# 32ος ΠΑΝΕΛΛΗΝΙΟΣ ΔΙΑΓΩΝΙΣΜΟΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

## ΘΕΜΑΤΑ ΤΕΛΙΚΗΣ ΦΑΣΗΣ

## Θέμα 1ο: Άθροισμα λίγων [30 Μονάδες]

Μας δίνεται μία ακολουθία αποτελούμενη από **N** θετικούς ακέραιους αριθμούς $\mathbf{x}_1$, $\mathbf{x}_2$, … $\mathbf{x}_N$ και ένας φυσικός αριθμός **K** ≤ **N**. Επιλέγουμε ένα συνεχόμενο διάστημα της ακολουθίας, έστω τους αριθμούς $\mathbf{x}_i$, $\mathbf{x}_{i+1}$, … $\mathbf{x}_j$ (για i ≤ j), μέσα στο οποίο όμως να μην εμφανίζονται περισσότεροι από **K** διαφορετικοί αριθμοί, και τους αθροίζουμε. Ποιο είναι το μέγιστο άθροισμα που μπορούμε να πετύχουμε με αυτόν τον τρόπο;

### Πρόβλημα:

Να αναπτύξετε ένα πρόγραμμα σε μια από τις γλώσσες της IOI (PASCAL, C, C++, Java) το οποίο θα διαβάζει τις τιμές των **N** και **K** καθώς και την ακολουθία των αριθμών και θα εκτυπώνει την απάντηση στο παραπάνω ερώτημα.

### Αρχεία Εισόδου:

Το αρχείο εισόδου με όνομα **sumoffew.in** είναι αρχείο κειμένου με την εξής δομή. Η πρώτη γραμμή έχει δύο ακέραιους αριθμούς **N** και **K**, χωρισμένους με ένα κενό διάστημα. Ο αριθμός **N** θα παριστάνει το πλήθος των αριθμών της ακολουθίας ενώ ο αριθμός **K** το μέγιστο επιτρεπτό πλήθος διαφορετικών αριθμών στο διάστημα που θα επιλέξουμε. Η επόμενη περιέχει **N** θετικούς ακέραιους αριθμούς, χωρισμένους ανά δύο με ένα κενό διάστημα: τους όρους της ακολουθίας.

### Αρχεία Εξόδου:

Το αρχείο εξόδου με όνομα **sumoffew.out** είναι αρχείο κειμένου που περιέχει μία μόνο γραμμή με έναν ακέραιο αριθμό: την απάντηση στο παραπάνω ερώτημα.

### Παράδειγμα Αρχείων Εισόδου - Εξόδου:

**1ο**

| sumoffew.in | sumoffew.out |
|---|---|
| 10 1<br>9 5 5 5 5 5 9 9 9 17 | 27 |

*Εξήγηση:* Αφού K=1, ψάχνουμε το διάστημα που αποτελείται από συνεχόμενες εμφανίσεις του ίδιου αριθμού και το οποίο έχει το μέγιστο άθροισμα. Τα τρία συνεχόμενα 9 δίνουν άθροισμα 27 και αυτή είναι η απάντηση εδώ.

**2ο**

| sumoffew.in | sumoffew.out |
|---|---|
| 10 3<br>11 17 12 12 17 1 12 19 18 1 | 71 |

*Εξήγηση:* Εδώ K=3. Στο διάστημα 17, 12, 12, 17, 1, 12 εμφανίζονται μόνο τρεις διαφορετικοί αριθμοί (1, 12, 17). Το άθροισμα είναι 17+12+12+17+1+12=71. Δεν υπάρχει άλλο διάστημα με το πολύ τρεις διαφορετικούς αριθμούς και μεγαλύτερο άθροισμα.

