# ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ

# ΕΞΕΤΑΣΕΙΣ ΦΕΒΡΟΥΑΡΙΟΥ 2014

**ΘΕΜΑ 1ο**

**α)** Δώστε τη διάταξη σωλήνωσης (pipeline) p επεξεργαστών που υπολογίζει την ακόλουθη προσέγγιση του συνημίτονου.

$$\sin\theta \approx \sum_{k=0}^{n} \frac{(-1)^k \theta^{2\kappa+1}}{(2\kappa+1)!} = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \ldots + \frac{(-1)^n \theta^{2n+1}}{(2n+1)!}$$

Η παράμετρος θ θα δίνεται ως είσοδο στο πρόβλημα ενώ η παράμετρος n είναι σταθερά και γνωστή εκ των προτέρων σε όλους τους επεξεργαστές. Επίσης αρχικά θεωρείστε ότι p=n. Θα πρέπει να δώσετε το διάγραμμα της διάταξης σωλήνωσης καθώς και τον ψευδοκώδικα σε κάθε επεξεργαστή. Πρέπει επίσης να τονισθεί ότι αρχικά κάθε επεξεργαστής «αγνοεί» τη θέση του μέσα στη διάταξη σωλήνωσης, δηλ. μαθαίνει την εργασία που θα εκτελέσει από την πληροφορία που έρχεται από τον προηγούμενο στη διάταξη επεξεργαστή.

**β)** Αν $T_{comp}$ και $T_{comm}$ είναι ο χρόνος μίας αριθμητικής πράξης και της μετάδοσης ενός αριθμητικού δεδομένου αντίστοιχα, προσδιορίστε το χρόνο εκτέλεσης της παραπάνω διάταξης για τον υπολογισμό του συνημίτονου για μία συγκεκριμένη τιμή θ.

**γ)** Ποιος είναι ο συνολικός χρόνος εκτέλεσης όταν θα πρέπει να υπολογισθεί το συνημίτονο για m διαφορετικές τιμές της παραμέτρου θ;

**δ)** Πως τροποποιείται η οργάνωση των υπολογισμών όταν n≥p και n mod p=0. Δώστε το συνολικό χρόνο εκτέλεσης του ερωτήματος γ) στην περίπτωση αυτή.

**ΘΕΜΑ 2ο**

**α)** Δώστε αναλυτικά τα βήματα του αλγόριθμου Odd-Even (Trasposition) sort για την ταξινόμηση της ακόλουθης λίστας n(=8) ακεραίων κατά αύξουσα σειρά:

13, 7, 9, 5, 11, 3, 12, 10

Θεωρείστε ότι για το πλήθος p των επεξεργαστών ισχύει p=n.

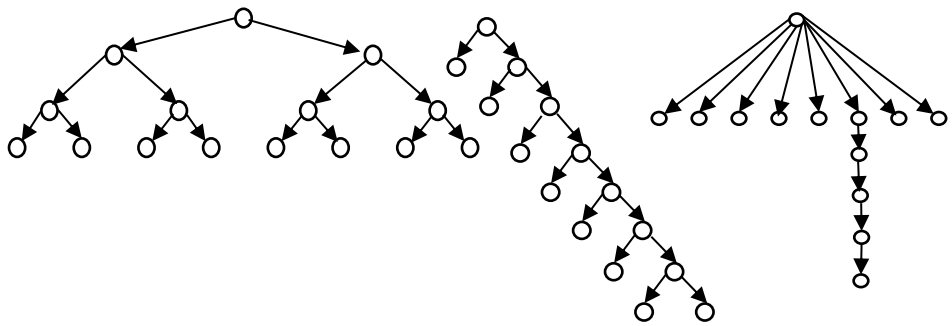
**β)** Δώστε τα βήματα του αλγορίθμου όταν p=4.

**γ)** Τροποποιώντας κατάλληλα τον αλγόριθμο Quicksort, δώστε ένα παράλληλο αλγόριθμο διαμοιραζόμενης μνήμης (shared memory) που θα υπολογίζει το κ-οστο μικρότερο στοιχείο σε ένα πίνακα n στοιχείων. Θεωρείστε ότι έχετε στη διάθεση σας p επεξεργαστές με n≥p.

**ΘΕΜΑ 3ο**

**α)** Σχεδιάστε ένα αλγόριθμο κατανεμημένης μνήμης (distributed memory) με τη βιβλιοθήκη του MPI που θα δέχεται ως είσοδο ένα πίνακα A διαστάσεων nxn και ένα διάνυσμα b διαστάσεων nx1 και θα υπολογίζει το γινόμενο Ab. Θεωρείστε ότι έχετε στη διάθεσή σας p διεργασίες/επεξεργαστές (n≥p). Μία από τις διεργασίες θα αναλάβει να διαβάσει τον πίνακα και το διάνυσμα από τον χρήστη και να μοιράσει κατάλληλα τα δεδομένα στις υπόλοιπες διεργασίες. Με την ολοκλήρωση του υπολογισμού η ίδια διεργασία θα τυπώνει το αποτέλεσμα στο χρήστη.

