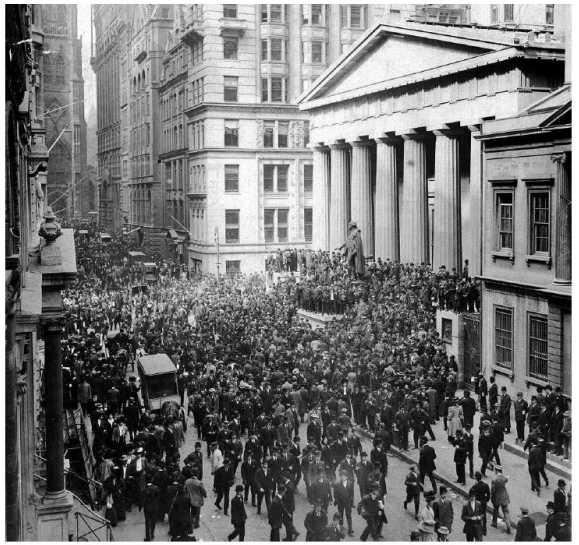
Παρουσίαση για χρήση με το σύγγραμμα, Αλγόριθμοι Σχεδίαση και Εφαρμογές, των M. T. Goodrich and R. Tamassia, Wiley, 2015 (στα ελληνικά από εκδόσεις Μ. Γκιούρδας)

Ουρές Προτεραιότητας



Wall Street during the Bankers Panic of 1907. From the New York Public Library's Digital Gallery, in the Irma and Paul Milstein Division of United States History, Local History and Genealogy. 1907. Public domain image.

Εφαρμογή: Μηχανές αντιστοίχισης μετοχών

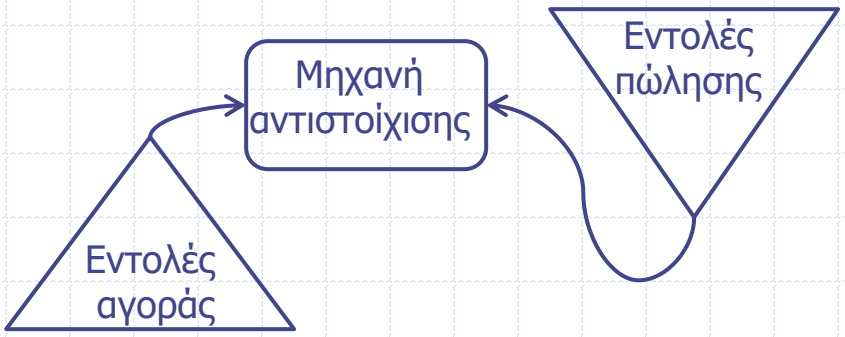
- Στην βάση των σύγχρονων συστημάτων αγοραπωλησίας μετοχών υπάρχουν συστήματα γνωστά ως **μηχανές αντιστοίχισης**, που αντιστοιχούν τις μετοχές των πωλητών με των αγοραστών.
- Μία απλοποίηση του πως λειτουργεί ένα τέτοιο σύστημα είναι το βιβλίο συνεχόμενου ορίου εντολών, όπου οι αγοραστές ζητούν να αγοράσουν έναν αριθμό μετοχών σε κάποια (ή κάτω) τιμή και οι πωλητές προσφέρουν προς πώληση έναν αριθμό μετοχών σε κάποια (ή πάνω) τιμή.

STOCK: EXAMPLE.COM

Buy Orders			Sell Orders		
Shares	**Price**	**Time**	**Shares**	**Price**	**Time**
1000	4.05	20 s	500	4.06	13 s
100	4.05	6 s	2000	4.07	46 s
2100	4.03	20 s	400	4.07	22 s
1000	4.02	3 s	3000	4.10	54 s
2500	4.01	81 s	500	4.12	2 s
			3000	4.20	58 s
			800	4.25	33 s
			100	4.50	92 s

Εφαρμογή: Μηχανές αντιστοίχισης μετοχών

- Οι εντολές αγοράς και πώλησης ταξινομούνται βάση ιεραρχίας πρώτα της τιμής και μετά του χρόνου:
  - "Η τιμή" έχει την υψηλότερη προτεραιότητα και εάν δύο εντολές έχουν την ίδια τιμή τότε αυτή που είναι περισσότερο καιρό στο βιβλίο έχει προτεραιότητα.
- Όταν μία νέα εντολή εισέρχεται στο σύστημα η μηχανή αντιστοίχισης αποφασίζει εάν η εντολή μπορεί να εκτελεστεί άμεσα.
- Αυτό αντιστοιχεί σε δύο στιγμιότυπα της δομής δεδομένων που θα συζητηθεί —την **ουρά προτεραιότητας**— μία για εντολές πώλησης και μία για εντολές αγοράς.
- Αυτή η δομή δεδομένων πραγματοποιεί αφαιρέσεις βάση της προτεραιότητας με την οποία εισήχθησαν.



