Algorithms - Assignment 1

Καζγκούτης Αθανάσιος Charbel Al Haddad
Παπαδόπουλος Δημήτριος-Λάζαρος

March 24, 2023

Πρόβλημα 1

Ερώτημα 1 Ο αλγόριθμος που παρατιύθεται παρακάτω δέχεται σαν είσοδο μια συστοιχία n στοιχίων όπου n ειναι οι ψήφοι -σε ονοματεπώνυμα-μιας κοινότητας. Στόχος του αλγόριθμου ειναι:

1. να ελέγξει ποιος/α έχει τους περισσότερους ψήφους.
2. αμα ξεπερνούν ή είναι ίσοι του 50% των συνολικών ψήφων.
3. εφόσων ισχύει το παραπάνω επιστρέψει στην έξοδο το ονοματεπώνυμο του υποψήφιου(ελέγχοντας το ακραίο σεναρίο να υπάρχουν 2 υποψήφιοι που ισοβαθμούν ΚΑΙ έχουν το 50% των ψήφων) .

ΑΛΓΟΡΙΘΜΟΣ 1

```
1  function MajorityFinder(A[1...n])
2   majority_person = [array of 2 elements]
3   maxcount = 0
4   count
5   temp
6   for(i = 1 to n)
7           count = 0
8           temp = A[i]
9           for(j = 1 to n)
10                  if(temp == A[j])
11                          count++
12          if(count > maxcount)
13                  maxcount = count
14                  majority_person[1] = temp
15                  majority_person[2] = null
16          else if((count == maxcount) AND (temp ≠ majority_person[1]))
17                  majority_person[2] = temp
18  if(maxcount ≥ ⌈n/2⌉)
19          return majority_person
20  else
21          return "no person has the majority"
```

ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΟΥ:

- Στη $1^η$ σείρα ο αλγόριθμος δέχεται τις n ψήφους μέσω μιας συστοιχίας "A[1...n]"
- Στη $2^η$ σειρα ορίζουμε τον πίνακα "majority_person" στον οποίο θα αποθηκευτεί το ονόμα του υπερέχοντα υποψήφιου , το μέγεθος του πίνακα έχει 2 θέσεις για να καλύπτει και την περίπτωση που 2 υποψήφιοι συγκεντρώνουν από 50% ο καθένας.
- Στις γραμμές 3-5 γίνονται αρχικοποιήσεις που χρησιμέυουν στη καταμέτρηση των ψήφων και των υποψήφιων με τους περισσότερους.
- Στις γραμμές 6-11 ο αλγόριθμος καταμετρεί όλες τις ψήφους .Συγκεκριμένα η αρχική η for χρησιμοποιείται για να προσπελαστούν όλες οι ψήφοι,ενώ η 2η για να γίνει έλεγχος ποιες ψηφοί έχουν το ίδιο ονοματεπώνυμο με την i-στη.Ετσι μεσω το count μετράμε τις συνολικες ψήφους που έχει ο υποψήφιος που βρισκεταιο στην i-στη θεση του αρχικου πινακα ενώ στο temp αποθηκέυται το ονοματεπώνυμο του.Επειδη εχουμε εμφωλευμενες for η πολυπολοκοτητα γινεται $O(n^2)$.

- Στις γραμμες 12-17 γίνεται ελεγχός για να βρέθει ποιος υποψήφιος εχει τις περισσοτερες ψήφους,ενω καλυπτεται η περιπτωση της ισοβαθμιας υποψηφιων μεσω του μεταβλητου πινακα "majority person"
- Στις τελευταιες γραμμες 18-21 γινεται ελεγχος αμα καποιος υποψήφιος εχει $\geq$ του 50% των ψηφών,η εξοδος ειτε θα εχει κανενα ενα ή ακομα και 2 ομνοματα στην ακραια περίπτωση που έχουμε 2 υποψηφιους με τις μισε ψηφους αμφοτεροι.
- Ανακεφαλαιώνοντας πρόκειται για εναν αργο αλγορίθμο O(n^2) καθως ελεγχονται n φορες όλα τα ονοματεπώνυμα ενώ συγκρίνουμε ολές τις ψήφους μεταξύ τους μια προς μια .Γεγονώς που θα προσπαθήσουμε να ανατρέψουμε στα επώμενα ερωτήματα

Ερώτημα 2

Merge Sort

```
function mergesort(a[1...n])
 if(n > 1)
        return merge(mergesort(a[1...⌊n/2⌋]), mergesort(a[⌊n/2⌋ + 1 ...n]))
 else
        return a

function merge(x[1...k], y[1...l])
 if(k = 0)
        return y[1...l]
 if(l = 0)
        return x[1...l]
 if(x[1] ≥ y[1])
        return x[1] ∘ merge(x[2...k], y[1...l])
 else
        return y[1] ∘ merge(x[1...k], y[2...l])
```

```
function MajorityFinder2(A[1...n])
 majority_person = []
 mergesort(A)
 for(i = 1 to n)
        if(A[i] == A[⌈n/2⌉ - 1 + i])
                if(majority_person[1] == null)
                        majority_person[1] = A[i]
                else
                        majority_person[2] = A[i]
 return majority_person
```

