ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΑΡΑΛΛΗΛΟ ΥΠΟΛΟΓΙΣΜΟ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΣΚΗΣΗ 1

ΣΤΟΙΧΕΙΑ ΦΟΙΤΗΤΗ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ : ΑΘΑΝΑΣΙΟΥ ΒΑΣΙΛΕΙΟΣ ΕΥΑΓΓΕΛΟΣ
ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ : 19390005
ΕΞΑΜΗΝΟ ΦΟΙΤΗΤΗ : 7ο
ΚΑΤΑΣΤΑΣΗ ΦΟΙΤΗΤΗ : ΠΡΟΠΤΥΧΙΑΚΟ
ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ : ΠΑΔΑ
ΤΜΗΜΑ ΕΡΓΑΣΤΗΡΙΟΥ : Ε3 ΔΕΥΤΕΡΑ 14:00 – 16:00
ΚΑΘΗΓΗΤΗΣ ΕΡΓΑΣΤΗΡΙΟΥ : ΙΟΡΔΑΝΑΚΗΣ ΜΙΧΑΗΛ
ΗΜΕΡΟΜΗΝΙΑ ΠΑΡΑΔΟΣΗΣ : 11/12/2022

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΑΡΑΛΛΗΛΟ ΥΠΟΛΟΓΙΣΜΟ

ΦΩΤΟΓΡΑΦΙΑ ΦΟΙΤΗΤΗ :

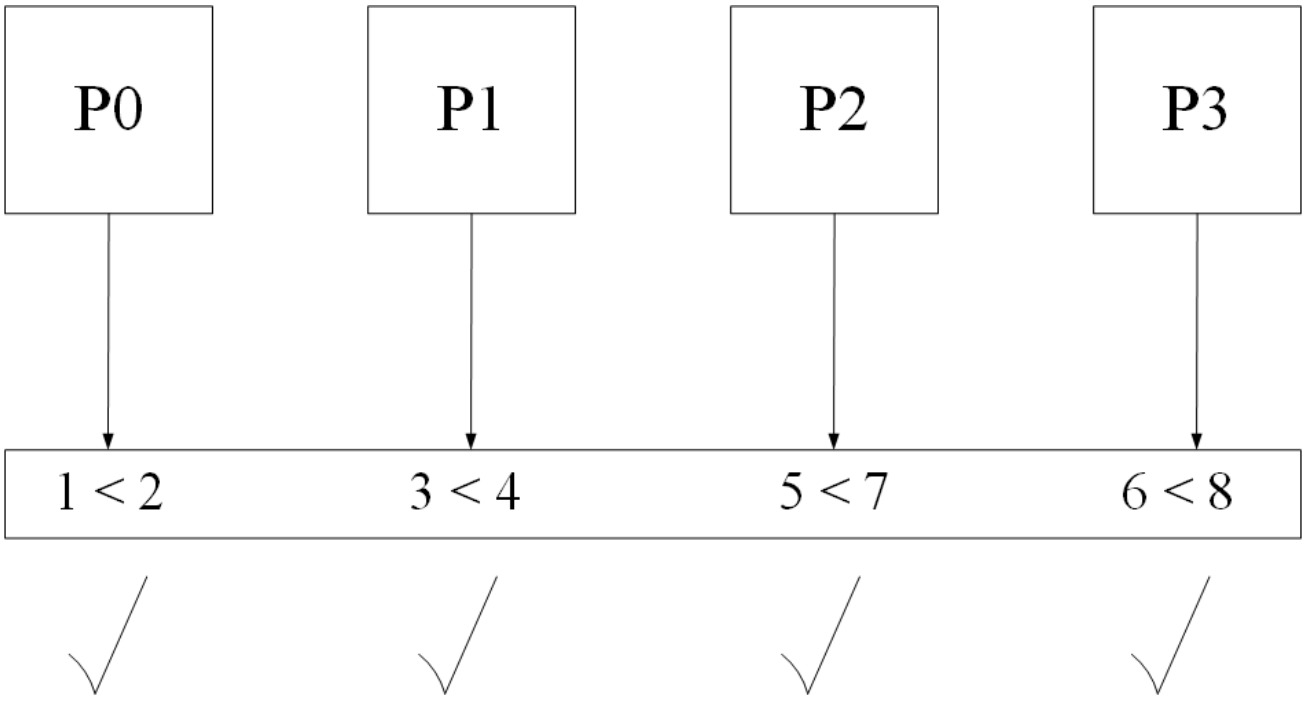
ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΑΡΑΛΛΗΛΟ ΥΠΟΛΟΓΙΣΜΟ

