Ταξινόμηση σε γραμμικό χρόνο

Ταξινόμηση μέσω απαρίθμησης (Counting-Sort)

Είσοδος: n ακέραιοι από το σύνολο $[1..k]$.

Έξοδος: Οι n ακέραιοι ταξινομημένοι σε αύξουσα σειρά.

Ιδέα: Αποφάσισε για κάθε στοιχείο x της εισόδου, τον αριθμό των στοιχείων τα οποία είναι $\leq x$.

- Ο αλγόριθμος Counting-Sort χρησιμοποιεί 3 πίνακες:

  $A[1..n]$: Περιέχει τα στοιχεία της εισόδου.

  $B[1..n]$: Περιέχει την ταξινομημένη έξοδο.

  $C[1..k]$: Το $C[i]$ περιέχει τον αριθμό των στοιχείων που είναι $\leq i$.


```
COUNTING-SORT(A, B, k)
for i ← 1 to k
    do C[i] ← 0
for j ← 1 to length[A]
    do C[A[j]] ← C[A[j]] + 1
/* C[i] now contains the number of */
/* elements equal to i (see (a) ). */
for i ← 2 to k
    do C[i] ← C[i] + C[i − 1]
/* C[i] now contains the number of */
/* elements ≤ i (see (b) ). */
for j ← length[A] downto 1
    do B[C[A[j]]] ← A[j]
       C[A[j]] ← C[A[j]] − 1
```

(a)

	1	2	3	4	5	6	7	8
A	3	6	4	1	3	4	1	4

	1	2	3	4	5	6
C	2	0	2	3	0	1

(b)

	1	2	3	4	5	6
C	2	2	4	7	7	8

(c)

	1	2	3	4	5	6	7	8
B							4	

	1	2	3	4	5	6
C	2	2	4	6	7	8

(d)

	1	2	3	4	5	6	7	8
B		1					4	

	1	2	3	4	5	6
C	1	2	4	6	7	8

(e)

	1	2	3	4	5	6	7	8
B		1				4	4	

	1	2	3	4	5	6
C	1	2	4	5	7	8

(f)

	1	2	3	4	5	6	7	8
B	1	1	3	3	4	4	4	6

Πολυπλοκότητα χρόνου: $O(k+n)$ Εάν $k = O(n) \Longrightarrow O(n)$ αλγόριθμος.

Παρατήρηση Το αποτέλεσμα αυτό δεν αντικρούει το $\Omega(n \log n)$ κάτω φράγμα. Δεν χρησιμοποιούμε το μοντέλο συγκρίσεων. Στην πραγματικότητα, ο αλγόριθμος COUNTING-SORT() δεν έκανε καμία σύγκριση μεταξύ στοιχείων της εισόδου.

Ορισμός Ένας αλγόριθμος ταξινόμησης είναι *ευσταθής (stable)* εάν τα στοιχεία με την ίδια τιμή εμφανίζονται στην έξοδο με την ίδια σειρά που εμφανίστηκαν στην είσοδο.

Πρόταση Ο αλγόριθμος COUNTING-SORT() είναι ευσταθής.
(Η απόδειξη γίνεται με εξέταση του αλγορίθμου.)

Ταξινόμηση βάσης (αριθμητικού συστήματος) - Radix-Sort

Είσοδος: n στοιχεία τέτοια ώστε κάθε στοιχείο αποτελείται από d ψηφία όπου το πρώτο (από τα αριστερά) ψηφίο είναι το μικρότερης τάξης ψηφίο και το d-οστό ψηφίο είναι το μεγαλύτερης τάξης ψηφίο.

Έξοδος: Ένας ταξινομημένος πίνακας που περιέχει τα n στοιχεία σε αύξουσα σειρά.

Ιδέα: Ταξινόμησε τα στοιχεία πρώτα με βάση το *ελάχιστα σημαντικό ψηφίο.*

Παράδειγμα:

329		72**0**		7**2**0		**3**29
457		35**5**		3**2**9		**3**55
657		43**6**		4**3**6		**4**36
839	$\Longrightarrow$	45**7**	$\Longrightarrow$	8**3**9	$\Longrightarrow$	**4**57
436		65**7**		3**5**5		**6**57
720		32**9**		4**5**7		**7**20
355		83**9**		6**5**7		**8**39

$RADIX - SORT(A, d)$
for $i = 1$ **to** d
do use a stable sorting algorithm to sort array A on digit i.

