ΗΥ404: ΠΑΡΑΛΛΗΛΟΣ ΚΑΙ ΔΙΚΤΥΑΚΟΣ ΥΠΟΛΟΓΙΣΜΟΣ
Χειμερινό Εξάμηνο 2018-2019

Εργασία 1

Ομάδα φοιτητών:
Φωτόπουλος Αποστόλος – 2202
Χατζηευφραιμίδης Λευτέρης - 2209

Χαρακτηριστικά Υπολογιστή

Επεξεργαστής : Intel(R) Core(TM) i5-6600K CPU @ 3.50Ghz (Quad core) .

Ram: 16 GB.

Λειτουργικό : Ubuntu 16.04 (x64)

Ερώτηση 1

Τα βασικά στάδια για το σχεδιασμό ενός παράλληλου αλγορίθμου είναι :

- Τεμαχισμός : διαμερισμός του προβλήματος σε μικρότερες εργασίες που εκτελούνται ταυτόχρονα.
- Επικοινωνία : ορισμός επικοινωνίας μεταξύ επιμέρους εργασιών.
- Συνχώνευση : συνχώνευση μικρών επιμέρους-εργασιών σε μία εργασία, με σκοπό τη μείωση επικοινωνιακού κόστους.
- Αντιστοίχιση: αντιστοίχιση εργασιών σε επξεργαστές, με έμφαση στην ταυτόχρονη εργασία και στο ελάχιστο επικοινωνιακό κόστος.

Σε αυτά τα στάδια θα πρέπει να προσέξουμε τα εξής :

Στον τεμαχισμό :

1. Το πλήθος των επιμέρους διεργασιών θα πρέπει να είναι τουλάχιστον ίσο με το πλήθος των επεξεργαστών στο σύστημα που θα χρησιμοποιήσουμε.
2. Αποφυγή άσκοπων υπολογισμών ή αποθήκευση στοιχείων.
3. Οι επιμέρους διεργασίες πρέπει να είναι όμοιου μεγέθους.
4. Με την αύξηση του μεγέθους του προβλήματος, θα πρέπει να αυξάνεται το πλήθος των επιμέρους διεργασιών.
5. Το μέγεθος των διεργασιών, δεν πρέπει να αυξάνεται με τον ιδιο βαθμό που αυξάνεται το όλο πρόβλημα.

Στην επικοινωνία :

1. Τον τύπο της επικοινωνίας (Τοπική ή γενική,δομημένη ή μη,στατική ή δυναμική,σύγχρονη ή ασύγχρονη).
2. Η επικοινωνία μεταξύ των επιμέρους διεργασιών θα πρέπει να είναι σύχρονη,όμοια σε συχνότητα και όγκο,όσο το δυνατόν τοπική καθώς και επικαλυπτόμενη από υπολογισμούς.
3. Η επικοινωνία δεν πρέπει να αναστέλλει τις σύγχρονες εκτελέσεις των επιμέρους διεργασιών.

Στην συνχώνευση :

1. Αυξάνοντας το μέγεθος των διεργασιών ελλατώνεται η επικονωνία αλλά ταυτόχρονα ελλατώνεται και η ευελιξία και ο βαθμός παραλληλισιμότητας.
2. Επιμέρους διεργασίες που δεν μπορούν να εκτελεσθούν ταυτόχρονα, πρέπει να συγχωνεύονται.
3. Μπορούμε να αποφύγουμε επιπλέον επικοινωνία με εκτέλεση των ίδιων υπολογισμών σε κάθε επεξεργαστή.
4. Ο όγκος της επικοινωνίας είναι ανάλογος με το μέγεθος της διεπιφάνειας, ενώ το πλήθος των υπολογισμών είναι ανάλογο με τον όγκο του υποχωρίου.
5. Οι διαχωρισμοί μεγαλύτερης διάστασης ακολουθούν τον προηγούμενο νόμο περισσότερο από τους διαμερισμούς μικρότερης διάστασης.