Το ζητούμενο

Η άσκηση αποσκοπεί στην επίτευξη της «point-to-point» επικοινωνίας μεταξύ «p» διεργασιών έχοντας ως αφορμή τον έλεγχο ταξινόμησης μίας ακολουθίας «Seq» μεγέθους «N». Πιο συγκεκριμένα, τα στοιχεία της ακολουθίας πρέπει να μοιραστούν ισοκατανεμημένα στους επεξεργαστές, έχοντας εξασφαλίσει πρώτα ότι το μέγεθος της ακολουθίας θα είναι ακέραιο πολλαπλάσιο του πλήθους των επεξεργαστών, και στη συνέχεια να ελεγχθούν παράλληλα, αν τα στοιχεία τους είναι ταξινομημένα κατά αύξουσα σειρά. Αν δεν είναι, τότε να επιστρέφεται στην διεργασία P0 που θα είναι και η κύρια, το πρώτο στοιχείο για το οποίο χαλάει η ταξινόμηση της ακολουθίας. Τέλος, η P0 θα τυπώνει την συμβολοσειρά «yes» στην περίπτωση που η ακολουθία είναι ταξινομημένη κατά αύξουσα σειρά ή την συμβολοσειρά «no» στην περίπτωση που η ακολουθία δεν είναι ταξινομημένη κατά αύξουσα σιερά και επιπρόσθετα θα τυπώνει και το πρώτο στοιχείο που χαλάει την ταξινόμηση. Η διαδικασία αυτή θα εκτελείται επαναληπτικά με ένα μενού επιλογών για συνέχεια ή για έξοδο της διαδικασίας. Την επιλογή θα την δίνει ο χρήστης.

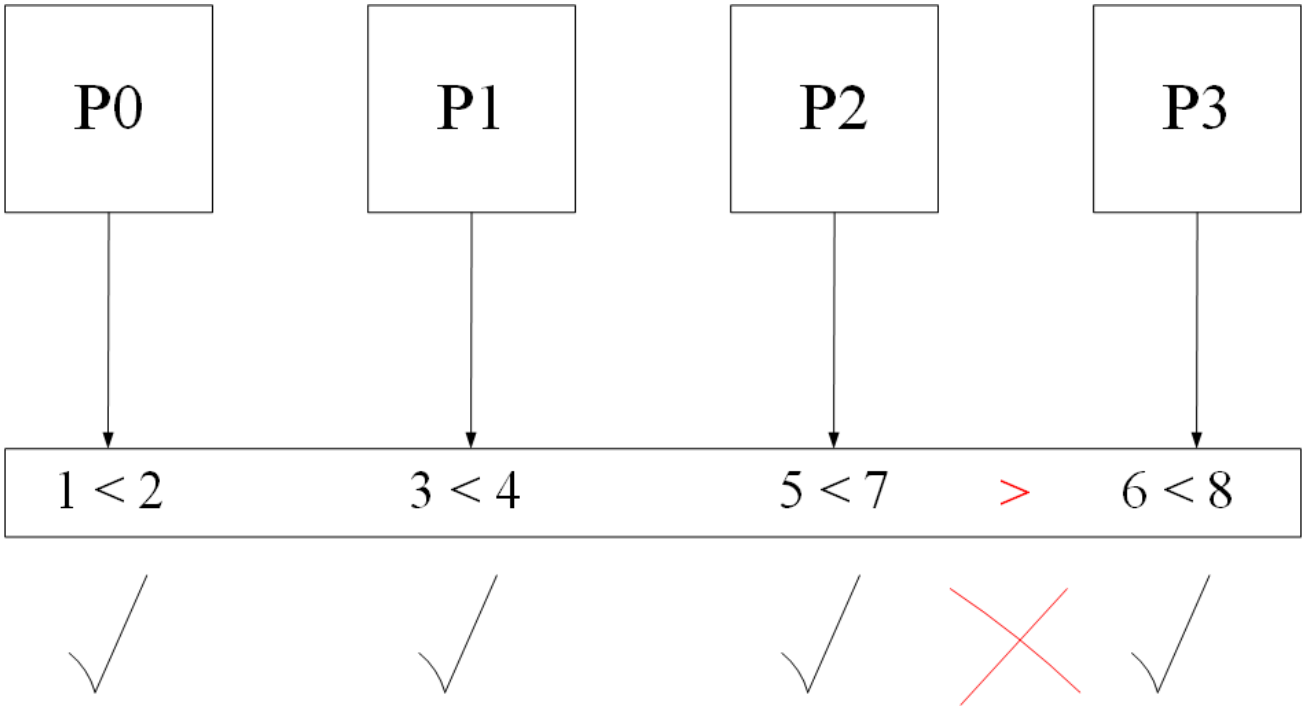
Το πρόβλημα και η υλοποίηση σε φυσική γλώσσα

Η διεργασία «P0» αναλαμβάνει καθήκοντα «διαχειριστή», δηλαδή, θα μοιράσει ισοκατανεμημένα τα στοιχεία στις υπόλοιπες διεργασίες, ώστε να πάρει και αυτή τον ίδιο αριθμό στοιχείων. Σε πρώτη φάση οι διεργασίες θα έλεγξουν τα στοιχεία τους αν είναι ταξινομημένα κατά αύξουσα σειρά. Για παράδειγμα, έχουμε 4 επεξεργαστές (P0, P1, P2, P3) και την ακολουθία «1, 2, 3, 4, 5, 7, 6 ,8» μεγέθους 8. Για τον έλεγχο της ταξινόμησης ο κάθε επεξεργαστής θα πάρει από 8 / 4 = 2 στοιχεία. Όπως φαίνεται και στην Εικόνα 1, ο «P0» θα πάρει τα στοιχεία 1, 2, ο «P1» τα 3, 4, ο «P2» τα 5, 7 και ο «P3» τα 6, 8.