**3ο**

| sumoffew.in | sumoffew.out |
|---|---|
| 7 4<br>1 6 3 8 2 4 7 | 21 |

**4ο**

| sumoffew.in | sumoffew.out |
|---|---|
| 9 4<br>1 2 3 2 3 1 3 1 2 | 18 |

**Περιορισμοί:**

- $1 \le K \le N \le 1.000.000$
- Το άθροισμα όλων των $\mathbf{x}_i$ δε θα υπερβαίνει το 1.000.000.000
- Για περιπτώσεις ελέγχου συνολικής αξίας 10%, θα είναι:
  $1 \le N \le 1000$ και $K = 1$
- Για περιπτώσεις ελέγχου συνολικής αξίας 50%, θα είναι:
  $1 \le N \le 1000$, δηλαδή η ακολουθία θα είναι σχετικά μικρή
- Για περιπτώσεις ελέγχου συνολικής αξίας 70%, θα είναι:
  $\mathbf{x}_i \le 1.000.000$, δηλαδή οι όροι της ακολουθίας θα είναι σχετικά μικροί

**Προσοχή!** Φροντίστε να διαβάζετε την είσοδο και να εκτυπώνετε την έξοδο αποδοτικά, ειδικά αν προγραμματίζετε σε C++ ή Java.

**Μορφοποίηση:** Στην έξοδο, όλες οι γραμμές τερματίζουν με ένα χαρακτήρα newline.
**Μέγιστος χρόνος εκτέλεσης:** 1 sec.
**Μέγιστη διαθέσιμη μνήμη:** 64 MB.

## Θέμα 2ο: Η τελευταία πίστα [30 Μονάδες]

Παίζουμε ένα ηλεκτρονικό παιχνίδι που αποτελείται από **N** πίστες (επίπεδα δυσκολίας), αριθμημένες από 1 έως N. Το παιχνίδι ξεκινάει υποχρεωτικά από την πίστα **S** και τερματίζεται μόλις φτάσουμε στην πίστα **T**. Κάθε πίστα γενικά μπορεί να έχει πολλές εξόδους, που κάθε μία οδηγεί σε κάποια άλλη πίστα. Γνωρίζουμε όλες τις εξόδους και, για κάθε πίστα, έχουμε υπολογίσει πόσο χρόνο χρειαζόμαστε για να «κινηθούμε» μέσα σε αυτήν και να βγούμε από κάθε συγκεκριμένη έξοδό της. Ο χρόνος αυτός μπορεί να διαφέρει (δηλαδή, αν βρισκόμαστε στην πίστα 1, είναι πιθανό να χρειαζόμαστε 3 λεπτά για να βγούμε από την έξοδο που οδηγεί στην πίστα 17 και 6 λεπτά για να βγούμε από την έξοδο που οδηγεί στην πίστα 42).

Στο παιχνίδι όμως υπάρχει ένας ακόμα περιορισμός. Υπάρχουν δύο συγκεκριμένες πίστες, η **P** και η **Q**, και το παιχνίδι δε μας επιτρέπει να πάμε στην πίστα Q αν πρώτα δεν έχουμε πάει στην πίστα P.

Θέλουμε να ολοκληρώσουμε το παιχνίδι στον ελάχιστο δυνατό χρόνο. Δε χρειάζεται να τελειώσουμε όλες τις πίστες, αρκεί να φτάσουμε στην T.

### Πρόβλημα

Να αναπτύξετε ένα πρόγραμμα σε μια από τις γλώσσες της IOI (Pascal, C, C++, Java) το οποίο θα διαβάζει την περιγραφή των πιστών του παιχνιδιού και θα εκτυπώνει τον ελάχιστο δυνατό χρόνο στον οποίο μπορούμε να ολοκληρώσουμε το παιχνίδι.

### Αρχεία εισόδου:

Το αρχείο εισόδου με όνομα **supgame.in** είναι αρχείο κειμένου με την εξής δομή. Η πρώτη γραμμή του περιέχει έξι ακέραιους αριθμούς **N**, **M**, **S**, **T**, **P**, **Q**, χωρισμένους ανά δύο με ένα κενό διάστημα: N είναι το πλήθος των πιστών, M είναι το συνολικό πλήθος των εξόδων που οδηγούν από μία πίστα σε μία άλλη, S και T είναι η αρχική και η τελική πίστα, P και Q είναι οι δύο πίστες που αναφέρει ο παραπάνω περιορισμός.

