Lineární datové struktury 2

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZADS - Základní algoritmy a datové struktury

Zásobník

Zásobník

```
def push(S, x):
    if S["top"] < length(S["data"]):
        S["data"][S["top"]]=x
        S["top"]=S["top"] + 1

def pop(S):
    if S["top"] > 0:
        S["top"]=S["top"] - 1
        return S["data"][S["top"]]
    return None
```

Zásobník

```
def empty(S):
    return S["top"] == 0

def full(S):
    return S["top"] == length(S["data"])
```

Fronta

Fronta

```
def init_queue(n):
    k=[None for x in range(n)]
    s={"data":k,"head":0,"tail":0}
    return s

def empty(Q):
    return Q["head"] == Q["tail"]

def full(Q):
    return (Q["tail"] + 1)%length(Q["data"]) == Q["head"]
```

Fronta

```
def enqueue(Q,x):
    if not full(Q):
        Q["data"][Q["tail"]]=x
        Q["tail"]=(Q["tail"] + 1)%length(Q["data"])

def dequeue(Q):
    if not empty(Q):
        x=Q["data"][Q["head"]]
        Q["head"]=(Q["head"] + 1)%length(Q["data"])
        return x
    else:
        return None
```

Spojový seznam

Spojový seznam

- Prvky jsou tvořeny slovníky:

```
{"key":10,"next":None}
```

- Funkce:

```
def print_list(l):
    i=l
    while i!=None:
        print(i["key"],end=" ")
        i=i["next"]
    print()
```

Přidávání prvků do seznamu

- Přidání na začátek seznamu:

```
def insert_at_the_start (r, added):
    added["next"] = r
    return added
```

- Přidání uzlu r za uzel added:

```
def insert_after(r, added):
    added["next"] = r["next"]
    r["next"] = added
```

Přidávání prvků do seznamu

- Vyhledání n-tého prvku seznamu:

```
def nth(r, n):
    nn=1
    x=r
    while x != None and nn < n:
        x=x["next"]
        nn=nn + 1
    return x
```

- Přidání za n-tý prvek seznamu:

```
def insert_nth (r, added, n):
    if n == 1:
        return insert_at_the_start(r, added)
    x=nth(r,n-1)
    if x != None:
        insert_after(x, added)
    return r
```

Odebrání prvku

- Odebere uzel za uzlem r:

```
def remove_after(r):
    ret=r["next"]
    if r["next"] != None:
       r["next"] = r["next"]["next"]
    return ret
```

- Odebrání n-tého prvku:

```
def remove_nth(r, n):
    if n == 0:
        return r["next"], r
    x=nth(r, n-1)
    if x != None:
        x=remove_after(x)
    return r, x
```

Odebrání prvku

- Odeber prvek d

```
def remove(r, d):
    if r == d:
        x=r
    while x["next"]!=d:
        x=x["next"]
    remove_after(x)
    return r
```

Hledání prvku

- Hledání prvku s klíčem k:

```
def search(r, k):
    x=r
    while x != None and x["key"] != k:
        x=x["next"]
    return x
```

Spojení dvou seznamů

```
def concatenate(r,s):
    if r==None:
        return s
    if s==None:
        return r
    x=r
    while x["next"]!=None:
        x=x["next"]
    x["next"]=s
    return r
```

Úkol

1. Implementujte frontu pomocí dvou zásobníků.
2. Napište funkce:
   - def enqueue(x)
   - def dequeue()

   ve kterých použijete místo fronty dva zásobníky.

- Příklad:
  - Příkazy:

```
for i in range(10):
    enqueue(i)
for i in range(10):
    print(dequeue(),end=",")
```

Vypíší:

```
0,1,2,3,4,5,6,7,8,9,
```