Breviar Laborator 6: **Cozi**
Cozi

Alexandra Maria Vieru
Facultatea de Automatică și Calculatoare

31 martie 2014

1 Noțiuni generale

Coada este un tip de date abstract care menține o ordine a elementelor și funcționează după principiul FIFO (First In First Out). Operațiile principale pe care o coadă le pune la dispoziție sunt cele de adăugare a unui element la sfârșitul cozii și de ștergere a elementului de la începutul cozii.

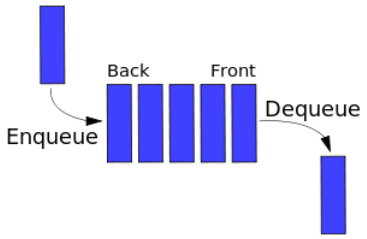

Figura 1: Coada [sursă: wikipedia]

Coada are următoarele operații:

init - face inițializările necesare cozii

enqueue - adaugă un element la sfârșitul cozii

getFront - returnează valoarea elementului de la începutul cozii

dequeue - elimină elementul de la începutul cozii și îi retunrează valoarea

isEmpty - verifică dacă este goală coada, sau există măcar un element în ea

2 Implementări

O coadă poate fi implementată folosind vectori sau liste înlănţuite.

2.1 Vectori

Coada implementată cu ajutorul vectorilor ar trebui să fie redimensionată în mod dinamic, atunci când nu mai este spaţiu în coadă, vectorul trebuie să fie realocat. Pentru a nu fi nevoie să copiem toate elementele la începutul vectorului atunci când se face dequeue unui element, vom considera că vectorul este circular și vom reține pozițiile de început și de sfârșit în variabilele front și rear.
Front va reprezenta prima poziție ocupată, iar rear, prima poziție liberă de după elemente. În imaginea următoare aveți ilustrată evoluția cozii în timpul efectuării câtorva operații enqueue și dequeue.

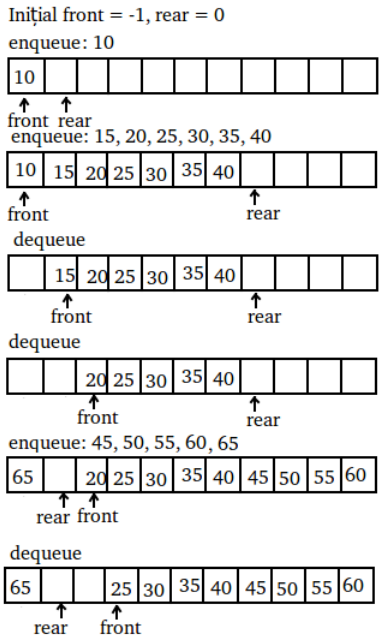

Figura 2: Operații pe coada implementată cu vectori

Algoritmul 1 init(c)

$front \leftarrow -1$ {reprezintă poziția de pe care se elimină}
$rear \leftarrow 0$ {reprezintă poziția pe care se adauga}
$cap \leftarrow c$ {cap e capacitatea inițială a cozii}
$cnt \leftarrow 0$ {numărul de elemente din coadă}
$array \leftarrow aloca(cap)$ {array e vectorul}

Algoritmul 2 enqueue(e)

```
if cnt == cap then
    cap ← 2 * cap
    array ← realoca(cap) {elementele trebuie copiate într-o anumită ordine în
    vectorul nou!}
end if
cnt ← cnt + 1
array[rear] ← e
rear ← (rear + 1)%cap {vectorul e circular}
```

Algoritmul 3 dequeue()

```
if cnt = 0 then
    eroare: coada e goală
    return
end if
cnt ← cnt − 1
e ← array[front]
front ← (front + 1)%cap {vectorul e circular}
return e
```

Algoritmul 4 getFront()

```
if cnt = 0 then
    eroare: coada e goală
    return
end if
return array[front]
```

Algoritmul 5 isEmpty()

```
if cnt = 0 then
    return true
else
    return false
end if
```

2.2 Liste înlănţuite

Pentru ca atât operația de enqueue, cât și cea de dequeue să fie eficiente (să nu fie nevoie să fie parcursă întreaga listă) se folosesc doi pointeri. Unul pointează la începutul listei, iar altul la sfârșitul ei.