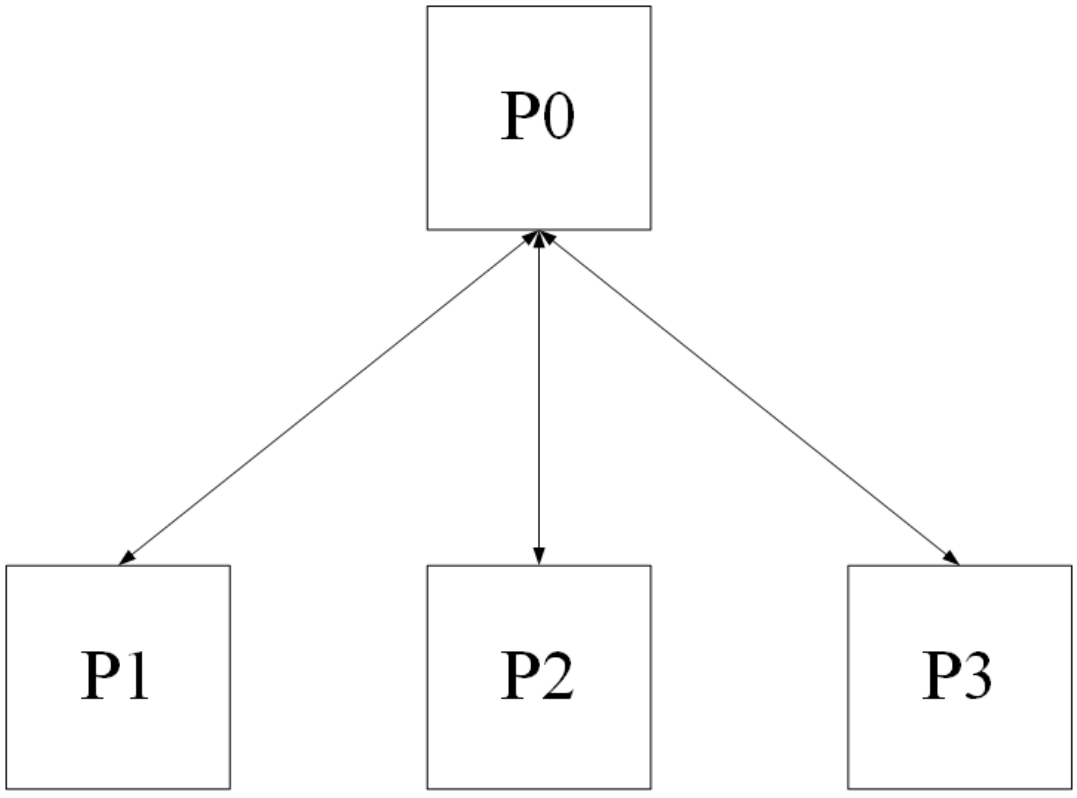
Εικόνα 1. Έλεγχος των στοιχείων του κάθε επεξεργαστή

Τα στοχεία και των τεσσάρων διεργασιών είναι ταξινομημένα κατά αύξουσα σειρά μεμονωμένα. Αυτό δεν οδηγεί όμως στο συμπέρασμα ότι όλη η ακολουθία είναι ταξινομημένη, καθώς υπάρχουν στοιχεία που δεν έχουν ελεγχθεί μεταξύ τους με αποτέλεσμα να δημιουργείται αυτό το πρόβλημα της Εικόνας 2.