- Ερώτημα 3

```
function MajorityFinder3(A[1...n])
 majority_person = []
 HashMap T
 for(i = 1 to n)
        if(T.search(A[i]) == true)
                T[A[i]] = T[A[i]] + 1
        else
                T.put([A[i], 1)
        if(T[A[i]] ≥ ⌈n/2⌉)
                if(majority_person[1] == null)
                        majority_person[1] = A[i]
                else
                        majority_person[2] = A[i]
 return majority_person
```

Πρόβλημα 2

- Ερώτημα 1

Algorithm 1

Έστω πίνακας T με στοιχεία n θετικούς ακεραίους με εύρος [0,...,k] (k ακέραιος)

```
for i = 0,...,k do
        H[i] = 0
end for
for j = 1,...,n do
        H[T[j]] = H[T[j]] + 1
end for
for i = 1,...,k do
        H[i] = H[i] + H[i - 1]
end for
for j = n,...,1 do
        S[H[T[j]]] = T[j]
        H[T[j]] = H[T[j]] -1
end for
```

Ανάλυση Αλγορίθμου

- Στα βήματα (1-3) δημιουργείται ο πίνακας H με k στοιχεία (δηλαδή με μέγεθος ίσο με το εύρος των αριθμών) και αρχικοποιούνται όλα τα στοιχεία του με 0.
- Στα βήματα (4-6) χρησιμοποιείται ο πίνακας H για να μετρήσει πόσες φορές εμφανίζεται κάθε αριθμός στη λίστα T. Συγκεκριμένα διατρέχει τη λίστα T και αυξάνει το μετρητή H[T[j]] κάθε φορά που συναντά ένα στοιχείο T[j].
- Στα βήματα (7-9) εκτελεί μια σωρευτική καταμέτρηση, αθροίζει δηλαδή το H[i] με το προηγούμενό του το H[i - 1] ώστε κάθε στοιχείο του πίνακα H να αποθηκεύει το άθροισμα του στοιχείου αυτού με όλα τα προηγούμενά του. Έτσι τελικά το κάθε στοιχείο H[i] δείχνει πόσοι αριθμοί προηγούνται του αριθμού i δηλαδη του T[j].
- Στα βήματα (10-13) δημιουργείται ο τελικός πίνακα S, ο οποίος θα περιέχει τα στοιχεία του πίνακα T ταξινομημένα. Συγκεκριμένα ξεκινάει από το τέλος του πίνακα T και τοποθετεί κάθε στοιχείο στη θέση που του αντιστοιχεί στον πίνακα S με βάση τον πίνακα καταμετρητών H. Στη συνέχεια μειώνει κατά 1 τον μετρητή του στοιχείου που τοποθέτησε.

Παρακάτω ακολουθεί ένα παράδειγμα με $n = 9$ και $k = 3$. Έστω ο πίνακας T:

	1	2	3	4	5	6	7	8	9
T	3	2	2	1	3	0	0	2	3

Μετά τα βήματα (1-3) ο πίνακας H είναι ο εξής:

	0	1	2	3
H	0	0	0	0

Μετά τα βήματα (4-6) ο πίνακας H έχει αποθηκεύσει πόσες φορές εμφανίζονται οι αριθμοί του πίνακα T:

	0	1	2	3
H	2	1	3	3

Μετά τα βήματα (7-9) ο πίνακας H έχει αποθηκεύσει τα σωρευτικά αθροίσματα:

	0	1	2	3
H	2	3	6	9

Μετά τα βήματα (10-13) ο πίνακας S που προκύπτει είναι ο ταξινομημένος πίνακας T:

	1	2	3	4	5	6	7	8	9
S	0	0	1	2	2	2	3	3	3

Ερώτημα 2

Εντολές	costs (c)	times (t)
1 for i = 0,...,k do	c1	k+2
2 H[i] = 0	c2	k+1
3 end for		
4 for j=1,...,n do	c3	n+1
5 H[T[j]] = H[T[j]]+1	c4	n
6 end for		
7 for i =1,...,k do	c5	k+1
8 H[i]= H[i]+H[i-1]	c6	k
9 end for		
10 for j = n,...,1 do	c7	n+1
11 S[H[T[j]]] = T[j]	c8	n
12 H[T[j]] = H[T[j]]-1	c9	n
13 end for		

$$T(n) = \sum_{a=1}^{9} c_a \cdot t_a = c_1 \cdot (k+2) + c_2 \cdot (k+1) + c_3 \cdot (n+1) + c_4 \cdot n + c_5 \cdot (k+1) + c_6 \cdot k$$
$$+ c_7 \cdot (n+1) + c_8 \cdot n + c_9 \cdot n \implies$$

Όπου οι $c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9$ είναι κάποιες σταθερές

$$\implies T(n) = \mathcal{O}(n+k)$$