Απόδειξη της ορθότητας (με επαγωγή ως προς τον αριθμό των θέσεων.)

- Υποθέτουμε ότι τα ψηφία χαμηλότερης τάξης είναι ταξινομημένα.
- Αποδεικνύουμε ότι η ταξινόμηση ως πρός το επόμενο ψηφίο αφήνει τον πίνακα ορθά ταξινομημένο.

- Έστω δύο τυχόντα στοιχεία:
  - Εάν τα δύο ψηφία σε αυτήν την θέση είναι διαφορετικά, τότε ταξινομώντας τα σε αυτή την θέση θα βρεθούν στην σωστή διάταξη. Τα μικρότερης τάξης ψηφία δεν επηρεάζουν το αποτέλεσμα.

  - Εάν τα δύο ψηφία σε αυτήν την θέση είναι ίσα, τότε τα στοιχεία αυτά βρίσκονται ήδη στην σωστή θέση. *Αφού χρησιμοποιούμε έναν ευσταθή αλγόριθμο ταξινόμησης, τα στοιχεία θα παραμείνουν στην σωστή σειρά.*

□

Ανάλυση

- Κάθε θέση (pass) απαιτεί $\Theta(n + k)$ χρόνο, όπου κάθε ψηφίο μπορεί να πάρει τιμές από το σύνολο $[1..k]$.
  $\Longrightarrow \Theta(d(n + k))$
- Εάν $k = O(n) \Longrightarrow \Theta(dn)$, χρησιμοποιώντας των αλγόριθμο COUNTING-SORT().

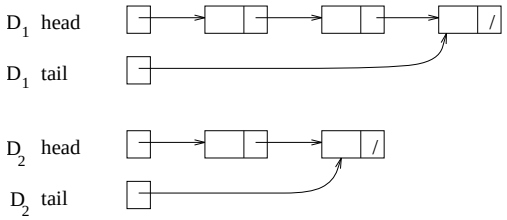
Sort-σε δοχεία(όμοιος με τον Radix-Sort)

Είσοδος: Μία συνδεδεμένη λίστα των n d-ψήφιων στοιχείων όπου κάθε ψηφίο ανήκει στο διάστημα $[1..k]$.

Έξοδος: Μία ταξινομημένη συνδεδεμένη λίστα των n στοιχείων σε αύξουσα σειρά.

Ιδέα:
- Να διαχωρίσουμε τα στοιχεία σε δοχεία *(bins)*.
- Ομοίως με τον αλγόριθμο Radix-Sort.

Απεικόνιση δομής δεδομένων ($k = 2$)



Παράδειγμα Ταξινόμηση της λίστας $< 36,\ 9,\ 0,\ 25,\ 1,\ 49,\ 64,\ 16,\ 81,\ 4 >$.

Bin	Contents
0	0
1	1, 81
2	
3	
4	64, 4
5	25
6	36, 16
7	
8	
9	9, 49
	d = 1

Bin	Contents
0	0, 1, 4, 9
1	16
2	25
3	36
4	49
5	
6	64
7	
8	81
9	
	d = 2

$BIN - SORT(L, d, k)$ /* L είναι η λίστα των στοιχείων στην είσοδο. */
/* Κάθε στοιχείο αποτελείται από d ψηφία. */
/* Κάθε ψηφίο είναι στο σύνολο $[1..k]$. */

for $i = 1$ **to** d **do**

- Process list L sequentially and place each element of L at the top of the list $L_j,\ 1 \leq j \leq k$, where j is the i^{th} digit of the element under consideration.
- Concatenate $L_1,\ L_2, \ldots,\ L_k$ into L.

Πολυπλοκότητα $\Theta(dn + dk)$

- Εάν $k = O(n)$ και d είναι σταθερά $\implies \Theta(n)$.