ΓΙΩΡΓΟΣ ΙΩΑΝΝΙΔΗΣ *ΑΕΜ : 10490*
ΚΥΡΙΑΚΟΣ ΤΣΟΚΚΟΣ *ΑΕΜ : 10496* **ΟΜΑΔΑ 19**
ΧΡΙΣΤΟΔΟΥΛΟΣ ΛΥΜΠΟΥΡΙΔΗΣ *ΑΕΜ : 9525*

ΠΡΟΒΛΗΜΑ 1ο

(Ερώτημα 1)

Υλοποίηση Αλγορίθμου:

```
1.  function candidate (A[1 to N])
2.
3.   for i= 1 to N
4.      cnt=0
5.        for  j=1 to N
6.             if(A[i]==A[j])
7.                cnt = cnt+1
8.             if(cnt>N/2)
9.                return A[i]
10.      end for
11.  end for
12.
13.print candidate(A)
```

Περιγραφή-Σχόλια:

Αρχικά για να έχουμε χρόνο εκτέλεσης $O(n^2)$ κάναμε μία συνάρτηση candidate η οποία υλοποιείται με δύο βρόγχους for. Η συνάρτηση candidate δέχεται ως όρισμα ένα πίνακα με N ψήφους και επιστρέφει τον υποψήφιο με περισσότερο από το 50% των ψήφων. Έπειτα καλούμε την συνάρτηση και εκτυπώνουμε τον υποψήφιο αυτόν, αν υπάρχει.

Πιο συγκεκριμένα, χρησιμοποιήσαμε διπλό βρόγχο for έτσι ώστε να συγκρίνω κάθε στοιχείο με ολόκληρο τον πίνακα. Επίσης έχουμε ένα μετρητή ο οποίος μετρά τους ψήφους του κάθε υποψηφίου και μηδενίζεται σε κάθε επανάληψη. Στη γραμμή 6 ελέγχω αν το i-στο στοιχείο ισούται με το j-στο στοιχείο του πίνακα, άρα συγκρίνω το i με ολόκληρο τον πίνακα και τον μετρητή να αυξάνεται κάθε φορά που βρίσκω το ίδιο στοιχείο (γραμμή 7). Έπειτα στην γραμμή 8 ελέγχουμε αν ο μετρητής μας έχει ξεπεράσει το 50% των ψήφων, αν όντως ο μετρητής έχει ξεπεράσει το 50% τότε επιστρέφει το όνομα του υποψηφίου αυτού. Στην περίπτωση που κανένας υποψήφιος δεν έχει ξεπεράσει το ποσοστό, δεν επιστρέφει κάτι.

Τέλος τρέχουμε τη συνάρτηση με είσοδο τον πίνακα Α και εκτυπώνω αυτό που επιστρέφει.

Ερώτημα 2

Υλοποίηση Αλγορίθμου:

```
1.  function majority(A[1 to n])
2.     If (n = 1)
3.      return A[1]
4.
5.     mid = n / 2;
6.
7.     Lmajority = majority(A[1 to mid])     // L=left  R=right
8.     Rmajority = majority(A[ mid+1 to  n])
9.
10.   if (Lmajority = Rmajority)
11.      return Lmajority
12.
13.
14.   Lpsifoi = Psifoi(Lmajority, A)
15.   Rpsifoi = Psifoi(Rmajority, A)
16.
17.   if (Lpsifoi  > n / 2)
18.       return Lmajority
19.   else if  ( Rpsifoi  >  n / 2)
20.       return Rmajority
21.    else
22.      return X
23.
24.   function Psifoi(candidate,  A[1 to n])
25.        cnt = 0
25.      for ( i= 1 to n)
27.         if (A[i]  = candidate)
28.           cnt=cnt+1
29.      end for
30.     return cnt
31.
32.    Print majority(A)
```

Περιγραφή-Σχόλια:

Για να έχουμε χρόνο εκτέλεσης O(nlogn) υλοποιήσαμε 3 συναρτήσεις οπου η η μια από αυτές (majority) χρησιμοποιεί τη τεχνική διαίρει και βασίλευε. Η συνάρτηση majority η οποία καλείται αναδρομικά με μισό πλήθος στοιχείων σε κάθε κλήση μέχρι το πλήθος των στοιχείων να φτάσει 1 (O(nlogn)). Μια συνάρτηση Psifoi που καλείται σε κάθε κλήση της majority για να βρίσκω τις ψήφους κάθε υποψήφιου(O(n)). Τέλος έχουμε τη συνάρτηση Ipopsifios που καλεί τις 2 παραπάνω συναρτήσεις, συνεπώς η συνολική πολυπλοκότητα είναι O(nlogn + n) δηλαδή O(nlogn).