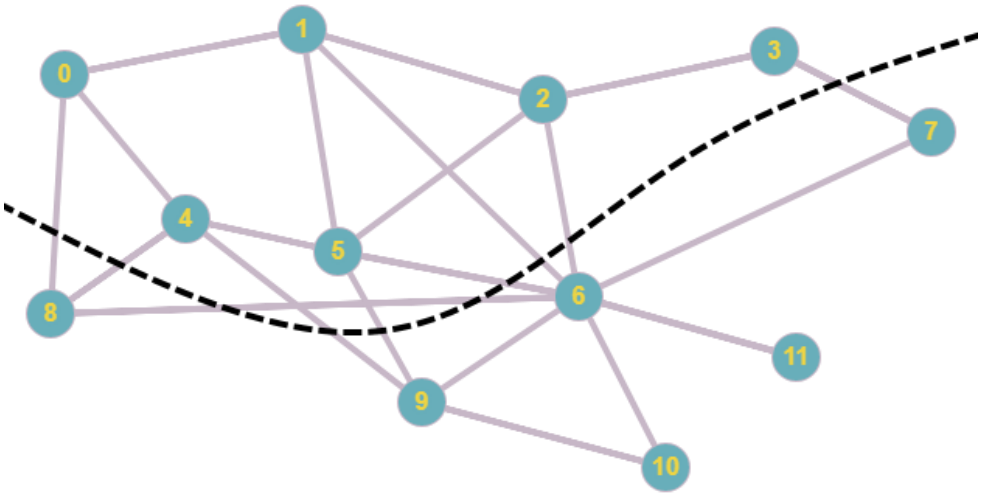
Στην ανάθεση :

1. Υπάρχουν δύο βασικές στρατηγικές για να αναθέτουμε διεργασίες σε επεξεργαστές:
    - Ανάθεση διεργασιών που μπορούν να εκτελεσθούν ταυτόχρονα σε διαφορετικούς(πολυ)επεξεργαστές.
    - Ανάθεση διεργασιών που επικοινωνούν συχνά στον ίδιο(πολυ)επεξεργαστή.
2. Οι δύο στρατηγικές συγκρούονται. Η βέλτιστη λύση ανάθεσης διεργασιών σε επεξεργαστές είναι NP-complete πρόβλημα διάφορες μέθοδοι χρησιμοποιούνται για να συμβιβαστούν οι δύο στρατηγικές.
3. Οι αναθέσεις διεργασιών μπορούν να γίνουν με στατικό τρόπο(static) ή με δυναμικό τρόπο(dynamic).

Ερώτηση 2

Διαχωρισμός σε block γραμμών

Α) Όταν έχουμε δύο επεξεργαστές ο 1ος επεξεργαστής C_0 θα έχει τις γραμμές 0-5 ενώ ο 2ος επεξεργαστής C_1 θα έχει από 6-11.



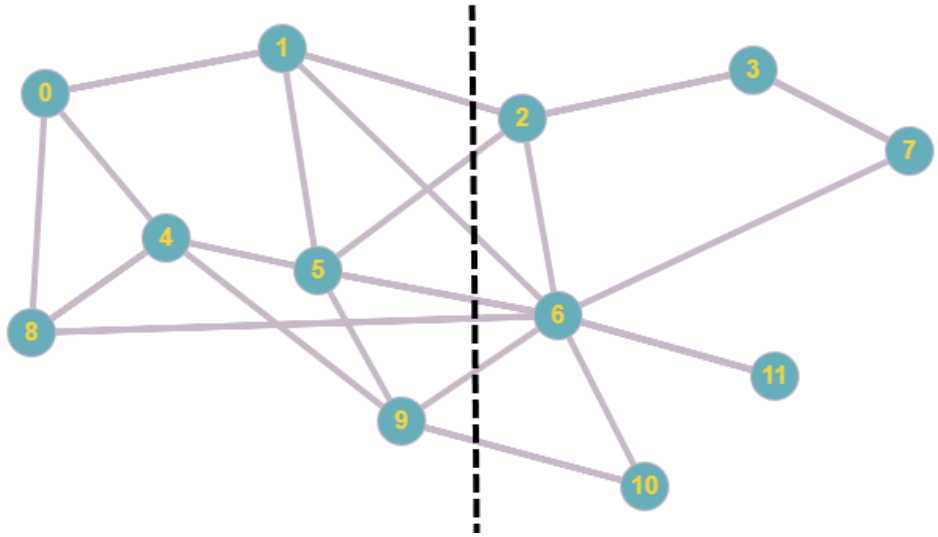
Εικόνα 1. *Διάγραμμα για δύο επεξεργαστές.*

