس سه Remove رخ داده است.

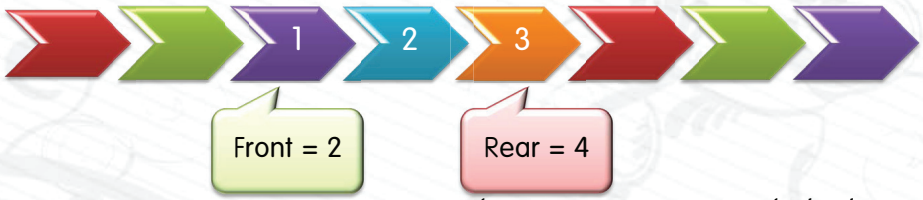
- ویژگی‌های این روش:
  - بعد از چند اضافه و حذف نمودن از صف، front و rear به انتهای آرایه نزدیک می‌شوند. برای نمونه، سه Remove در صف فوق رخ داده است.(با حلقوی نمودن می‌توان این نقیصه را مرتفع نمود ☺).
  - باید بیشترین اندازه صف معلوم باشد ☹.

پیاده سازی ساختار صف در C

```
#define MAX 100
struct Queue
{
    int Front;
    int Rear;
    int Data[MAX];
};

void main()
{
    Queue A;
    A.Front = 0;
    A.Rear = 0;
    Add(A, 1);
}
```


چگونگی اضافه نمودن به صف (Add)



- برای اضافه نمودن عنصر جدید باید
  - به متغیر rear یک واحد افزوده شود، و
  - مقدار جدید در مکان rear نوشته شود.

الگوریتم اضافه نمودن به صف

```
void Add(Queue *A, int d)
{
    A->Rear++;
    A->Data[A->Rear] = d;
}
```

- آیا ممکن است این الگوریتم با خطا مواجه شود؟ اگرجواب بلی هست، که هست چگونه می‌توان آن را حل نمود؟

نحوه حذف عنصر ابتدای(Remove)

```
void Remove(Queue *A)
{
A->Front++;
}
```

- آیا ممکن است این الگوریتم درست عمل نکند؟ اگرجواب بلی است چگونه می‌توان آن را تصحیح نمود؟

چگونگی تشخیص خالی بودن صف (IsEmpty)

- در ابتدا مقدار Front و Rear را برابر 0 قرار می‌دهیم؛ از آنجا که
  - با اضافه نمودن عنصر جدید متغیر Rear بعلاوه یک می‌شود و
  - با حذف عنصر، یک واحد از متغیر Front کاهش می‌یابد،

بنابراین هرگاه این متغیر برابر 1- باشد صف خالی است.

```
int IsEmpty(Queue *A)
{
return (A->Front == A->Rear ? 1 : 0);
}
```


پیچیدگی اعمال مختلف روی صف ایجاد شده با آرایه

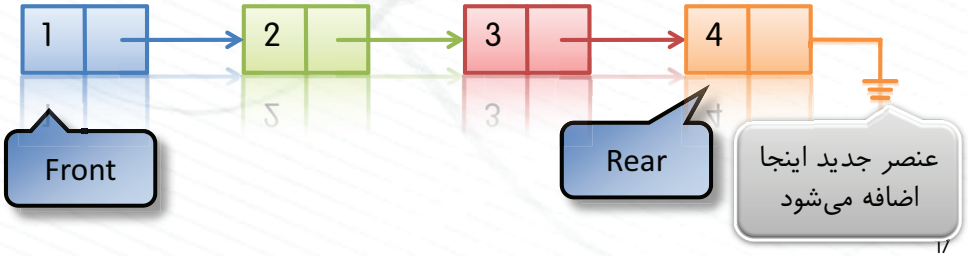
1. ایجاد (Create)، O(1) ←
2. اضافه نمودن یک عضو (Add)، O(1) ←
3. حذف یک عضو (Remove)، O(1) ←
4. تعیین خالی بودن صف (IsEmpty) O(1) ←

پیاده سازی با استفاده از فهرست پیوندی

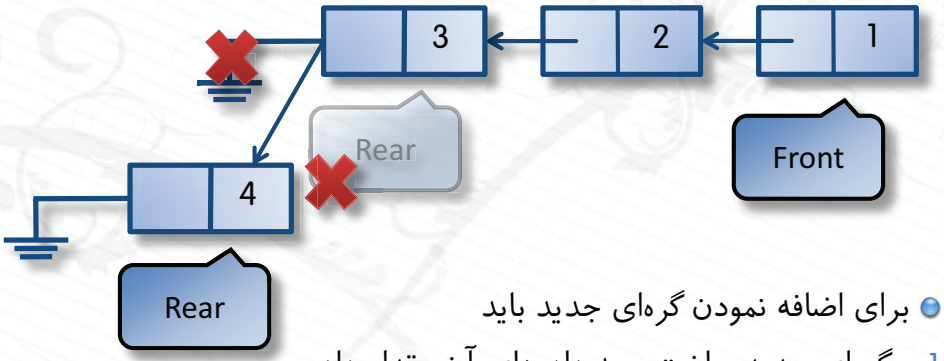
- پیاده سازی صف با استفاده از فهرست پیوندی
- چگونگی اضافه نمودن به صف
- نحوه حذف از صف

