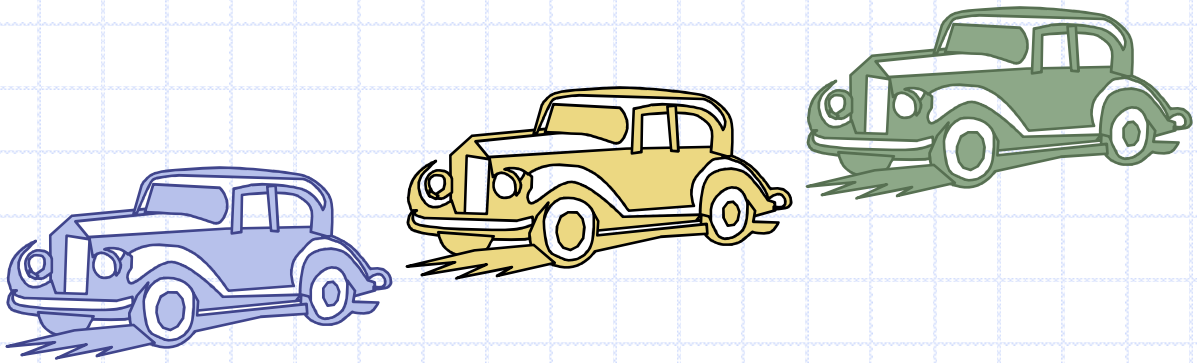
Ουρές



Ο ΑΤΔ Ουρά

- Ο ΑΤΔ Ουρά αποθηκεύει αυθαίρετα αντικείμενα
- Οι εισαγωγές και οι διαγραφές ακολουθούν την πολιτική εξαγωγής του παλαιότερου (FIFO)
- Οι εισαγωγές γίνονται στο τέλος της ουράς και οι εξαγωγές από την αρχή
- Βασικές πράξεις της ουράς :
  - enqueue(object): εισάγει ένα στοιχείο στο τέλος της ουράς
  - object dequeue(): διαγράφει και επιστρέφει το στοιχείο στην αρχή της ουράς
- Βοηθητικές πράξεις της ουράς :
  - object front(): επιστρέφει το στοιχείο στην αρχή της ουράς χωρίς να το διαγράφει
  - integer size(): επιστρέφει το πλήθος των στοιχείων της ουράς
  - boolean isEmpty(): δείχνει αν έχει στοιχεία η ουρά
- Εξαιρέσεις
  - Προσπάθεια εκτέλεσης της dequeue ή της front σε μια κενή ουρά EmptyQueueException

Παράδειγμα

Πράξη	*Έξοδος*	*Q*
enqueue(5)	–	(5)
enqueue(3)	–	(5, 3)
dequeue()	*5*	(3)
enqueue(7)	–	(3, 7)
dequeue()	*3*	(7)
front()	*7*	(7)
dequeue()	*7*	()
dequeue()	*“error”*	()
isEmpty()	*true*	()
enqueue(9)	–	(9)
enqueue(7)	–	(9, 7)
size()	*2*	(9, 7)
enqueue(3)	–	(9, 7, 3)
enqueue(5)	–	(9, 7, 3, 5)
dequeue()	*9*	(7, 3, 5)

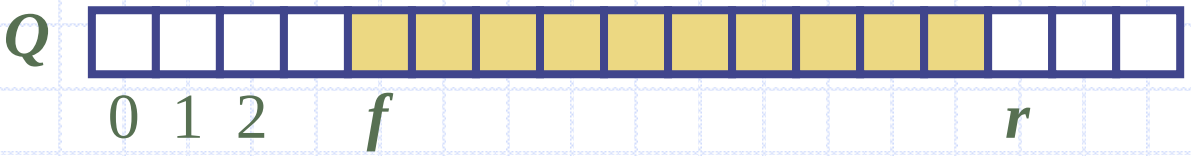
Εφαρμογές των Ουρών

- Άμεσες εφαρμογές
  - Λίστες αναμονής, γραφειοκρατεία
  - Προσπέλαση σε κοινούς πόρους (π.χ., εκτυπωτής)
  - Μικροπρογραμματισμός
- Έμμεσες εφαρμογές
  - Βοηθητική δομή δεδομένων για αλγόριθμους
  - Τμήμα άλλων δομών δεδομένων

