ALGORITMI I STRUKTURE PODATAKA

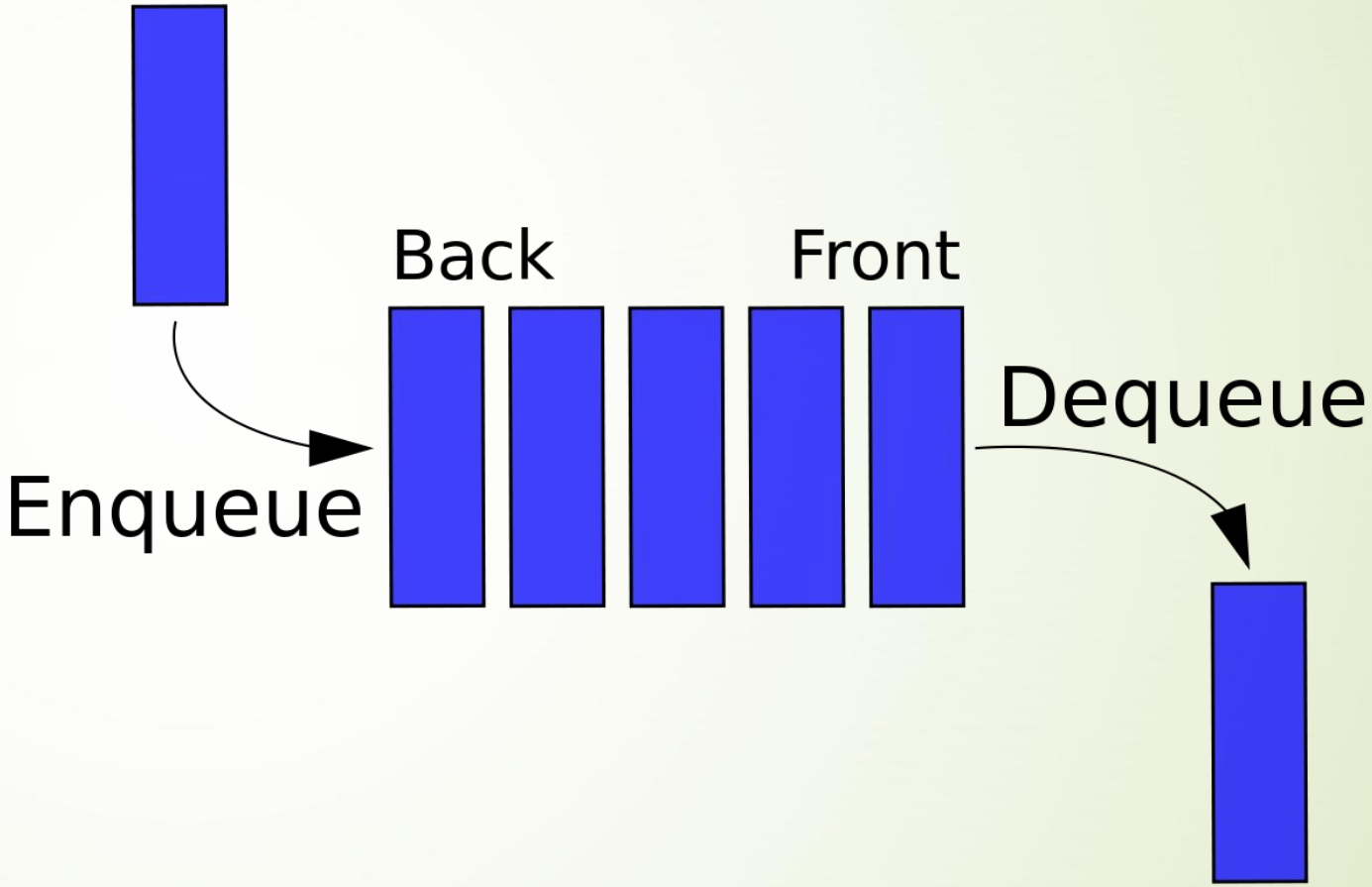
RAČUNSKE VEŽBE - TERMIN BR. 7 - RED

ALDINA AVDIĆ, DIPL. INŽ. - apljaskovic@np.ac.rs

RAČUNARSKA TEHNIKA, SOFTVERSKO INŽENJERSTVO, INFORMATIKA I MATEMATIKA

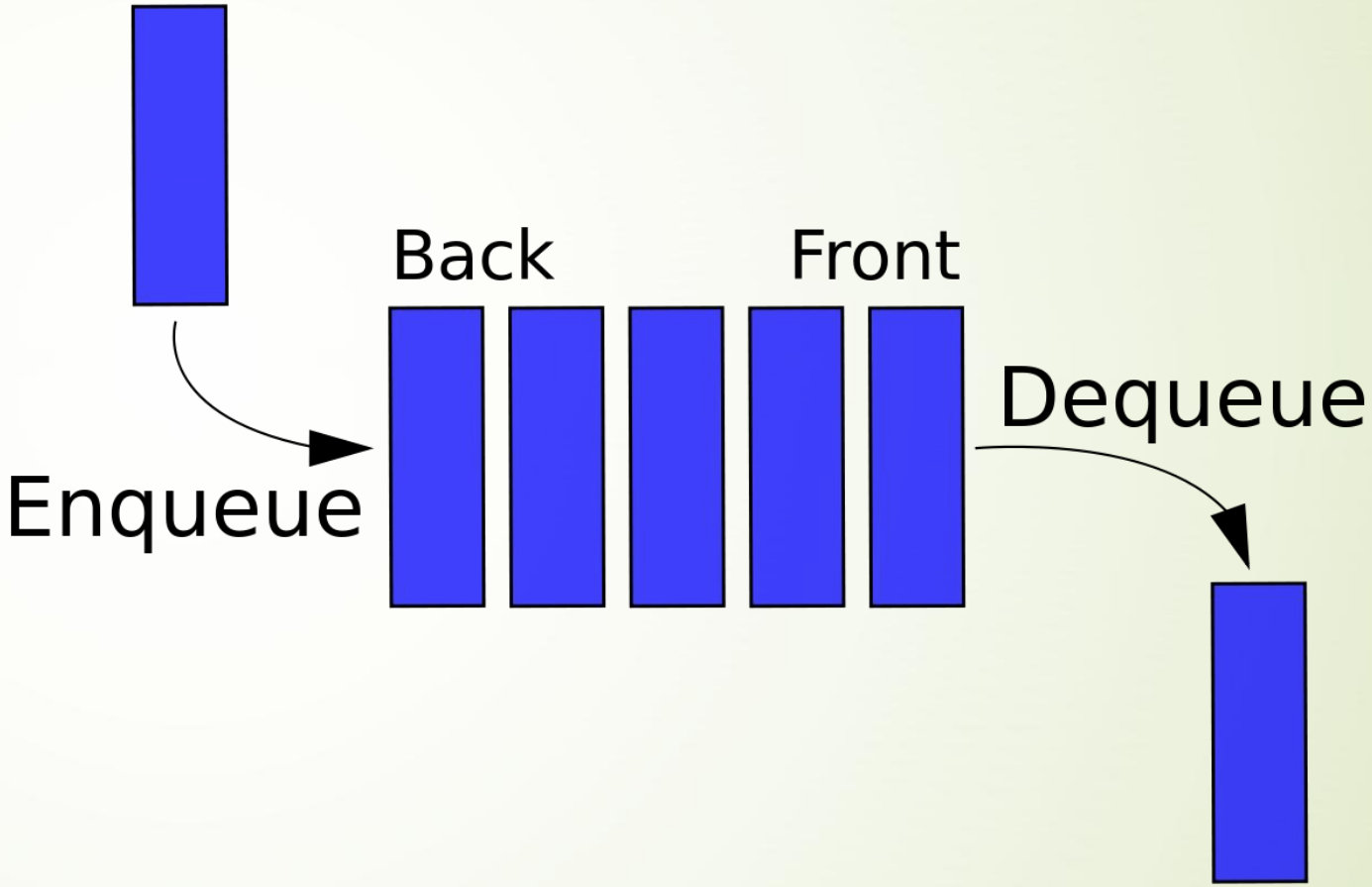
Red

- **FIFO** - First In First Out struktura podataka
- **Enqueue** - dodavanje elementa u red (na kraj)
- **Dequeue** - brisanje elementa iz reda (s početka)



Red

- Zamislite sada paket koji može da se otvori na obe strane
- Ne možete pristupiti knjigama koje su u sredini, ali možete onoj koja je na početku, i onoj koja je na kraju



Implementacija kružnog reda preko niza

Ispitivanje da li je red pun

```
#define SIZE 5
int CQ[SIZE],f=-1,r=-1;
int  CQfull()
{                          /* Function to
Check Circular Queue Full */
/* proverava se pre povećanja r*/
    if( (f==r+1) || (f == 0 && r==
SIZE-1)) return 1;
    return 0;
}
```

- Definišemo konstantu za veličinu niza dužine 5
- Zatim CQ niz (circular queue)
- F za front (glava)
- R za rear (rep)
- Funkcija ispituje jel red pun, red je puno ako je razlika između fronta i reara 1, ili ako je front 0, a rear 4