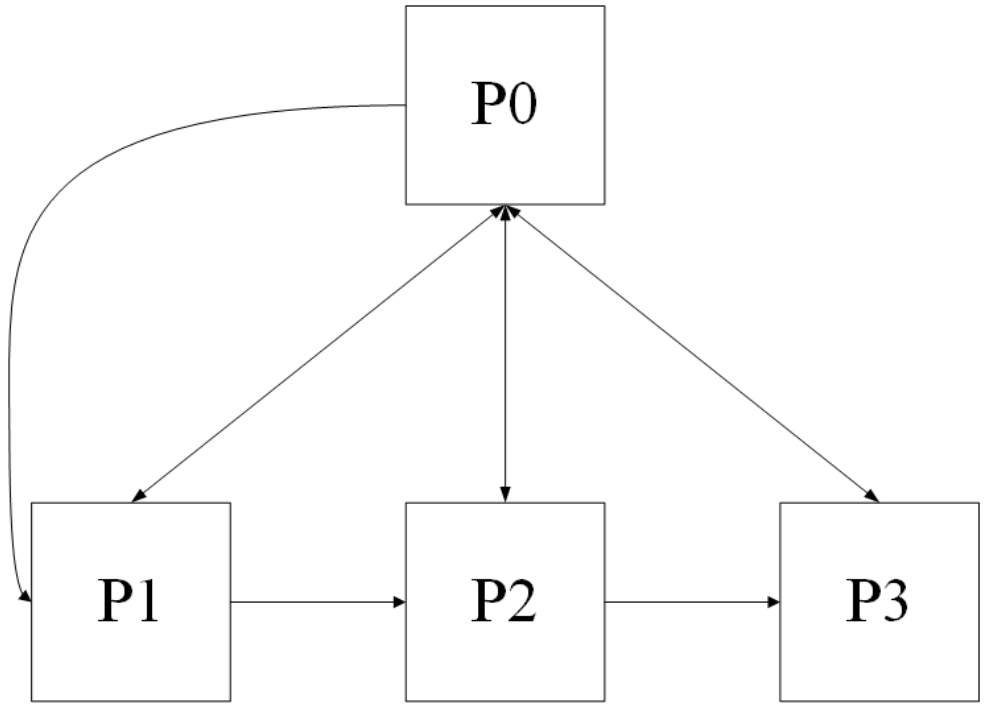
Εικόνα 2. Το πρόβλημα με την απουσία ελέγχου των ενδιάμεσων στοιχείων

Παρόλου που οι διεργασίες δεν εντόπισαν στοιχείο που χαλάει την ταξινόμηση των στοιχείων της, η ακολουθία δεν είναι ταξινομημένη κατά αύξουσα σειρά. Αυτό οφείλεται στο γεγονός ότι η επικοινωνία διεργασιών βασίζεται μόνο μεταξύ της «P0» και των υπολοίπων όπως φαίνεται στην Εικόνα 3.



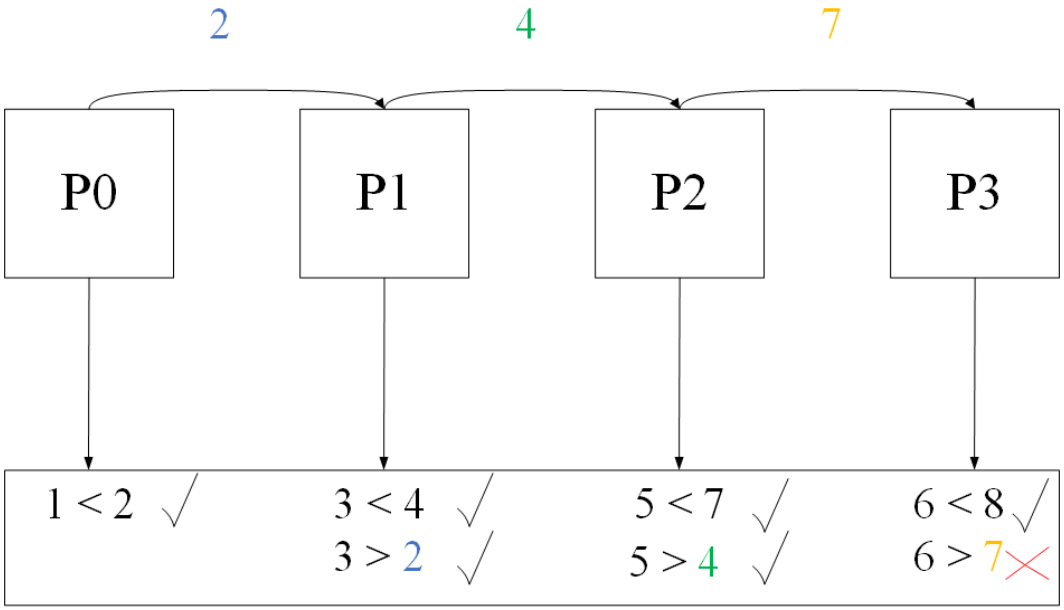
Εικόνα 3. Το πρόβλημα με την απουσία ελέγχου των ενδιάμεσων στοιχείων με διάγραμμα

Συνεπώς, το ζητούμενο δεν είναι μόνο η «P0» να επικοινωνεί με τις άλλες διεργασίες, αλλά και οι ίδιες οι διεργασίες να επικοινωνούν μεταξύ τους ώστε να ενημερώνουν για ένα δικό τους στοιχείο (το τελευταίο). Επομένως, η ορθή «point-to-point» επικοινωνία για την λύση του προβλήματος των ενδιάμεσων στοιχείων είναι αυτή της Εικόνας 4.