Κάθε μία από τις επόμενες M γραμμές περιέχει τρεις ακέραιους αριθμους $\mathbf{X}_i$, $\mathbf{Y}_i$ και $\mathbf{W}_i$ χωρισμένους ανά δύο με ένα κενό διάστημα. Η τριάδα αυτή περιγράφει μία έξοδο: παίζοντας την πίστα $X_i$ μπορούμε να βγούμε από την έξοδο $Y_i$ σε χρόνο ίσο με $T_i$ λεπτά.

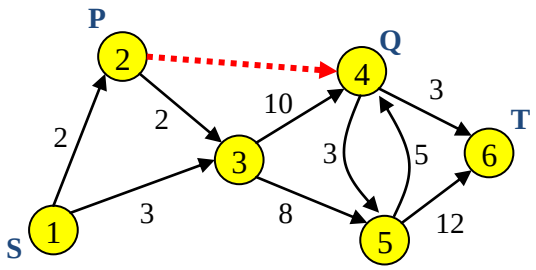
Θεωρήστε δεδομένο ότι θα είναι δυνατό να ολοκληρώσουμε το παιχνίδι.

### Αρχεία εξόδου:

Το αρχείο εξόδου με όνομα **supgame.out** είναι αρχείο κειμένου με την εξής δομή. Έχει ακριβώς μία γραμμή που περιέχει ακριβώς έναν ακέραιο αριθμό: τον ελάχιστο χρόνο, σε λεπτά, που απαιτείται για την ολοκλήρωση του παιχνιδιού.

## Παράδειγμα αρχείων εισόδου – εξόδου

| supgame.in | supgame.out |
|---|---|
| 6 9 1 6 2 4<br>1 2 2<br>2 3 2<br>1 3 3<br>3 4 10<br>3 5 8<br>4 5 3<br>5 4 5<br>4 6 3<br>5 6 12 | 17 |




*Εξήγηση:* Το παιχνίδι αντιστοιχεί στο διπλανό σχήμα. Μπορούμε να παίξουμε τις πίστες με την εξής σειρά: 1, 2, 3, 4, 6. Θα χρειαστούμε 2+2+10+3=17 λεπτά. Ο περιορισμός ικανοποιείται, γιατί όταν φτάνουμε στην πίστα 4 έχουμε ήδη περάσει από την πίστα 2. Δεν είναι δυνατό να τα καταφέρουμε σε λιγότερα από 17λεπτά.

## Περιορισμοί:

- $2 \le N \le 60.000$, $1 \le M \le 200.000$
- $1 \le S \le N$, $1 \le T \le N$, $S \ne T$
- $1 \le P \le N$, $1 \le Q \le N$, $P \ne Q$
- $1 \le X_i \le N$, $1 \le Y_i \le N$, $X_i \ne Y_i$
- $1 \le W_i \le 50.000$
- Για περιπτώσεις ελέγχου συνολικής αξίας 30%, η βέλτιστη διαδρομή από την πίστα S προς την πίστα T δε θα περνάει από την πίστα Q.

**Προσοχή!** Φροντίστε να διαβάζετε την είσοδο και να εκτυπώνετε την έξοδο αποδοτικά, ειδικά αν προγραμματίζετε σε C++ ή Java.

**Μορφοποίηση:** Στην έξοδο, όλες οι γραμμές τερματίζουν με ένα χαρακτήρα newline.
**Μέγιστος χρόνος εκτέλεσης:** 1 sec.
**Μέγιστη διαθέσιμη μνήμη:** 256 MB.