Επομένως από το σχήμα βγάζουμε και τις **εξαρτήσεις** :

C_0 = (6, 7, 8, 9)

C_1 = (0, 1, 2, 3, 4, 5)

Ένας άλλος τρόπος για τον καταμερισμό του προβλήματος με σκοπό την μείωση της επικοινωνίας είναι ο εξής :



Εικόνα 2. *Βελιωμένο διάγραμμα για δύο επεξεργαστές.*

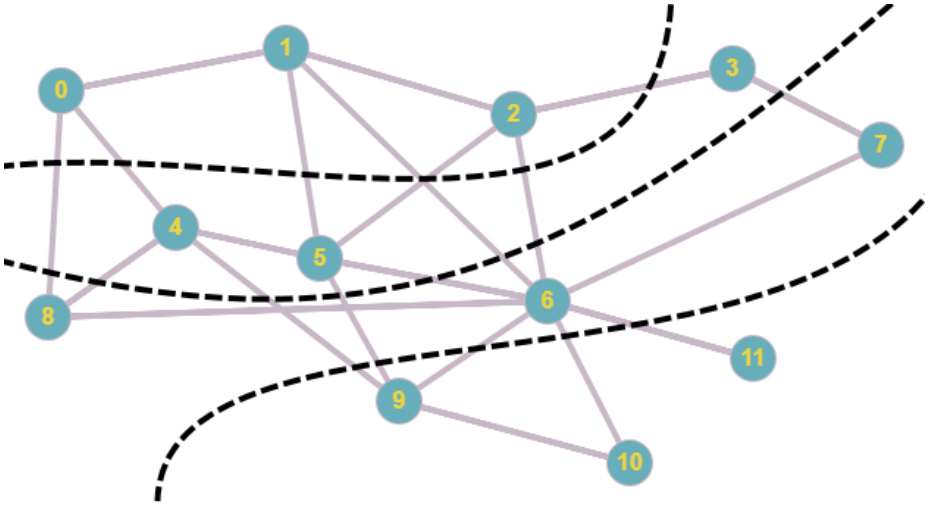
Σε αυτή την περίπτωση το C_0 περιέχει τα 0,1,4,5,8,9 ενώ το C_1 τα 2,6,3,7,10,11.

Με αυτόν τον τρόπο οι **εξαρτήσεις** γίνονται :

C_0 = (2, 6, 10)

C_1 = (1, 5, 8, 9)

Β)) Όμοια για τέσσερις επεξεργαστές προκύπτει το παρακάτω σχήμα.



Εικόνα 3. *Διάγραμμα για τέσσερις επεξεργαστές.*

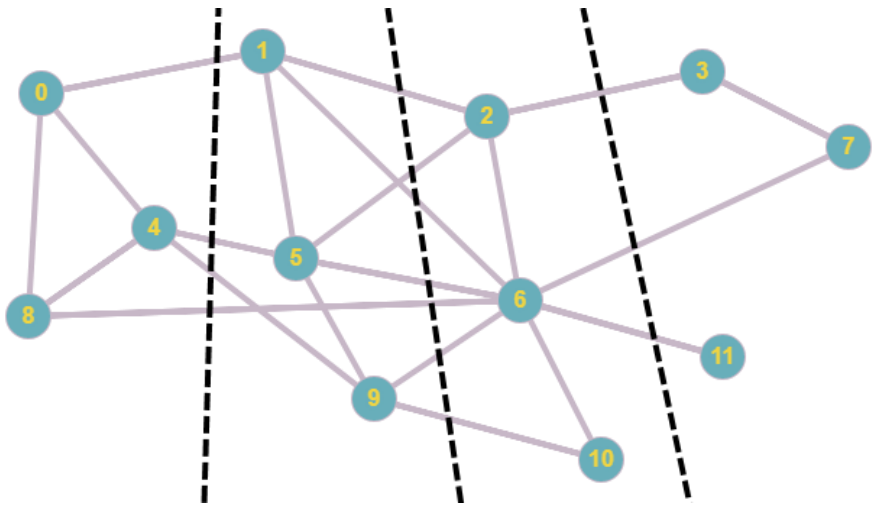
Το C_0 περιέχει τα 0,1,2 , το C_1 τα 3,4,5, το C_2 τα 6,7,8 και το C_3 τα 9,10,11.