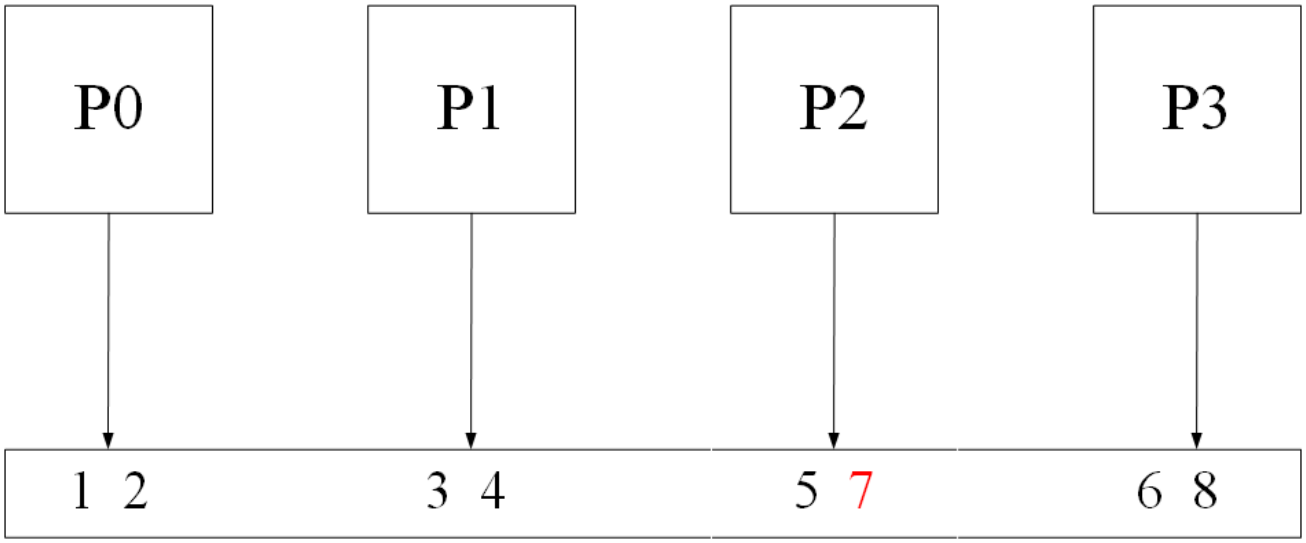


Εικόνα 4. Η λύση στο πρόβλημα με την απουσία ελέγχου των ενδιάμεσων στοιχείων με διάγραμμα

Τώρα, πλέον ορθά βγαίνει το πόρισμα ότι η ακολουθία δεν είναι ταξινομημένη (Εικόνα 5) και μάλιστα εντοπίζεται και το πρώτο στοιχείο που χαλάει την ταξινόμηση (Εικόνα 6).



Εικόνα 5. Η λύση στο πρόβλημα με την απουσία ελέγχου των ενδιάμεσων στοιχείων



Εικόνα 6. Το πρώτο στοιχείο που χαλάει την ταξινόμηση

Το υπολογιστικό φόρτο της κάθε διεργασίας

Στον παρακάτω πίνακα αναφέρεται το υπολογιστικό φόρτο που θα έχει η κάθε διεργασία.

P0	**P1…P(p-1)**	**Κοινό υπολογιστικό φόρτο**
1) Διαβάζει το «N» (μέγεθος της ακολουθίας). 2) Διαβάζει την «Seq» (ένα ένα τα στοιχεία της ακολουθίας). 3) Στέλνει στις υπόλοιπες διεργασίες το «N». 4) Υπολογίζει το πόσα στοιχεία «n» πρέπει να πάρει η κάθε διεργασία, ώστε να γίνει ορθά η ισοκατανομή των δεδομένων. 5) Μοιράζει τα στοιχεία ισοκατανεμημένα στις υπόλοιπες διεργασίες ξεκινώντας από την θέση «n» της ακολουθίας, προκειμένου και η P0 να πάρει τα δικά της στοιχεία που θα είναι τα πρώτα «n» της ακολουθίας. 6) Παίρνει τα δικά της στοιχεία. 7) Στέλνει το τελευταίο στοιχείο της στην αμέσως επόμενη διεργασία (P1). 8) Ελέγχει από τα δικά της στοιχεία αν εντόπισε στοιχείο που χαλάει την ταξινόμηση και το αποθηκεύει, καθώς είναι και το πρώτο όλης της ακολουθίας.	1) Λαμβάνουν από την P0 το «N». 2) Υπολογίζουν με βάση το «N» που μόλις έλαβαν το πόσα στοιχεία της ακολουθίας αναμένουν να παραλάβουν από την «P0» (n). 3) Λαμβάνουν από την «P0» τα «n» στοιχεία τους. 4) Η «P1» λαμβάνει από την «P0» το τελευταίο της στοιχείο για έλεγχο με τα δικά της στοιχεία. Η ίδια ρουτίνα επαναλαμβάνεται και με τις άλλες διεργασίες (π.χ. η «P1» στέλνει στην «P2» το δικό της τελευταίο στοιχείο κλπ.) με εξαίρεση την τελευταία διεργασία την «P(p-1)» που λαμβάνει το τελευταίο στοιχείο της «P(p-2)», αλλά δεν	1) Ελέγχουν αν τα στοιχεία τους είναι ταξινομημένα κατά αύξουσα σειρά. 2) Εφόσον, βρουν στοιχείο που χαλάει την ταξινόμηση το αποθηκεύουν.