Implementacija kružnog reda preko niza

Ispitivanje da li je red prazan

```
int CQempty()
{                                   /*
Function to Check
Circular Queue Empty */
    if(f== -1) return 1;
    return 0;
}
```

- Red je prazan ako mu je front -1
- Ova funkcija nam treba da bismo proverili da li možemo da brišemo iz reda
- Prethodna f-ja nam treba da proverimo da li možemo dodati element u red

Operacije Enqueue i Dequeue

Enqueue

```
void CQinsert(int elem)
{ if( CQfull()) printf("\n\n Overflow!!!!\n\n");
    else {
        if(f==-1)f=0;
        r=(r+1) % SIZE;
        CQ[r]=elem;
    }
}
```

- Kod reda se dodaje element na kraju
- Zamislite red u pošti, novi klijenti dolaze na kraj reda, a oni koji su stigli na red kod šaltera, završavaju posao i brišu se iz reda
- U red može da se doda element ako mesta u redu ima
- Pošto se radi o kružnom redu rear se inkrementira po modulu size
- To znači da možemo da imamo red u kom je prvi elemnt na poziviji 4, a drugi na poziciji 0, treći na poziciji 1, četvrti na poziciji 2 itd.

Operacije Enqueue i Dequeue

- Brisanje iz reda je moguće ako imamo šta da obrišemo
- Ako je to bio jedini element u redu onda f i r vraćamo na -1
- Ako nije front se inkrementira po modulu SIZE

Dequeue

```
int CQdelete()
{ int elem;

if(CQempty())
{ printf("\n\nUnderflow!!!!\n\n");
    return(-1); }
    else {
        elem=CQ[f];
        if(f==r){ f=-1; r=-1;} /* Q has only one element ? */
        else
            f=(f+1) % SIZE;
        return(elem);}}
```

Prikazivanje (štampanje) elemenata

```
void display()
{       /* Function to display status of Circular Queue */
    int i;
    if(CQempty()) printf(" \n Empty Queue\n");
    else
    {
        printf("Front[%d]->",f);
        for(i=f;i!=r;i=(i+1)%SIZE)
            printf("%d ",CQ[i]);
        printf("%d ",CQ[i]);
        printf("<-[%d]Rear",r); }}
```

- Prilikom štampanja, obilazimo niz, od f do r, a inkrementiranje vršimo po modulu SIZE

Main funkcija - 1. deo

```
int opn,elem;
    do
    {
        clrscr();
        printf("\n ### Circular Queue Operations ### \n\n");
        printf("\n Press 1-Insert, 2-Delete,3-Display,4-Exit\n");
        printf("\n Your option ? ");
        scanf("%d",&opn);
```

Main funkcija - 2. deo

```
switch(opn)
        {
        case 1: printf("\n\nRead the element to be Inserted ?");
            scanf("%d",&elem);
            CQinsert(elem); break;
        case 2: elem=CQdelete();
            if( elem != -1)
                printf("\n\nDeleted Element is %d \n",elem);
            break;
```

Main funkcija - 3. deo

```
case 3: printf("\n\nStatus of Circular Queue\n\n");
            display(); break;
        case 4: printf("\n\n Terminating \n\n"); break;
        default: printf("\n\nInvalid Option !!! Try Again !! \n\n");
            break;
        }
        printf("\n\n\n\n  Press a Key to Continue . . . ");
        getch();
    }while(opn != 4);
}
```

Implementacija reda preko ulančane liste

```
#include <stdlib.h>
typedef struct node
{
    int data;
    struct node *link;
}NODE;

void Insert(int);
int  Delete();
void Display();
NODE *front,*rear;
```

- Implementacija reda preko lančane liste slična je kao implementacija jednostruke kružne liste

Operacija Enqueue (dodavanje elementa)

```
void Insert(int info)
{
    NODE *temp;
    temp=(NODE *)malloc(sizeof(NODE));
    if( temp == NULL)
        printf(" Out of Memory !! Overflow !!!");
    else
    {
        temp->data=info;
        temp->link=NULL;
```

```
 if(front == NULL) { front = rear = temp; } /* First Node? */
        else
        { rear->link=temp; rear = temp; }       /* Insert End */
        printf(" Node has been inserted at End Successfully !!");
    }
}
```

Operacija Dequeue (uklanjanje elementa)

```
int Delete()
{
    int info;
    NODE *t;
    if( front ==  NULL) { printf("Underflow!!!"); return -1; }
    else
    {
        t=front;
        info=front->data;
```

```
  if(front == rear) rear=NULL;
        front=front->link;
        t->link=NULL;
        free(t);
        return(info);
    }
}
```

Prikaz (štampanje) elemenata

```
void Display()
{
    NODE *t;
    if( front == NULL) printf("Empty Queue\n");
    else
    {
        t=front;
        printf("Front->");
        while(t)
        {
            printf("[%d]->",t->data);
            t=t->link;
```