STOCK: EXAMPLE.COM

Buy Orders			Sell Orders		
Shares	Price	Time	Shares	Price	Time
1000	4.05	20 s	500	4.06	13 s
100	4.05	6 s	2000	4.07	46 s
2100	4.03	20 s	400	4.07	22 s
1000	4.02	3 s	3000	4.10	54 s
2500	4.01	81 s	500	4.12	2 s
			3000	4.20	58 s
			800	4.25	33 s
			100	4.50	92 s

Σχέσεις ολικής διάταξης

- Τα κλειδιά σε μια ουρά προτεραιότητας μπορούν να είναι οποιοδήποτε αντικείμενο που ταξινομείται.
- Κάθε ζεύγος τέτοιων κλειδιών θα πρέπει να είναι συγκρίσιμα βάσει **ολικής διάταξης**.

- Μαθηματική έννοια της **ολικής διάταξης**
  - Ιδιότητα **Σύγκρισης**: $x \leq y$ ή $y \leq x$
  - **Ανακλαστική** ιδιότητα: $x \leq x$
  - **Αντισυμμετρική** ιδιότητα: $x \leq y$ και $y \leq x \Rightarrow x = y$
  - **Μεταβατική** ιδιότητα: $x \leq y$ and $y \leq z \Rightarrow x \leq z$

Λειτουργίες ουράς προτεραιότητας

- Μία ουρά προτεραιότητας αποθηκεύει μία συλλογή εγγραφών
- Κάθε εγγραφή είναι ένα ζεύγος (κλειδί, τιμή)
- Κυρίες μέθοδοι:
  - insert(k, v)
    εισάγει μία εγγραφή με κλειδί k και τιμή v
  - removeMin()
    αφαιρεί και επιστρέφει την εγγραφή με το μικρότερο κλειδί ή null εάν η ουρά προτεραιότητας είναι κενή
- Επιπλέον μέθοδοι:
  - min()
    επιστρέφει χωρίς να αφαιρεί την εγγραφή με το μικρότερο κλειδί ή null εάν η ουρά προτεραιότητας είναι κενή
  - size()
  - isEmpty()
- Εφαρμογή:
  - Πελάτες πτήσεων εν αναμονή
  - Δημοπρασίες
  - Χρηματιστήρια

Παράδειγμα

- Μία ακολουθία από κλήσεις μεθόδων μιας ουράς προτεραιότητας:

Method	Return Value	Priority Queue Contents
insert(5,A)		{ (5,A) }
insert(9,C)		{ (5,A), (9,C) }
insert(3,B)		{ (3,B), (5,A), (9,C) }
min( )	(3,B)	{ (3,B), (5,A), (9,C) }
removeMin( )	(3,B)	{ (5,A), (9,C) }
insert(7,D)		{ (5,A), (7,D), (9,C) }
removeMin( )	(5,A)	{ (7,D), (9,C) }
removeMin( )	(7,D)	{ (9,C) }
removeMin( )	(9,C)	{ }
removeMin( )	null	{ }
isEmpty( )	true	{ }

Ουρές Προτεραιότητας βάσει λίστας

- Υλοποίηση με αταξινόμητη λίστα

- Απόδοση:
  - insert $O(1)$ αφού μπορούμε να προσθέσουμε κάποιο αντικείμενο στην αρχή ή στο τέλος της λίστας
  - removeMin και min $O(n)$ αφού πρέπει να διασχίσουμε όλη την ακολουθία για να βρούμε το μικρότερο κλειδί

- Υλοποίηση με ταξινομημένη λίστα

- Απόδοση :
  - insert $O(n)$ καθώς πρέπει να βρούμε που θα προσθέσουμε το κλειδί
  - removeMin και min $O(1)$ αφού το μικρότερο κλειδί είναι στην αρχή