9) Λαμβάνει από τις υπόλοιπες διεργασίες τα μηνύματα για την ταξινόμηση των στοιχείων τους. 10) Εφοσόν, λάβει μήνυμα ότι υπάρχει στοιχείο που χαλάει την ταξινόμηση, λαμβάνει το ίδιο το στοιχείο από την διεργασία που το εντόπισε και το αποθηκεύει. 11) Τυπώνει με ένα μήνυμα την κατάσταση της ταξινόμησης όλης της ακολουθίας, καθώς και το πρώτο στοιχείο που χαλάει την ταξινόμηση, εφόσον, υπάρχει. 12) Διαβάζει την επιλογή του χρήστη από το μενού επιλογών. 13) Στέλνει την επιλογή και στις άλλες διεργασίες	στέλνει το δικό της κάπου. 5) Ελέγχουν αν το τελευταίο στοιχείο που έλαβαν από την αμέσως προηγούμενη διεργασία είναι μεγαλύτερο από το πρώτο τους στοιχείο 6) Στέλνουν στη «P0» ένα μήνυμα για την ταξινόμηση των στοιχείων τους. 7) Στην περίπτωση που εντόπισουν στοιχείο που χαλάει την ταξινόμηση το στέλνουν και αυτό στην «P0». 8) Λαμβάνουν από την «P0» την επιλογή του χρήστη από το μενού επιλογών.	

Το πρόβλημα και η υλοποίηση σε γλώσσα C

Για την υλοποίηση του ζητουμένου χρησιμοποιήθηκε το περιβάλλον του MPI που επιτυγχάνει παράλληλο υπολογισμό και η γλώσσα C. Οι MPI ρουτίνες που χρησιμοποιήθηκαν είναι οι MPI_Init, MPI_Comm_rank, MPI_Comm_size, MPI_Send, MPI_Recv και MPI_Finalize. Η επικοινωνία μεταξύ των διεργασιών είναι «point-to-point» και αναστέλουσσα. Το πρόγραμμα αναλύεται σε αυτά τα μεγάλα κεφάλαια :

Δήλωση μεταβλητών (Γραμμές 13-29)

Οι μεταβλητές περιγράφονται αναλυτικά στα σχόλια του «Check_Sort_Seq.c»

Έναρξη του MPI περιβάλλοντος (Γραμμές 31-33)

Η διαδικασία περιγράφεται αναλυτικά στα σχόλια του «Check_Sort_Seq.c»

Ατέρμονος βρόχος (Γραμμές 35-184)

Ο ατέρμονος βρόχος χρησιμοποιήθηκε για τον λόγο ότι η διαδικασία θα επαναλαμβάνεται μέχρι ο χρήστης να διαλέξει στο μενού επιλογών την επιλογή της εξόδου. Η «point-to-point» επικοινωνία επιτυγχάνεται με την χρήση των MPI_Send και MPI_Recv ρουτίνων. Οι ρουτίνες αυτές είναι αναστέλουσσες που σημαίνει ότι η διεργασία που καλεί την μία από αυτές μπλοκάρεται μέχρι να κληθεί η αντίστοιχη ρουτίνα από την άλλη διεργασία με την οποία υπάρχει επιθυμία για επικοινωνία. Το σημείο που ίσως φαίνεται περίπλοκο και θέλει μία τεκμηρίωση είναι το πώς αποθηκεύεται το πρώτο στοιχείο που χαλάει την ταξινόμηση, εφόσον, υπάρχει. Αρχικά, ανεξαρτήτως εύρεσης σημείου που χαλάει την ταξινόμηση, η διαδικασία δεν διακόπτεται, καθώς ο σκοπός είναι να βρεθεί το πρώτο. Όλες οι διεργασίες θα ελέγξουν τα στοιχεία τους και το τελευταίο στοιχείο