Main f-ja - 1. deo

```
main()
{
    /* Main Program */
    int opn,elem;
    front=rear=NULL;
    do
    {
        clrscr();
        printf("\n ### Linked List Implementation of QUEUE Operations ### \n\n");
        printf("\n Press 1-Insert, 2-Delete, 3-Display,4-Exit\n");
        printf("\n Your option ? ");
        scanf("%d",&opn);
```

Main f-ja - 2. deo

```
switch(opn)
        {
        case 1:
            printf("\n\nRead the Element to be Inserted ?");
            scanf("%d",&elem);
            Insert(elem);
            break;
        case 2:
            elem=Delete();
            if(elem != -1)
                printf(" Deleted Node(From Front)with the Data: %d\n",elem);
            break;
```

Main f-ja - 3. deo

```
case 3: printf("Linked List Implementation of Queue: Status:\n");
            Display(); break;
        case 4: printf("\n\n Terminating \n\n"); break;
        default: printf("\n\nInvalid Option !!! Try Again !! \n\n");
            break;
        }
        printf("\n\n\n\n  Press a Key to Continue . . . ");
        getch();
    }while(opn != 4);
}
```

Red sa dva pristupna kraja

- Može i da se dodaje i briše sa početka...
- ...i sa kraja
- Dek

Red sa dva pristupna kraja - 1. deo

```
//Red sa dva pristupna kraja- preko vektora
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define SIZE 100


int queue[SIZE];
int F = -1;
int R = -1;
```

Red sa dva pristupna kraja – 2. deo

```
void insert_r(int x)
{
if(F == (R+1)%SIZE)
printf("\nQueue Overflow");
else if(R == -1)
{
F = 0;
R = 0;
queue[R] = x;
}
else
{
```

Red sa dva pristupna kraja – 3.deo

```
void insert_f(int x)
{
if(F == (R+1)%SIZE)
{
printf("\nQueue Overflow");
}
else if(R == -1)
{
F = 0;
R = 0;
queue[R] = x;
}
```

Red sa dva pristupna kraja – 4.deo

```
int delete_r()
{
int x;
if(F == -1)
{
printf("\nQueue Underflow");
}
else if(F == R)
{
x = queue[F];
F = -1;
R = -1;
}
```

Red sa dva pristupna kraja – 5.deo

```
int delete_f()
{
int x;
if(F == -1)
{
printf("\nQueue Underflow");
}
else if(F == R)
{
x = queue[F];
F = -1;
R = -1;
}
```

Red sa dva pristupna kraja – 6.deo

```
display()
{/* Function to display status of Circular Queue */
int i;
if(F==-1)printf(" \n Empty Queue\n");
else
{
printf("Front[%d]->",F);
for(i=F;i!=R;i=(i+1)%SIZE)
printf("%d ",queue[i]);
printf("%d ",queue[i]);
printf("<-[%d]Rear",R);
}
```

Red sa dva pristupna kraja – 7.deo

```
void main()
{
char choice;
int x;
while(1)
{
system("cls");
printf("1: Insert on Front\n");
printf("2: Insert on Rear\n");
printf("3: Delete From Front\n");
printf("4: Delete From Rear\n");
printf("5: Display list \n");
```

Red sa dva pristupna kraja – 8.deo

```
switch(choice)
{
case '1':
printf("\nEnter Integer Data :");
scanf("%d",&x);
insert_f(x);
break;
case '2':
printf("\nEnter Integer Data :");
scanf("%d",&x);
insert_r(x);
break;
```

Red sa dva pristupna kraja – 9.deo

```
case '4':
printf("\nDeleted Data From Back End: %d",delete_r());
break;
case '5':
printf("\n Red izgleda ovako \n: %d",display());
break;
case '6':
exit(0);
break;
}
system("pause");
}
```

Zanimljive animacije

- https://www.cs.usfca.edu/~galles/visualization/Algorithms.html

Test

1. Opisati kako funkcioniše struktura podataka red?
2. Koja dva pokazivača ima red?
3. Kako se može implementirati red?
4. Koje osnovne metode ima red?
5. Šta je dek?
6. Napisati kod za metodu koja proverava da li je red pun (niz).
7. Napisati kod za metodu koja proverava da li je red prazan (lančana lista.
8. Napisati kod metode za dodavanje elementa u red (niz).
9. Napisati kod metode za brisanje elementa u red (lančana lista).
10. Napisati kod za brisanje s kraja kod deka.

Test

- Test poslati do 20.04.2020. u 14h na mejl apljaskovic@np.ac.rs prema uputstvima sa sajta univerziteta

Hvala na pažnji!