## Θέμα 3ο: Push-Complement-Reverse [40 Μονάδες]

Μας δίνεται μία ακολουθία αποτελούμενη από **N** αριθμούς $\mathbf{x}_1$, $\mathbf{x}_2$, … $\mathbf{x}_N$ που καθένας ανήκει στο σύνολο {0, 1, 2, 3}. Παίζουμε το εξής παιχνίδι για έναν παίκτη και δύο φύλλα χαρτί (το αριστερό και το δεξιό):

1. Ξεκινάμε γράφοντας την ακολουθία των αριθμών στο αριστερό χαρτί. Το δεξιό χαρτί είναι κενό.
2. Επιλέγουμε μία από τις ακόλουθες τρεις κινήσεις:
   - 'p' (push): σβήνουμε τον πρώτο αριθμό της ακολουθίας του αριστερού χαρτιού και τον γράφουμε τελευταίο στο δεξιό χαρτί.
   - 'c' (complement): αντικαθιστούμε όλους τους αριθμούς του αριστερού χαρτιού με τους συμπληρωματικούς τους ως προς 3. Δηλαδή, ο αριθμος **x** αντικαθίσταται από 3−**x**.
   - 'r' (reverse): αντιστρέφουμε τη σειρά των αριθμών του δεξιού χαρτιού. Δηλαδή ο πρώτος γίνεται τελευταίος, κ.ο.κ.
3. Το παιχνίδι τελειώνει όταν διαγραφούν όλοι οι αριθμοί από το αριστερό χαρτί.
4. Ο παίκτης κερδίζει αν στην ακολουθία αριθμών του δεξιού χαρτιού στο τέλος του παιχνιδιού όλα τα 0 βρίσκονται σε γειτονικές θέσεις, όλα τα 1 σε γειτονικές θέσεις, κ.ο.κ.

Ο σκοπός μας είναι φυσικά να βρούμε μία ακολουθία κινήσεων με την οποία ο παίκτης να κερδίζει. Όμως:

- Αν αυτό μπορούμε να το καταφέρουμε με περισσότερους του ενός τρόπους, θέλουμε να κάνουμε τις λιγότερες δυνατές κινήσεις.
- Τέλος, αν και αυτό μπορούμε να το καταφέρουμε με περισσότερους του ενός τρόπους, επιλέγουμε τη λεξικογραφικά μικρότερη ακολουθία κινήσεων.

Ονομάζουμε «βέλτιστη» την ακολουθία κινήσεων που κερδίζει το παιχνίδι και ικανοποιεί τα παραπάνω κριτήρια.

Για παράδειγμα, ας ξεκινήσουμε με αρχική ακολουθία 2, 3, 2, 1, 0. Ας υποθέσουμε ότι ο παίκτης επιλέγει να παίζει συνέχεια την κίνηση 'p'. Τότε, στο τέλος του παιχνιδιού, η ακολουθία στο δεξιό χαρτί θα είναι φυσικά και πάλι η 2, 3, 2, 1, 0. Επομένως, ο παίκτης δεν κερδίζει, καθώς το ζεύγος των αριθμών 2 δε βρίσκονται σε γειτονικές θέσεις (παρεμβάλλεται ένα 3).

Αντίθετα, αν ο παίκτης επιλέξει να παίξει τις παρακάτω κινήσεις:

| *Κίνηση* | *Αριστερό χαρτί* | *Δεξιό χαρτί* |
|---|---|---|
| | 2, 3, 2, 1, 0 | |
| p | 3, 2, 1, 0 | 2 |
| p | 2, 1, 0 | 2, 3 |
| r | 2, 1, 0 | 3, 2 |
| p | 1, 0 | 3, 2, 2 |
| c | 2, 3 | 3, 2, 2 |
| p | 3 | 3, 2, 2, 2 |
| r | 3 | 2, 2, 2, 3 |
| p | | 2, 2, 2, 3, 3 |

τότε στο τέλος του παιχνιδιού τόσο τα 2 όσο και τα 3 βρίσκονται σε γειτονικές θέσεις. Επομένως, η ακολουθία αυτή κερδίζει. Όμως δεν είναι βέλτιστη: η ακολουθία κινήσεων 'pprppp' κερδίζει επίσης και έχει μικρότερο μήκος.

## Πρόβλημα

Να αναπτύξετε ένα πρόγραμμα σε μια από τις γλώσσες της IOI (Pascal, C, C++, Java) το οποίο θα διαβάζει μερικές αρχικές ακολουθίες αριθμών και για κάθε μία από αυτές θα εκτυπώνει τη βέλτιστη ακολουθία κινήσεων.