που ελάβαν από την αμέσως προηγούμενη διεργασία και εφόσον βρούν στοιχείο που χαλάει την ταξινόμηση των στοιχείων τους αυτές θα το στείλουν στην «P0». Η μεταβλητή «sort_breaker_exist» είναι η μεταβλητή ένδειξη ότι υπάρχει στοιχείο που χαλάει την ταξινόμηση και βοηθάει επίσης στην γραμμή 88 με την κλήση της «Check_Sort_Breaker» (με την βοήθεια ενός δείκτη που δείχνει στην μεταβλητή αυτή) να αποθηκευτεί το πρώτο στοιχείο από τις διεργασίες πλην της «P0» που χαλάει την ταξινόμηση των στοιχείων που κατέχει η μία διεργασία. Στις γραμμές 101-105 χρησιμοποιείται από τις διεργασίες P1…P(p-1) για την σύγκριση του πρώτου στοιχείου μίας διεργασίας με το τελευταίο στοιχείο της αμέσως προηγούμενης, ώστε να ελεγχθούν και τα ενδιάμεσα στοιχεία. Στις γραμμές 125-131 την χρησιμοποιεί η «P0» για τα δικά της στοιχεία που είχε βγάλει πόρισμα, καθώς αν εντοπίστηκε στοιχείο που χαλάει την ταξινόμηση είναι και το πρώτο όλης της ακολουθίας. Η «first_sort_breaker_stored» από το όνομα της είναι η μεταβλητή ένδειξη ότι το πρώτο στοιχείο που χαλάει την ταξινόμηση της ακολουθίας έχει αποθηκευτεί. Επομένως, οι μεταβλητές-ενδείξεις χρησιμεύουν ώστε να μπορέσω να κρατήσω το πρώτο στοιχείο που χαλάει την ταξινόμηση, καθώς οι διεργασίες, εφόσον, βρουν θα στέλνουν ορθά τα δικά τους στοιχεία που χαλάνε την ταξινόμηση της δικιάς τους ακολουθίας στην «P0». Η όλη διαδικασία περιγράφεται αναλυτικά στα σχόλια του «Check_Sort_Seq.c».

Μενού (Γραμμές 159-183)

Το μενού επιλογών περιλαμβάνει δύο επιλογές. 1) Συνέχεια της διαδικασίας, 2) Έξοδος του προγράμματος. Οποιαδήποτε άλλη επιλογή που πληκτρολογηθεί από τον χρήστη λαμβάνεται ως άκυρη και του ζητείται να εισάγει μία έγκυρη. Ο λόγος που η «P0» στέλνει την επιλογή του χρήστη και στις υπόλοιπες διεργασίες είναι για να καταλάβουν ότι αν πρόκειται για την επιλογή εξόδου θα πρέπει και αυτές να βγουν με μια «break» εντολή από τον ατέρμονο βρόχο (γραμμές 181-182), όπως βγαίνει η «P0» στις γραμμές 173-174. Έτσι, όλες οι διεργασίες θα καλέσουν την MPI_Finalize ρουτίνα στην γραμμή 186 και το πρόγραμμα θα τερματίσει ομαλά. Ο σκοπός ήταν να αποφύγω την εύκολη λύση του «exit (1)» που δεν θα επέτρεπε την κλήση της MPI_Finalize ρουτίνας και του ομαλού τερματισμού του προγράμματος, καθώς χωρίς την επιλογή του χρήστη κατανεμημένη σε όλες τις διεργασίες παρά μόνο στην «P0» δεν θα επέτρεπε στις άλλες διεργασίες τρόπο διαφυγής από τον ατέρμονο βρόχο. Η όλη διαδικασία περιγράφεται αναλυτικά στα σχόλια του «Check_Sort_Seq.c».

Check_MPI_Routines (Γραμμές 192-199)

Η συνάρτηση παίρνει σαν ορίσματα :
int return_value: Η τιμή που επιστρέφει η MPI ρουτίνα που την καλεί
char *mpi_routine: Το όνομα της MPI ρουτίνας που την καλεί

Η όλη διαδικασία περιγράφεται αναλυτικά στα σχόλια του «Check_Sort_Seq.c».

Check_Memory (Γραμμές 201-208)

Η συνάρτηση παίρνει σαν ορίσματα :
int *seq: Ο δυναμικά δεσμευμένος πίνακας που είναι αποθηκευμένη η ακολουθία της διεργασίας που την καλεί

Η όλη διαδικασία περιγράφεται αναλυτικά στα σχόλια του «Check_Sort_Seq.c».

Check_Sort_Break **(Γραμμές 211-228)**

Η συνάρτηση παίρνει σαν ορίσματα :
int *seq: Το τμήμα της ακολουθίας που κατέχει η διεργασία που την καλεί
int n: Το πλήθος των στοιχείων που κατέχει η διεργασία που την καλεί
int *ptr_sort_breaker: Δείκτης που δείχνει στην μεταβλητή «sort_breaker»
int *ptr_sort_breaker_exist: Δείκτης που δείχνει στην μεταβλητή «sot_breaker_exist»

Η όλη διαδικασία περιγράφεται αναλυτικά στα σχόλια του «Check_Sort_Seq.c».