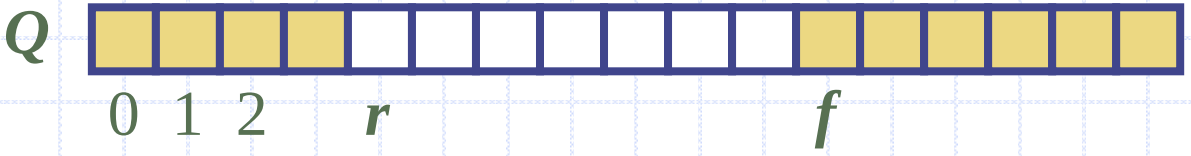
Ουρά που βασίζεται σε πίνακα

- Χρήση ενός πίνακα μεγέθους N κατά κυκλικό τρόπο
- Δύο μεταβλητές καταγράφουν την αρχή και το τέλος της ουράς
  - f δείχνει στο πρώτο στοιχείο
  - r ιδείχνει ένα πάνω από το τέλος
- Η θέση r μένει κενή

Κανονική ρύθμιση



Κυκλική ρύθμιση

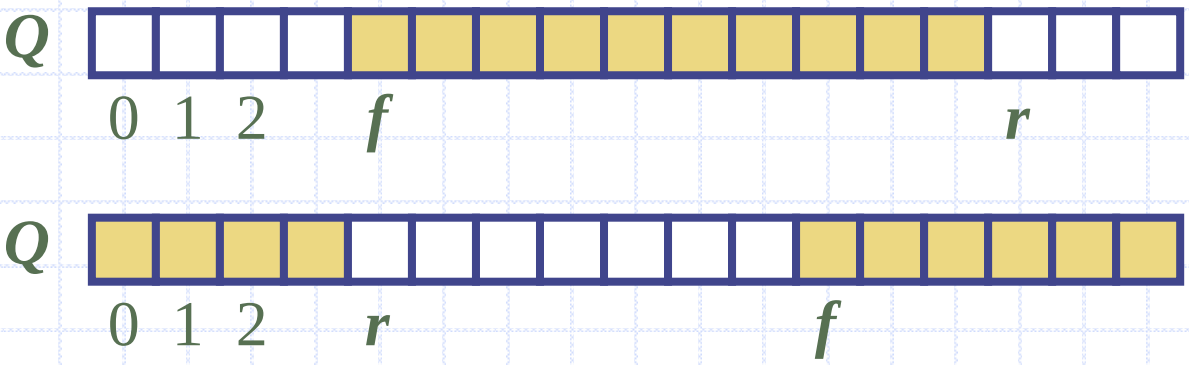


Πράξεις σε Ουρές

- Χρησιμοποιούμε τον τελεστή modulo (υπολοίπου της διαίρεσης)

Algorithm ***size***()
  return $(N - f + r) \bmod N$

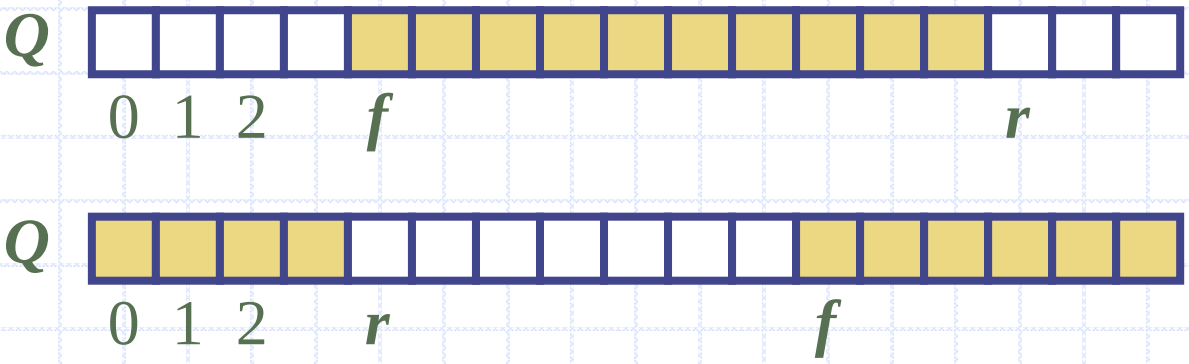
Algorithm ***isEmpty***()
  return $(f = r)$