Ταξινόμηση με ουράς προτεραιότητας

- Μπορούμε να χρησιμοποιήσουμε μία ουρά προτεραιότητας για να ταξινομήσουμε μία λίστα συγκρίσιμων στοιχείων
  1. Είσοδος των στοιχείων ένα προς ένα με μία σειρά λειτουργιών insert
  2. Αφαίρεση των στοιχείων με μία σειρά λειτουργιών removeMin
- Ο χρόνος εκτέλεσης εξαρτάται από την υλοποίηση της ουράς προτεραιότητας.

Algorithm PQ-Sort(C, P):
Input: An n-element array, C, index from 1 to n, and a priority queue P that compares keys, which are elements of C, using a total order relation
Output: The array C sorted by the total order relation

for $i \leftarrow 1$ **to** n **do**
    $e \leftarrow C[i]$
    P.insert(e, e) // the key is the element itself
for $i \leftarrow 1$ **to** n **do**
    $e \leftarrow P$.removeMin$()$ // remove a smallest element from P
    $C[i] \leftarrow e$

Ταξινόμηση με επιλογή

- Η ταξινόμηση με επιλογή είναι μία παραλλαγή της ταξινόμησης με ουρά προτεραιότητας όπου η ουρά προτεραιότητας υλοποιείται με αταξινόμητη ακολουθία
- Χρόνος εκτέλεσης της ταξινόμησης με επιλογή:
  1. Η είσοδος των στοιχείων στην ουρά προτεραιότητας με n λειτουργίες insert απαιτεί χρόνο $O(n)$
  2. Η αφαίρεση των στοιχείων με ταξινομημένη σειρά από την ουρά προτεραιότητας με n removeMin απαιτεί χρόνο ανάλογο του:

$$1 + 2 + \ldots + n$$

- Η ταξινόμηση με επιλογή είναι $O(n^2)$

Παράδειγμα ταξινόμησης με επιλογή

	Sequence S	Priority Queue P
Input:	(7,4,8,2,5,3,9)	()
Phase 1		
(a)	(4,8,2,5,3,9)	(7)
(b)	(8,2,5,3,9)	(7,4)
..	.. ..	
(g)	()	(7,4,8,2,5,3,9)
Phase 2		
(a)	(2)	(7,4,8,5,3,9)
(b)	(2,3)	(7,4,8,5,9)
(c)	(2,3,4)	(7,8,5,9)
(d)	(2,3,4,5)	(7,8,9)
(e)	(2,3,4,5,7)	(8,9)
(f)	(2,3,4,5,7,8)	(9)
(g)	(2,3,4,5,7,8,9)	()

Ταξινόμηση με επιλογή επί τόπου

- Ένας αλγόριθμος ταξινόμησης λέγεται **«επιτόπου»** όταν χρησιμοποιεί μόνο λίγη μνήμη επιπλέον του πίνακα εισαγωγής.

Algorithm SelectionSort(A):
Input: An array A of n comparable elements, indexed from 1 to n
Output: An ordering of A so that its elements are in nondecreasing order.

```
for i ← 1 to n − 1 do
    // Find the index, s, of the smallest element in A[i..n].
    s ← i
    for j ← i + 1 to n do
        if A[j] < A[s] then
            s ← j
    if i ≠ s then
        // Swap A[i] and A[s]
        t ← A[s]; A[s] ← A[i]; A[i] ← t
return A
```