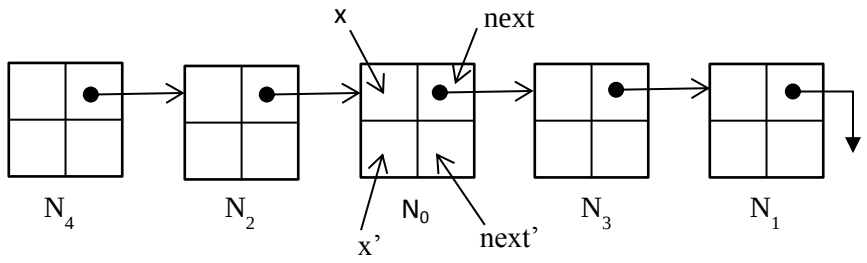
**β)** Για κάθε ένα από τα ακόλουθα γραφήματα εξαρτήσεων έργων (task dependency graph), δώστε τη μέγιστη επιτάχυνση (speedup) που μπορεί να επιτευχθεί όταν έχουμε στη διάθεση μας απεριόριστο πλήθος επεξεργαστών. Ποια είναι η επιτάχυνση όταν έχουμε 4 επεξεργαστές στη διάθεσή μας; Υποθέστε ότι κάθε έργο έχει τον ίδιο χρόνο επεξεργασίας και ότι ο χρόνος επικοινωνίας μεταξύ των επεξεργαστών είναι αμελητέος. Σχετικά με τα διαγράμματα εξαρτήσεων εργασιών, υπενθυμίζεται ότι κάθε κόμβος υποδηλώνει ένα έργο, ενώ ένα τόξο από το έργο A στο έργο B δηλώνει ότι το B εξαρτάται από το αποτέλεσμα του έργου A.



**Θέμα 4$^{ο}$**

**α)** Δίνονται δύο ακολουθίες ακεραίων $\{\alpha_1, \alpha_2,\ldots, \alpha_n\}$ και $\{\beta_1, \beta_2,\ldots, \beta_n\}$. Σε πολυεπεξεργαστικό περιβάλλον διαμοιραζόμενης μνήμης, σχεδιάστε ένα παράλληλο αλγόριθμο που θα υπολογίζει την ακολουθία $\{s_1, s_2,\ldots, s_n\}$ όπου $s_i=\alpha_i s_{i-1}+\beta_i$ $(i=1,\ldots,n)$ και $s_0=1$. Θεωρείστε ότι έχετε στη διάθεσή σας p(=n) επεξεργαστές.

**β)** Δίνεται μία μονά διασυνδεδεμένη λίστα n στοιχείων σε διαμοιραζόμενη μνήμη. Θεωρείστε ότι για το πλήθος των διεργασιών p ισχύει p=n. Κάθε διεργασία αναλαμβάνει ένα τυχαίο κόμβο της λίστας και έστω $N_i$ ο κόμβος που έχει αναλάβει η διεργασία $P_i$ (i=0,…,p-1). Στο παράδειγμα που ακολουθεί p=n=5. Κάθε κόμβος έχει τέσσερα πεδία. Το πεδίο x αποθηκεύει έναν ακέραιο, το πεδίο next δείχνει στον επόμενο κόμβο της λίστας, ενώ τα πεδία x' και next' είναι βοηθητικά και έχουν τον ίδιο τύπο με τα στοιχεία x και next αντίστοιχα.




Περιγράψτε τη λειτουργία του ψευδοκώδικα (α). Τι αποτέλεσμα δίνει; Πόσες επαναλήψεις γίνονται στο while loop στη χειρότερη περίπτωση; Ο αλγόριθμος θα μπορούσε να υλοποιηθεί πιο απλά όπως περιγράφεται στο ψευδοκώδικα (β);

```
While there exists i such that Ni->next  <> null do
        For each Pi  (i=0..p-1)
            Ni->x'= Ni-> x
            Ni->next'= Ni->next
            Barrier
            If  Ni->next' <> null
               Ni->x= Ni->x'+ Ni->next'->x'
               Ni->next= Ni->next'->next'
            endif
            Barrier
        End For each
end While
```

(α)

```
While there exists i such that Ni->next <> null do
        For each Pi  (i=0..p-1)
            If  Ni->next <> null
               Ni->x= Ni->x+ Ni->next->x
               Ni->next= Ni->next->next
            endif
            Barrier
        End For each
end While
```

(β)