Πράξεις σε ουρές (συν.)

- Η πράξη enqueue δίνει εξαίρεση αν είναι γεμάτος ο πίνακας
- Η εξαίρεση αυτή εξαρτάται από την υλοποίηση

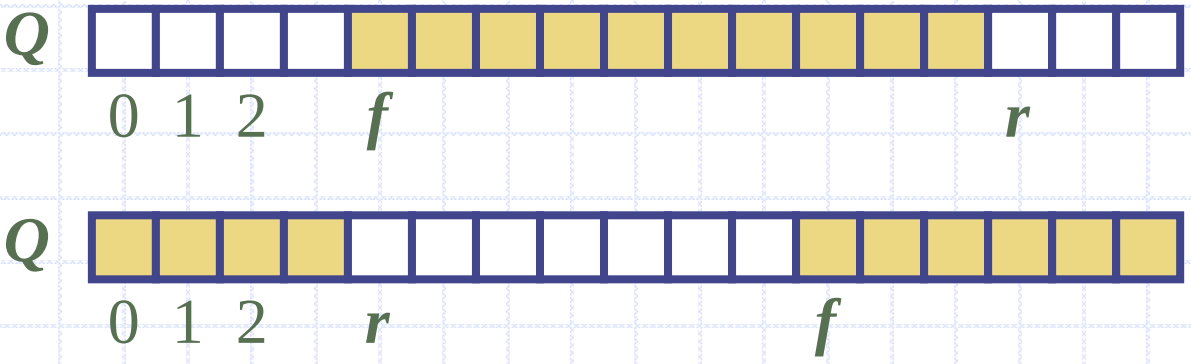
Algorithm *enqueue*(*o*)
  if *size*() = $N - 1$ **then**
    throw *FullQueueException*
  else
    $Q[r] \leftarrow o$
    $r \leftarrow (r + 1) \bmod N$



Πράξεις σε ουρές (συν.)