Ταξινόμηση με εισαγωγή

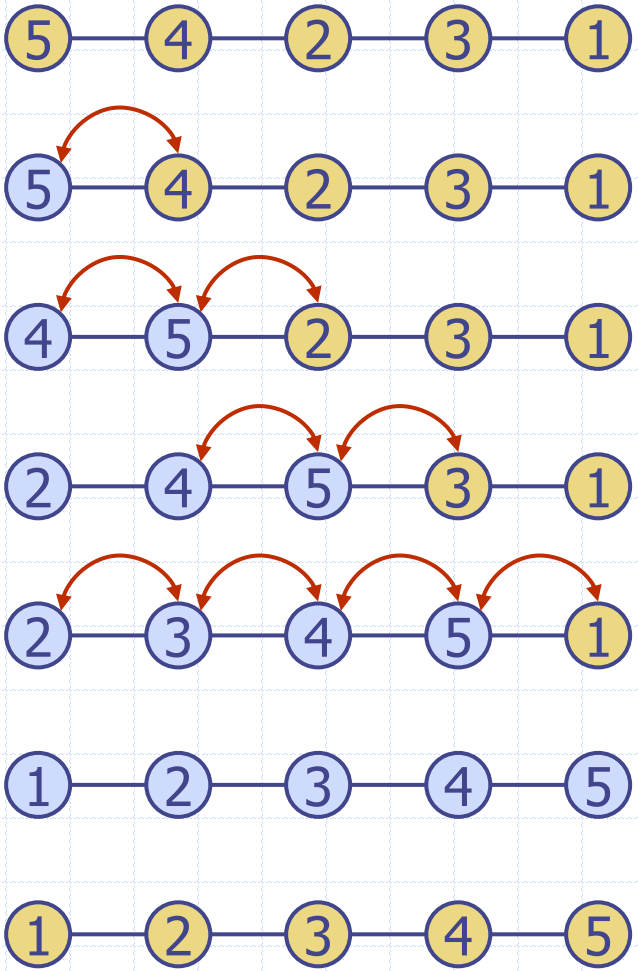
- Η ταξινόμηση με εισαγωγή είναι μία παραλλαγή της ταξινόμησης με ουρά προτεραιότητας όπου η ουρά προτεραιότητας υλοποιείται με μία ταξινομημένη ακολουθία
- Χρόνος εκτέλεσης της ταξινόμησης με εισαγωγή:
  1. Η είσοδος των στοιχείων στην ουρά προτεραιότητας με n λειτουργίες insert απαιτεί χρόνο ανάλογο του:
  $$1 + 2 + \ldots + n$$
  2. Η αφαίρεση ταξινομημένων στοιχείων από την ουρά προτεραιότητας με n λειτουργίες removeMin απαιτεί χρόνο $O(n)$
- Η ταξινόμηση με εισαγωγή είναι $O(n^2)$ στη χειρότερη περίπτωση.

Παράδειγμα ταξινόμησης με εισαγωγή

	Sequence S	Priority queue P
Input:	(7,4,8,2,5,3,9)	()
Phase 1		
(a)	(4,8,2,5,3,9)	(7)
(b)	(8,2,5,3,9)	(4,7)
(c)	(2,5,3,9)	(4,7,8)
(d)	(5,3,9)	(2,4,7,8)
(e)	(3,9)	(2,4,5,7,8)
(f)	(9)	(2,3,4,5,7,8)
(g)	()	(2,3,4,5,7,8,9)
Phase 2		
(a)	(2)	(3,4,5,7,8,9)
(b)	(2,3)	(4,5,7,8,9)
..	..	..
(g)	(2,3,4,5,7,8,9)	()

Ταξινόμηση με εισαγωγή επί τόπου

- Αντί να χρησιμοποιήσουμε μία εξωτερική δομή δεδομένων μπορούμε να υλοποιήσουμε την ταξινόμηση με επιλογή και την ταξινόμηση με εισαγωγή επί τόπου
- Ένα τμήμα της ακολουθίας εισόδου χρησιμοποιείται ως ουρά προτεραιότητας
- Για ταξινόμηση με εισαγωγή επί τόπου
  - Διατηρούμε ταξινομημένο το αρχικό μέρος της ακολουθίας
  - Χρησιμοποιούμε αντιμεταθέσεις αντί να τροποποιούμε την ακολουθία



Ταξινόμηση με εισαγωγή επί τόπου

- Ψευδοκώδικας:

Algorithm InsertionSort(A):
Input: An array, A, of n comparable elements, indexed from 1 to n
Output: An ordering of A so that its elements are in nondecreasing order.

for $i \leftarrow 2$ **to** n **do**
  $x \leftarrow A[i]$
  // Put x in the right place in $A[1..i]$, moving larger elements up as needed.
  $j \leftarrow i$
  while $j > 1$ **and** $x < A[j-1]$ **do**
    $A[j] \leftarrow A[j-1]$ // move $A[j-1]$ up one cell
    $j \leftarrow j - 1$
  $A[j] \leftarrow x$
return A