## Αρχεία εισόδου:

Το αρχείο εισόδου με όνομα **pcr.in** είναι αρχείο κειμένου. Η πρώτη γραμμή περιέχει δύο ακέραιους αριθμούς **T** και **N**, χωρισμένους με ένα κενό διάστημα. Ο αριθμός T παριστάνει το πλήθος των ακολουθιών που θα ακολουθήσουν, ενώ ο αριθμός N παριστάνει το πλήθος των αριθμών κάθε ακολουθίας. Κάθε μία από τις επόμενες T γραμμές περιέχει N αριθμούς χωρισμένους ανά δύο με ένα κενό διάστημα: τους όρους $\mathbf{x}_1$, $\mathbf{x}_2$, … $\mathbf{x}_N$ μίας αρχικής ακολουθίας, όπου $x_i \in \{0, 1, 2, 3\}$.

## Αρχεία εξόδου:

Το αρχείο εξόδου με όνομα **pcr.out** είναι αρχείο κειμένου με την εξής δομή. Έχει ακριβώς T γραμμές που κάθε μία περιέχει τη βέλτιστη ακολουθία κινήσεων για την αντίστοιχη αρχική ακολουθία της εισόδου (βλ. παραπάνω). Οι συμβολοσειρές αυτές θα αποτελούνται από τα γράμματα 'p', 'c' και 'r'.

## Παράδειγμα αρχείων εισόδου – εξόδου

| pcr.in | pcr.out |
|---|---|
| 6 5<br>1 3 1 0 2<br>1 1 3 0 2<br>0 3 3 0 3<br>2 1 1 3 2<br>1 0 3 0 1<br>1 1 3 1 3 | pprppp<br>ppppp<br>pcppcpp<br>pcpppp<br>ppcppp<br>pppcpp |

## Περιορισμοί:

- $1 \leq T \leq 20$, $1 \leq N \leq 100.000$
- $T \cdot N \leq 100.000$
- Για περιπτώσεις ελέγχου συνολικής αξίας 50%, θα είναι $1 \leq N \leq 100$

**Προσοχή!** Φροντίστε να διαβάζετε την είσοδο και να εκτυπώνετε την έξοδο αποδοτικά, ειδικά αν προγραμματίζετε σε C++ ή Java.

**Μορφοποίηση:** Στην έξοδο, όλες οι γραμμές τερματίζουν με ένα χαρακτήρα newline.
**Μέγιστος χρόνος εκτέλεσης:** 2 sec.
**Μέγιστη διαθέσιμη μνήμη:** 256 MB.

**ΝΑ ΑΠΑΝΤΗΣΕΤΕ ΚΑΙ ΣΤΑ ΤΡΙΑ ΘΕΜΑΤΑ**

**ΔΙΑΡΚΕΙΑ ΕΞΕΤΑΣΗΣ ΤΕΣΣΕΡΙΣ (4) ΩΡΕΣ**

# ΚΑΛΗ ΣΑΣ ΕΠΙΤΥΧΙΑ

**Ακολουθούν χρήσιμες οδηγίες !**

**Διαβάστε τις ακόλουθες παρατηρήσεις προσεκτικά!**

- ✓ Ερωτήσεις που αφορούν τις παρατηρήσεις αυτές δεν θα απαντηθούν. Η πιστή τήρηση των αναφερόμενων οδηγιών είναι απαραίτητη.
- ✓ Οι αναφερόμενοι σε κάθε θέμα χρόνοι είναι ενδεικτικοί. Η επιτροπή μπορεί να τους αυξομειώσει προκειμένου να επιτύχει καλύτερη κλιμάκωση της βαθμολογίας.