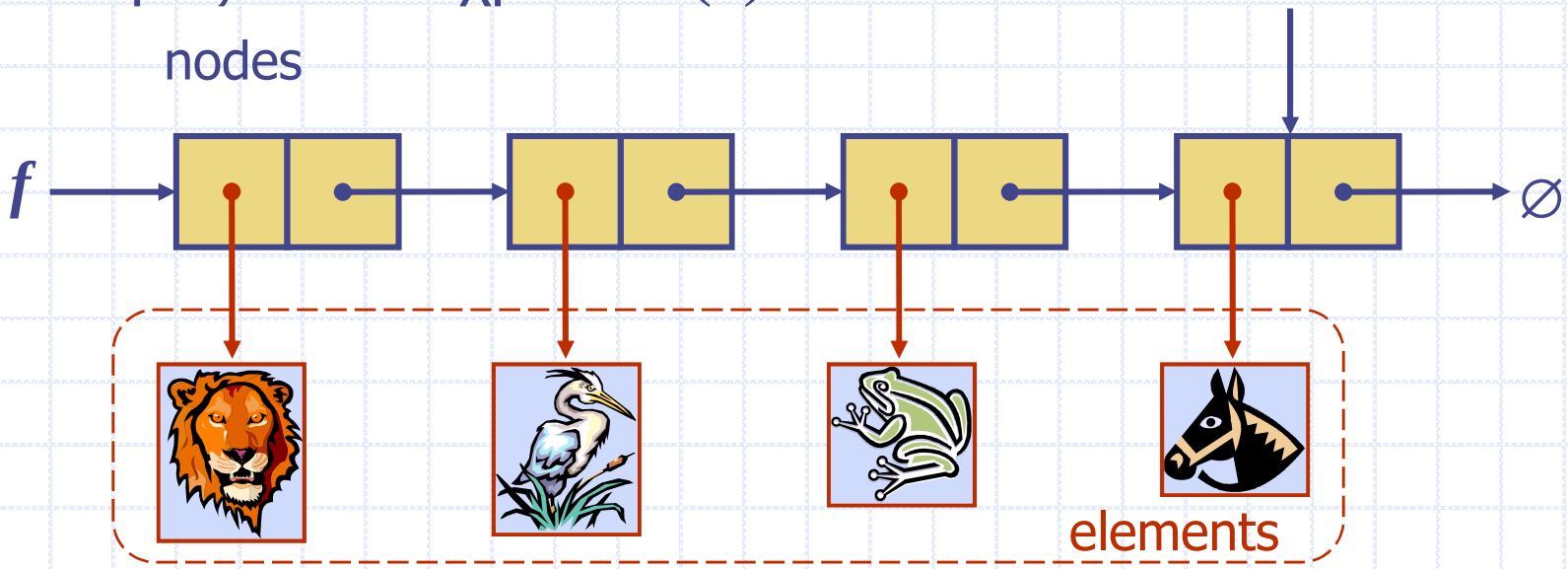
- Η πράξη dequeue δίνει εξαίρεση αν η ουρά είναι κενή
- Η εξαίρεση αυτή ορίζεται στον ΑΤΔ της ουράς

Algorithm ***dequeue***()
  if *isEmpty*() **then**
    throw *EmptyQueueException*
  else
    $o \leftarrow Q[f]$
    $f \leftarrow (f + 1) \bmod N$
    return o



Η ουρά σαν συνδεδεμένη λίστα

- Μπορούμε να υλοποιήσουμε μια ουρά με μια απλά συνδεδεμένη λίστα
  - Το στοιχείο της αρχής αποθηκεύεται στον πρώτο κόμβο
  - Το στοιχείο του τέλους πάει στον τελευταίο κόμβο
- Ο απαιτούμενος χώρος είναι $O(n)$ και κάθε πράξη της ουράς απαιτεί χρόνο $O(1)$



Διεπαφή ουράς στην Java

- Η Java διεπαφή που αντιστοιχεί στον ΑΤΔ ουρά
- Απαιτεί τον ορισμό της κλάσης EmptyQueueException
- Δεν υπάρχει αντίστοιχη ενσωματωμένη Java κλάση

```
public interface Queue<E> {
   public int size();
   public boolean isEmpty();
   public E front()
         throws EmptyQueueException;
   public void enqueue(E element);
   public E dequeue()
          throws EmptyQueueException;
}
```