Αρχικά η συνάρτηση majority με όρισμα έναν πίνακα με n στοιχεία (ψήφοι) ελέγχει αν το μέγεθος του πίνακα είναι 1 και αν ισχύει επιστρέφει το στοιχείο αυτό. Μετά βρίσκω τη μέση του πίνακα (mid) για να τον χωρίσω στα 2. Καλώ αναδρομικά τη συνάρτηση μέχρι ο υποπίνακας να εχει 1 στοιχείο (για να επιστραφεί). Στη συνέχεια συγκρίνει το αποτέλεσμα που έχω από τον αριστερό υποπίνακα, με αυτό που έχει ο δεξιός υποπίνακας και αν είναι ίδια τα 2 στοιχεία επιστρέφει το στοιχείο αυτό (υποψήφιος) . Αλλιώς αν διαφέρουν οι 2 υποψήφιοι υπολογίζει πόσες φορές εμφανίζεται το κάθε στοιχείο στον πίνακα που έχω ως είσοδο, με την συνάρτηση Psifoi. Ελέγχω ποια από τις μεταβλητές Lpsifoi και Rpsifoi έχει περισσότερους από τους μισούς ψήφους και την επιστρέφω. Στη περίπτωση που δεν ισχύει κανένα από τα πιο πάνω τότε, επιστρέφω μια μεταβλητή X που σημαίνει πως δεν βρέθηκε κάποιος με περισσότερους από τους μισούς ψήφους.

Η συνάρτηση Psifoi έχει ως είσοδο τον υποψήφιο που θέλω να ελέγξω και τον πίνακα με τα στοιχεία(ψήφους). Με ένα for υπολογίζω πόσες φορές το στοιχείο εμφανίζεται μέσα στον πίνακα με ένα μετρητή (cnt) που έχω ήδη μηδενίσει πριν απο το for.

Τέλος τρέχουμε τη συνάρτηση majority με είσοδο τον πίνακα A και εκτυπώνω αυτό που επιστρέφει.

Ερώτημα 3

Υλοποίηση Αλγορίθμου:

```
1.  function findMajority( A[1 to N] )
2.  cnt=0
3.      for i=1 to N
4.          if  cnt=0
5.              candidate=A[i]
6.              cnt=1
7.          else if m=A[i]
8.              cnt=cnt+1
9.          else
10.             cnt=cnt-1
11.    end for
12.
13. cnt=0
14.    for i=1 to N
15.         if  A[i]=candidate
16.         cnt=cnt+1
17.    end for
18.
19. If cnt > N/2
20.   return candidate
21.
22.
23. majority = findMajority(A)
24. print majority
```

Περιγραφή-Σχόλια:

Για να έχουμε χρόνο εκτέλεσης O(n) κάναμε μία συνάρτηση findMajority η οποία υλοποιείται με 2 βρόχους for οι οποίοι δεν είναι εμφωλευμένοι. Η συνάρτηση αυτή δέχεται ως είσοδο ένα πίνακα με N ψήφους και επιστρέφει τον υποψήφιο με περισσότερο από το 50% των ψήφων. Έπειτα καλούμε την συνάρτηση και εκτυπώνουμε τον υποψήφιο αυτόν, αν υπάρχει.

Η συνάρτηση αρχικά μηδενίζει έναν μετρητή(cnt). Έπειτα με ένα for διατρέχω τον πίνακα. Αν ο μετρητής ισούται με 0 βάζω στη μεταβλητή candidate το i-στό στοιχείο του πίνακα και τον μετρητή τον κάνει 1. Αν η μεταβλητή candidate ισούται με τον πίνακα στη i-στή θέση αυξάνω τον μετρητή κατά ένα και αν δεν ισχύει καμία από τις δύο παραπάνω περιπτώσεις μειώνει τον μετρητή κατά ένα. Με αυτό το τρόπο οταν το for τελειώσει η μεταβλητή candidate θα έχει τον υποψήφιο με τις περισσότερες ψήφους. Στη συνέχεια μηδενίζω ξανά τον μετρητή και με ένα for διατρέχω ξανά τον πίνακα, ελέγχω με ένα if αν ισούται ο υποψήφιος με τις περισσότερες ψήφους με το κάθε στοιχείο του πίνακα και όποτε ισχύει αυτό αυξάνω το μετρητή μου κατά 1. Μετά το τέλος του for ελέγχω αν ο μετρητής είναι μεγαλύτερος από το 50% των ψήφων, αν ισχύει επιστρέφει τον υποψήφιο αυτόν. Τέλος τρέχω τη συνάρτηση με είσοδο τον πίνακα A και εκτυπώνω αυτό που επιστρέφει.