1. Στην αρχή του πηγαίου κώδικά σας, θα πρέπει να χρησιμοποιήσετε τις επικεφαλίδες, ανάλογα με το πρόβλημα πχ.:

```
/*
USER: username
LANG: C
TASK: sumoffew
*/                  για κώδικα σε C

/*
USER: username
LANG: C++
TASK: sumoffew
*/                  για κώδικα σε C++

(*
USER: username
LANG: PASCAL
TASK: sumoffew
*)                  για κώδικα σε PASCAL

/*
USER: username
LANG: Java
TASK: sumoffew
*/                  για κώδικα σε Java
```

2. Έλεγχος τιμών δεν απαιτείται. Οι τιμές των αρχείων ελέγχου είναι πάντα έγκυρες.
3. Το σύστημα αξιολόγησης «τρέχει» σε Linux. Σας προτείνουμε να δοκιμάζετε τις λύσεις σας στο σύστημα. Έχετε δικαίωμα πολλαπλών υποβολών μέχρι το τέλος του

διαγωνισμού. Μετά από κάθε υποβολή θα λαμβάνετε την αξιολόγηση της λύσης σας, σε τμήμα των Αρχείων Ελέγχου.

4. Οι επιλογές του μεταγλωττιστή που χρησιμοποιούνται για τη βαθμολόγηση είναι οι εξής:
   - `C: gcc -std=c99 -O2 -DCONTEST -s -static -lm`
   - `C++: g++ -std=c++11 -O2 -DCONTEST -s -static -lm`
   - `Free Pascal: fpc -O2 -dCONTEST -XS`
   - `Java: javac`
5. Το Linux ξεχωρίζει μεταξύ κεφαλαίων και πεζών γραμμάτων. Ελέγξτε ότι τα ονόματα των αρχείων εισόδου και εξόδου είναι γραμμένα με μικρά (πεζά) γράμματα.
6. Τα προγράμματά σας πρέπει να επιστρέφουν ως κωδικό εξόδου το μηδέν.
7. Για προγραμματισμό σε C και C++ η συνάρτηση main() πρέπει πάντα να τερματίζει με τις εντολές "return(0);" ή "exit(0);".
8. Οι προγραμματιστές σε Pascal πρέπει να χρησιμοποιούν την εντολή "halt" μόνο με κωδικό εξόδου το μηδέν (μόνο δηλαδή με την μορφή "halt;" ή "halt(0);").
9. Το πρόγραμμα αξιολόγησης θα εξετάσει την τιμή που επιστρέφει το πρόγραμμά σας. Εάν η τιμή αυτή δεν είναι μηδέν, τότε το πρόγραμμα δεν θα βαθμολογηθεί για το συγκεκριμένο test.
10. Κανένας άλλος χαρακτήρας εκτός του χαρακτήρα νέας γραμμής (newline) (χαρακτήρας 0A στο ASCII εκφρασμένο στο δεκαεξαδικό σύστημα αρίθμησης, \n για προγραμματιστές C, C++ ή Java, $0A για προγραμματιστές Pascal) δεν θα υπάρχει μετά τον τελευταίο αριθμό κάθε γραμμής των αρχείων εισόδου και εξόδου. Δηλαδή, κάθε γραμμή των αρχείων εισόδου και εξόδου, συμπεριλαμβανομένης και της τελευταίας, τερματίζεται με τον χαρακτήρα νέας γραμμής όπως ορίστηκε παραπάνω.
11. Για προγραμματισμό σε Java, το αρχείο πηγαίου κώδικα πρέπει να περιέχει μόνο μία public class, με όνομα το ίδιο με το κωδικό όνομα του προβλήματος (με πεζά γράμματα, π.χ. sumoffew). Το αρχείο πρέπει να ονομάζεται με το ίδιο όνομα της κλάσης και κατάληξη .java. Αν χρειαστεί να ορίσετε επιπλέον κλάσεις, θα πρέπει να φροντίσετε να μην είναι public.

✓ Κάθε απόπειρα κακόβουλης εισόδου ή ακόμα και εξερεύνησης του συστήματος, εκτός της παρεχόμενης διεπαφής, θα εντοπίζεται και θα επιβάλλονται κυρώσεις.

Με τη συνεργασία:

Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης, Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών, Εθνικού Μετσόβιου Πολυτεχνείου, Πανεπιστημίου Αιγαίου, Πανεπιστημίου Ιωαννίνων, Πανεπιστημίου Πατρών, Πανεπιστημίου Πειραιώς, ΤΕΙ Αθήνας.