پیاده سازی صف با استفاده از فهرست پیوندی

- برای انجام این کار ما به یک فهرست پیوندی احتیاج داریم و دو اشاره‌گر برای دانستن نشانی ابتدا و انتهای صف آن.
- فهرست را چنان پیاده سازی می‌کنیم که
  1. هر گره به گره قبلی اشاره می‌کند،
  2. Rear به عنصر اول اشاره می‌کند
  3. Top همان head فهرست پیوندی باشد.



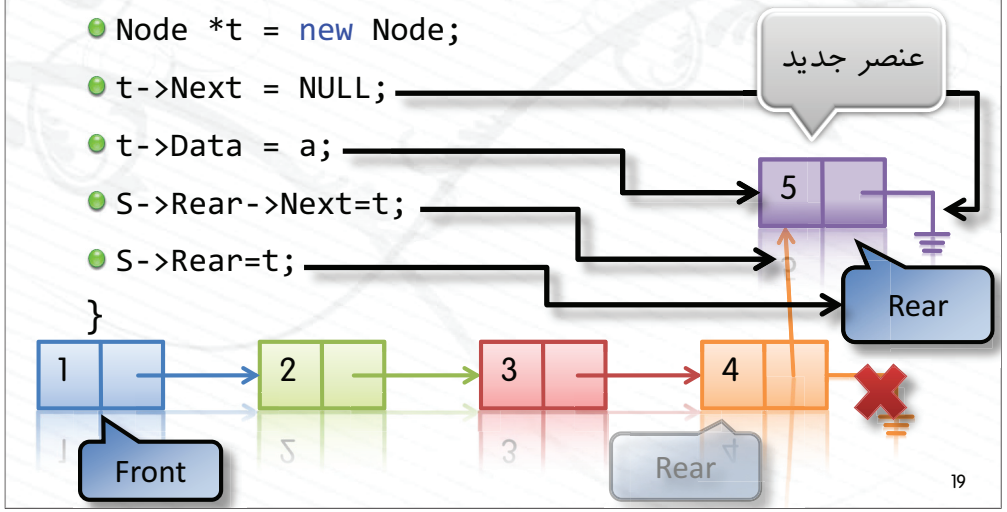
چگونگی اضافه نمودن به صف (Add)



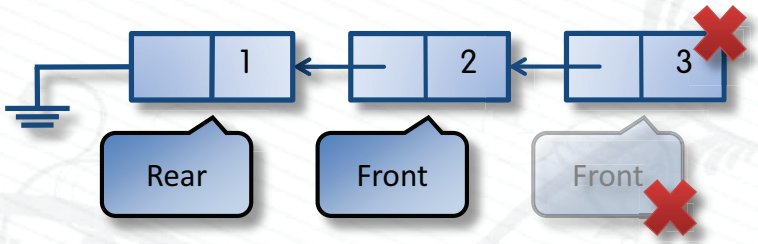
- برای اضافه نمودن گره‌ای جدید باید
  1. گره‌ای جدید ساخت و به داده‌های آن مقدار داد،
  2. اشاره‌گر گره جدید به Rear صف اشاره کند، و
  3. Rear صف برابر این گره جدید شود.

الگوریتم اضافه نمودن به صف

```
void Add(Queue *S, int a)
{
  Node *t = new Node;
  t->Next = NULL;
  t->Data = a;
  S->Rear->Next=t;
  S->Rear=t;
}
```



نحوه حذف عنصر Front (Remove)



- برای حذف عنصر Front باید
  1. نشانی گره Front را ذخیره نمود،
  2. گره Front را برابر Front->Next قرار داد، و
  3. نشانی ذخیره شده را از حافظه پاک نمود.

الگوریتم حذف عنصر Front (Remove)

```
void Remove(Queue *S)
{
  Node *t;
  t = S->Front;
  S-> Front = S-> Front ->Next;
   // or S->Front = t->Next;
  free(t);
}
```


فهرست پیوندی – زمان مورد نیاز اعمال مختلف

1. ایجاد (Create)، ← O(1)
2. اضافه نمودن یک عضو (Add)، ← O(1)
3. حذف یک عضو (Pop)، ← O(1)
4. دریافت مقدار بالای صف (Top)، ← O(1)
5. تعیین خالی بودن صف (IsEmpty) ← O(1)