ΠΡΟΒΛΗΜΑ 2ο

Ερώτημα 1ο

Σχόλια

1: for i = 0, . . . , k do
2: H[i] = 0
3: end for

Το πρώτο for αρχικοποιεί τον πίνακα Η με μηδενικά

Παράδειγμα:

T	4	4	4	0	1	3	3	4	2	1	1	3

Ο πίνακας Τ δεν αλλάζει κατα την εκτέλεση του αλγορίθμου .

H	0	0	0	0	0

4: for j = 1, . . . , n do
5: H[T[j]] = H[T[j]] + 1
6: end for

Το δευτερο for διατρέχει τον πίνακα Τ και το στοιχειο του Τ δινει τη θέση του Η η οποία θα αυξηθεί κατά ενα.
Ουσιαστικά βρίσκει πόσες φορές έχω τον κάθε αριθμό στον πίνακα Τ και τον βάζει στη θέση που του αντιστοιχεί στον Η.

H	1	3	1	3	4

7: for i = 1, . . . , k do
8: H[i] = H[i] + H[i − 1]
9: end for

Διατρέχει τον Η ξεκινώντας από τη δεύτερη θέση και προσθέτει στη i-οστή θέση το το στοιχείο της προηγούμενης θέσης.
Άρα ο Η μετα το for θα μας δείχνει τις θέσεις στις οποίες θα είναι τοποθετημένο το κάθε στοιχείο του Τ στον S ώστε να είναι ταξινομημένος.

H	1	3	1	3	4

H	1	4	5	8	12
		=H[1]+H[0] =3+1=4	=H[2]+H[1] =1+4=5	=H[3]+H[2] =3+5=8	=H[4]+H[3] =4+8=12

10: for j = n, . . . , 1 do
11: S[H[T[j]]] = T[j]
12: H[T[j]] = H[T[j]] − 1
13: end for

Διατρέχει τον T αντίστροφα και κάθε στοιχειο T[j] τοποθετείται στη θέση H[T[j]] του πίνακα S. Μετά μειώνεται η τιμή του αντιστοιχου στοιχειού στον H κατά 1 έστι ώστε να τοποθετήσουμε τυχόν άλλα στοιχεια του T που έχουν ίδιο τιμή στη σωστή θέση στον S.

	1	2	3	4	5	6	7	8	9	10	11	12
T	4	4	4	0	1	3	3	4	2	1	1	3

	0	1	2	3	4
H	1	4	5	8	12

- j=12 , T[j]=T[12]=3, H[T[j]] = H[3]=8 => S[H[T[j]]] = S[8] = 3 και H[3]=8-1=7

	1	2	3	4	5	6	7	8	9	10	11	12
S	-	-	-	-	-	-	-	3	-	-	-	-

- j=11 , T[j]=T[11]=1, H[T[j]]=H[1]=4 => S[H[T[j]]]=S[4]=1 και H[1]=4-1=3.

S	-	-	-	1	-	-	-	3	-	-	-	-

- j=10 , T[j]=T[10]=1, H[T[j]]=H[1]=3 => S[H[T[j]]]=S[3]=1 και H[1]=3-1=2.

S	-	-	1	1	-	-	-	3	-	-	-	-

- j=9 , T[j]=T[9]=2, H[T[j]]=H[2]=5 => S[H[T[j]]]=S[5]=2 και H[1]=5-1=4.

S	-	-	1	1	2	-	-	3	-	-	-	-

.
.
.
.

S	0	1	1	1	2	3	3	3	4	4	4	4

Αποτέλεσμα: Τα στοιχεία του T τοποθετούνται στον πινακα S ταξινομημενα με αυξουσα σειρά.

Ερώτημα 2ο

Εφόσον έχω 2 for που τρέχουν n φορές, 1 for που τρέχει k+1 φορές και ένα for που τρέχει k φορές αρα , θα έχω χρόνο εκτέλεσης 2O(n) + O(k+1) + O(k) το οποίο ισούται με 2O(n) + 2O(k) δηλαδή O(n+k). Η πολυπλοκότητα του πιο πάνω αλγορίθμου είναι γραμμική και εξαρτάται από το μέγεθος των δύο πινάκων Η και Τ. Τέλος το k δεν πρεπει να ξεπερνα την ταξη του n και αντίστροφα.