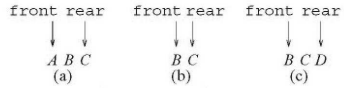
Ουρές

Δημήτρης Φωτάκης

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Πανεπιστήμιο Αιγαίου

Ουρές (queues)



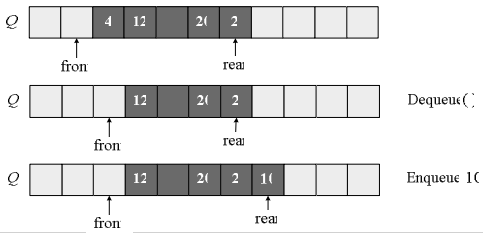
- Ειδική μορφή γραμμικής λίστας.
  Εισαγωγή (enqueue) στο **τέλος.**
  Διαγραφή (dequeue) από **αρχή.**
- Δομή **FIFO** (First-In-First-Out).
- Λειτουργίες:
  - Enqueue(x): εισαγωγή x στο τέλος.
  - Dequeue(): διαγραφή και επιστροφή πρώτου στοιχείου.
  - First(), Last(): επιστροφή πρώτου (τελευταίου) χωρίς διαγραφή.
  - isEmpty(), isFull(), size(): βοηθητικές λειτουργίες.

Εφαρμογές

- Άμεσες εφαρμογές:
  - Υλοποίηση FCFS ουρών αναμονής.
    - Προσπέλαση κοινόχρηστων πόρων.
    - Δρομολόγηση επεξεργαστών (πολύ-χρηστικά πολύ-επεξεργαστικά περιβάλλοντα)
    - Routing buffers.
- Έμμεσες εφαρμογές:
  - Βοηθητική Δομή Δεδομένων.
  - Συστατικό άλλων ΔΔ και αλγορίθμων.

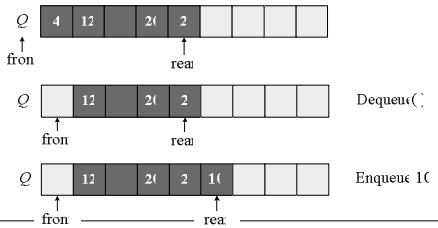
Υλοποίηση με Πίνακα

- Πίνακας `Q[nMax]` με μεταβλητές **`front`** για πρώτο και **`rear`** για τελευταίο στοιχείο.
- Εισαγωγή στο **τέλος.** Διαγραφή από **αρχή.**



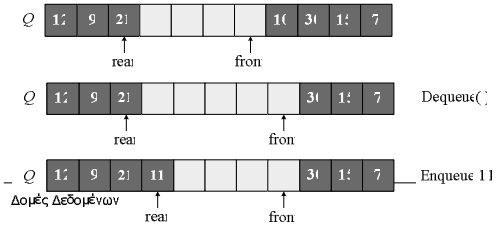
Υλοποίηση με Πίνακα

- Γιατί δεν ξεκινάει πάντα η ουρά από αρχή;
  - Μετακίνηση **όλων** των στοιχείων αριστερά σε διαγραφή.
- Σταδιακή μετακίνηση ουράς προς τα δεξιά!
  - Θα εξαντληθούν ελεύθερες θέσεις για εισαγωγή.



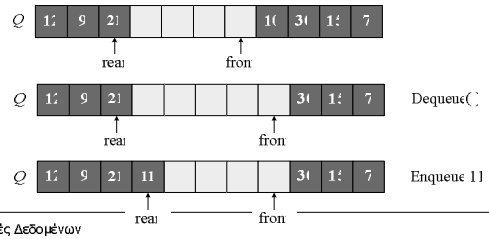
«Κυκλικός» Πίνακας

- **front** : μία θέση πριν πρώτο στοιχείο.
  rear : τελευταίο στοιχείο.
- **Διαγραφή:**
```
front = (front + 1) % nMax;
return(Q[front]);
```
- **Εισαγωγή:**
```
rear = (rear + 1) % nMax;
Q[rear] = x;
```

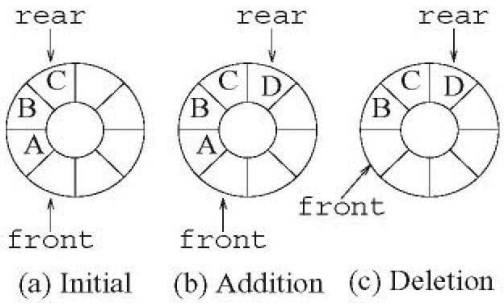


«Κυκλικός» Πίνακας

- Πως διακρίνουμε **άδεια** από **πλήρη** ουρά;
  - Δεν αφήνουμε ουρά να γεμίσει! **Πλήρης: nMax – 1 στοιχεία.**
  - `isFull() { return(((rear+1) % nMax == front) ? 1 : 0 ); }`
  - `isEmpty() { return(front == rear); }`



«Κυκλικός» Πίνακας



Απόδοση – Περιορισμοί

- Λειτουργίες σε χρόνο O(1) (βέλτιστο).
- Χώρος Θ(*nMax*) (πιθανή σπατάλη).
- Εύκολη και γρήγορη υλοποίηση.
- Μέγιστος αριθμός στοιχείων γνωστός εκ των προτέρων!
  - Υποεκτίμηση: γεμάτη ουρά.
  - Υπερεκτίμηση: σπατάλη χώρου.
- Δυναμική διαχείριση μεγέθους:
  - Διπλασιασμός μεγέθους όταν γεμίζει .
  - Υποδιπλασιασμός όταν utilization < 0.25 .
  - Δυσκολεύει η υλοποίηση.

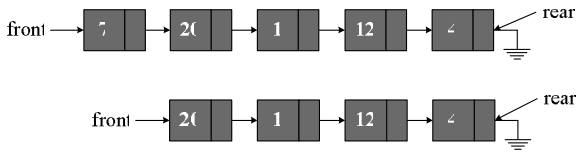
Υλοποίηση με Λίστα

- Πρώτο στοιχείο σε **`front`**, τελευταίο σε **`rear`**
- **Διαγραφή** (πρώτο στοιχείο) από **αρχή.**
- **Εισαγωγή** (τελευταίο στοιχείο) στο **τέλος.**

Υλοποίηση με Λίστα

- **Διαγραφή:**

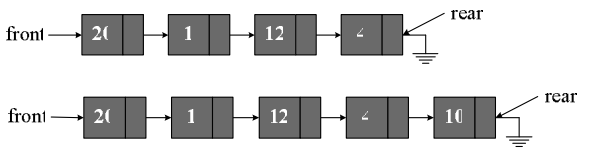
```
Dequeue( ) {
   q = front; x = q->elem;
   front = front->next;
   free(q); return(x); }
```



Υλοποίηση με Λίστα

- **Εισαγωγή:**

```
Enqueue(Node *p) {
   if (front) rear->next = p;
   else front = p;
   rear = p;
   p->next = NULL;  }
```



Απόδοση – Περιορισμοί

- Λειτουργίες σε χρόνο O(1) (βέλτιστο).
- Δυναμική διαχείριση χώρου: Θ(*n*) (βέλτιστο).
  Επιπλέον χώρος μόνο για pointers.
- Λιγότερο εύκολη και γρήγορη υλοποίηση.
- Συμπέρασμα:
  - Πίνακας όταν γνωρίζουμε μέγιστο αριθμό στοιχείων.
  - Λίστα διαφορετικά.