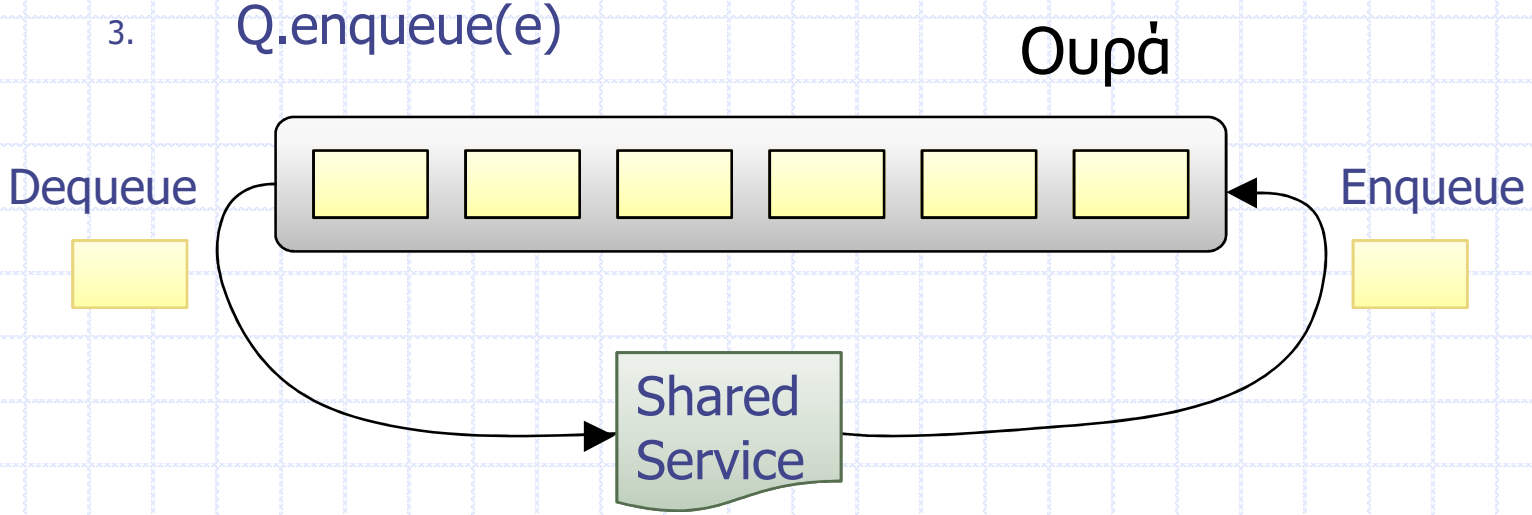
Ουρές και Java

- Η java υποστηρίζει μια διεπαφή ουράς java.uutil.Queue, που λειτουργικά μοιάζει με τον αφηρημένο τύπο της ουράς που περιγράφουμε. Η διαφορά είναι ότι δεν επιμένει στην FIFO αρχή.

ΑΦΤ ουράς	Java.util.Queue
size()	size()
isEmpty()	isEmpty()
enqueue(e)	add(e) ή offer(e)
dequeue()	remove() ή poll()
front()	peek() ή element()

Εφαρμογή: Κυκλικοί (Round Robin) Δρομολογητές

- Μπορούμε να υλοποιήσουμε ένα κυκλικό δρομολογητή με χρήση μιας ουράς Q εκτελώντας με επανάληψη τα παρακάτω βήματα:
  1. e = Q.dequeue()
  2. Service element e
  3. Q.enqueue(e)



Ουρές με δύο άκρα

- Αποτελούν επέκταση των ουρών και αναφέρονται στη βιβλιογραφία σαν διπλές ουρές επειδή οι πράξεις εκτελούται και στα δύο άκρα.

Πράξεις στον ΑΤΔ διπλής ουράς

- addFirst εισάγει ένα νέο στοιχείο στην κεφαλή της διπλής ουράς
- addLast εισάγει ένα νέο στοιχείο στο τέλος της διπλής ουράς
- removeFirst διαγράφει και επιστρέφει το πρώτο στοιχείο της διπλής ουράς
- removeLast διαγράφει και επιστρέφει το τελευταίο στοιχείο της διπλής ουράς

Άσκηση

- Έστω 3 διακριτοί ακέραιοι τους οποίους τοποθετώ τυχαία σε μια στοίβα S. Γράψτε ψευδοκώδικα (χωρίς επαναλήψεις και αναδρομή) που χρησιμοποιεί μόνο μια σύγκριση και μόνο μια μεταβλητή x, που εξασφαλίζει με πιθανότητα 2/3 ότι στο τέλος αυτού του κώδικα το x περιέχει τον μεγαλύτερο από τους τρεις αριθμούς

Διευθέτηση Βαγονιών (με στοίβα)

[581742963]

[987654321]