Δυσκολίες

Η δυσκολία μου ήταν στο πρόβλημα με τα ενδιάμεσα στοιχεία, καθώς είχα σκεφτεί αρκετούς τρόπους που παραβίαζαν τις προδιαγραφές, οπότε και τις απέρριψα. Για αρχή σκεφτόμουν στην $2^{η}$ φάση να βγάλω από την διαδικασία την «P0» και το πρώτο στοιχείο της ακολουθίας και να ισοκατανεμεθούν τα N – 1 δεδομένα σε p – 1 επεξεργαστές ($1^{η}$ φάση τα N δεδομένα ισοκατανεμημένα σε p επεξεργαστές), ώστε να ελεγχθούν και τα ενδιάμεσα στοιχεία και να «ξεκουραστεί» κατά κάποιο τρόπο η «P0» που είναι φορτωμένη με άλλες δουλειές. Δεν γνώριζα κατά πόσο παραβίαζε τις προδιαγραφές αυτή η πρώτη μου σκέψη οπότε για σιγουριά κατέληξα στην μεθοδολογία να στέλνει η μία διεργασία στην άλλη το τελευταίο της στοιχείο και να το συγκρίνουν με το πρώτο που κατέχουν ήδη από την ισοκατανομή. Το αποτέλεσμα είναι ορθό και από το κεφάλαιο «Το υπολογιστικό φόρτο της κάθε διεργασίας» παρατηρείται ότι και ο υπολογιστικός φόρτος είναι περίπου και αυτός ισοκατανεμημένος.

Ενδεικτικά Τρεξίματα

Μεταγλώττιση: mpicc -o Check_Sort_Seq Check_Sort_Seq.c mpicc -o mpi mpi.c

Παράδειγμα 1 (mpirun -np 4 ./Check_Sort_Seq) mpirun -np 4 ./mpi

Number of processors are 4

Size of integers' sequence must be integer multiple of number of processors (N mod processors == 0).

Input the size of integers' sequence : 8

Input the integers' sequence

Seq[0] : 1

Seq[1] : 2

Seq[2] : 3

Seq[3] : 4

Seq[4] : 5

Seq[5] : 7

Seq[6] : 6

Seq[7] : 8

---------- P0 ----------

Is sequence sorted? no

Which is the 1st sort breaker? 7

[1] Continue...

[2] Exit...

Input a choice :

Παράδειγμα 2 (mpirun -np 4 ./Check_Sort_Seq) mpirun -np 4 ./mpi

Number of processors are 4

Size of integers' sequence must be integer multiple of number of processors (N mod processors == 0).

Input the size of integers' sequence : 24

Input the integers' sequence

Seq[0] : 1

Seq[1] : 2

Seq[2] : 3

Seq[3] : 4

Seq[4] : 5

Seq[5] : 6

Seq[6] : 7

Seq[7] : 12

Seq[8] : 34

Seq[9] : 56

Seq[10] : 55

Seq[11] : 23

Seq[12] : 111

Seq[13] : -9

Seq[14] : 0

Seq[15] : 123

Seq[16] : 45

Seq[17] : 56

Seq[18] : 7

Seq[19] : 8

Seq[20] : 1

Seq[21] : 3

Seq[22] : 45

Seq[23] : 24

---------- P0 ----------

Is sequence sorted? no

Which is the 1st sort breaker? 56

[1] Continue...

[2] Exit...

Input a choice :

Παράδειγμα 3 (mpirun -np 25 ./Check_Sort_Seq) mpirun -np 25 ./mpi

Number of processors are 25

Size of integers' sequence must be integer multiple of number of processors (N mod processors == 0).

Input the size of integers' sequence : 50

Input the integers' sequence

Seq[0] : 1

Seq[1] : 2

Seq[2] : 3

Seq[3] : 4

Seq[4] : 5

Seq[5] : 6

Seq[6] : 7

Seq[7] : 8

Seq[8] : 9

Seq[9] : 10

Seq[10] : 11

Seq[11] : 12

Seq[12] : 13

Seq[13] : 14

Seq[14] : 15

Seq[15] : 16

Seq[16] : 17

Seq[17] : 18

Seq[18] : 29

Seq[19] : 45

Seq[20] : 21

Seq[21] : 32

Seq[22] : 31

Seq[23] : 44

Seq[24] : 2

Seq[25] : 0

Seq[26] : 13

Seq[27] : 46

Seq[28] : 7

Seq[29] : 4

Seq[30] : 35

Seq[31] : 67

Seq[32] : 8

Seq[33] : 4

Seq[34] : 68

Seq[35] : 4

Seq[36] : 8

Seq[37] : 4

Seq[38] : 0

Seq[39] : 2

Seq[40] : -3

Seq[41] : -56

Seq[42] : 3

Seq[43] : 45

Seq[44] : -7

Seq[45] : 100

Seq[46] : 99

Seq[47] : 1

Seq[48] : 2

Seq[49] : 3

---------- P0 ----------

Is sequence sorted? no

Which is the 1st sort breaker? 45

[1] Continue...

[2] Exit...
Input a choice :

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΑΡΑΛΛΗΛΟ ΥΠΟΛΟΓΙΣΜΟ

Σας ευχαριστώ για την προσοχή σας.