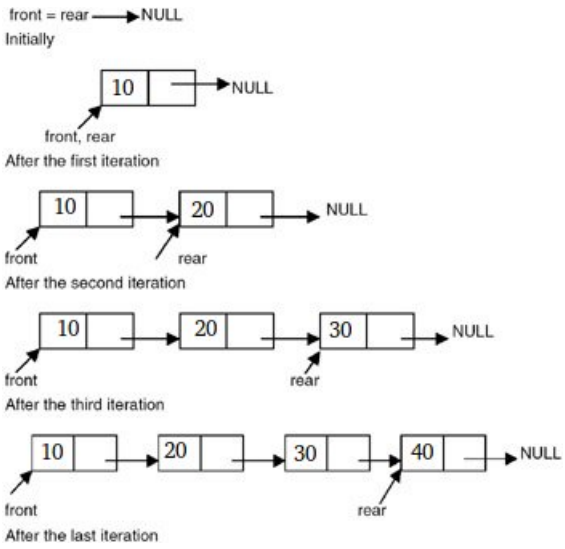

Figura 3: Evoluția cozii implementate cu liste

3 Radix sort

3.1 Algoritm

Radix sort este o metodă de sortare care la fiecare iterație va sorta elementele din vector după cifra de un anumit rang a fiecărui număr: la prima iterație se va sorta după cifra de pe poziția 0, la a doua iterație se va sorta după cifra de pe poziția 1 etc. Una dintre variantele de implementare presupune folosirea a 10 cozi (câte o coadă pentru fiecare cifră din baza 10) în care se vor găsi la fiecare pas numerele care au pe poziția curentă cifra cozii respective.

Exemplu:
Fie vectorul cu următoarele elemente: 86, 120, 105, 22, 48, 760, 15, 220.

Pasul 1:
coada 0 : 120, 760, 220
coada 1 :
coada 2 : 22
coada 3 :
coada 4 :
coada 5 : 105, 15
coada 6 : 86
coada 7 :
coada 8 : 48
coada 9 :
Vectorul a devenit: 120, 760, 220, 22, 105, 15, 86, 48.

Pasul 2:
coada 0 : 105
coada 1 : 15
coada 2 : 120, 220, 22
coada 3 :
coada 4 : 48
coada 5 :
coada 6 : 760
coada 7 :
coada 8 : 86
coada 9 :
Vectorul a devenit: 105, 15, 120, 220, 22, 48, 760, 86.

Pasul 3:
coada 0 : 015, 022, 048, 086
coada 1 : 105, 120
coada 2 : 220
coada 3 :
coada 4 :
coada 5 :
coada 6 :
coada 7 : 760
coada 8 :
coada 9 :
Vectorul a devenit: 15, 22, 48, 86, 105, 120, 220, 760.
Algoritmul se încheie, deoarece au fost comparate toate cifrele, iar vectorul este sortat.

3.2 Pseudocod

Algoritmul 6 radixSort(v)

for $i \leftarrow 0$: (numarul maxim de cifre - 1) dintr-un element din v **do**
  for $j \leftarrow 0 : lungime(v) - 1$ **do**
    pune $v[j]$ în coada corespunzătoare cifrei i
  end for
  scoate elementele din cozi în ordine și le pune în vectorul v
end for

4 Coada cu priorități

Coada cu priorități este un tip de date abstract asemănător unei cozi obișnuite, dar ale cărei elemente au asociată, pe lângă valoare, și o prioritate. Când se scoate un element din coadă, se va scoate elementul cu prioritatea maximă.

Dacă sunt mai multe elemente cu prioritate maximă, se va elimina primul introdus dintre ele.
Cozile și stivele obișnuite pot fi considerate particularizări ale cozilor cu priorități. Prioritatea elementelor din coadă este o valoare care descrește cu fiecare element adăugat în coadă. Prioritatea elementelor din stivă este o valoare care crește cu fiecare element adăugat.