Επομένως οι **εξαρτήσεις** διαμορφώνονται ως εξής :

C_0 = (3, 4, 5, 6, 8) C_1 = (0, 1, 2, 6, 7, 8, 9)

C_2 = (0, 2, 3, 4, 5, 9, 10, 11) C_3 = (4, 5, 6)

Ένας άλλος τρόπος για τον καταμερισμό του προβλήματος με σκοπό την μείωση της επικοινωνίας είναι ο εξής :



Εικόνα 4. *Βελτιωμένο διάγραμμα για τέσσερις επεξεργαστές.*

Το C_0 περιέχει τα 0,4,8 , το C_1 τα 1,5,9, το C_2 τα 2,6,10 και το C_4 τα 3,7,11.

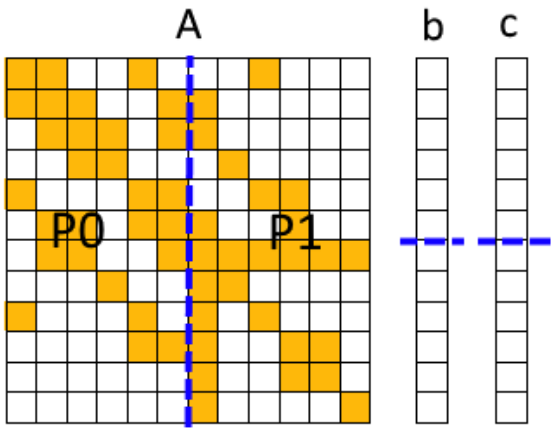
Επομένως οι **εξαρτήσεις** διαμορφώνονται ως εξής :

$C_0 = (1, 5, 6, 9)$ $C_1 = (0, 2, 4, 6, 10)$

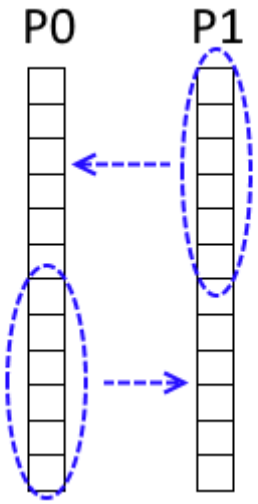
$C_2 = (1, 3, 5, 7, 9, 11)$ $C_3 = (2, 6)$

Διαχωρισμός σε block στηλών

Στην περίπτωση των **δύο** επεξεργαστών ο πίνακας διαχωρίζεται σε δύο block στηλών.Αρχικά ο κάθε επεξεργαστής υπολογίζει τα μερικά αθροίσματα που του αναλογούν.Για παράδειγμα ο P0 πολλαπλασιάζει το block στηλών του με το αντίστοιχο μέρος του b που του αναλογεί.Στη συνέχεια ο P0 στέλνει στον P1 τα μερικά αθροίσματα που χρειάζεται και που **δεν είναι μηδενικά** για να υπολογίσει το μέρος του c που του αντιστοιχεί .Ο P1 ακολουθεί την ίδια διαδικασία.Οι εξαρτήσεις είναι ίδιες με την περίπτωση διαχωρισμού γραμμών λόγω της συμμετρικότητας του πίνακα.

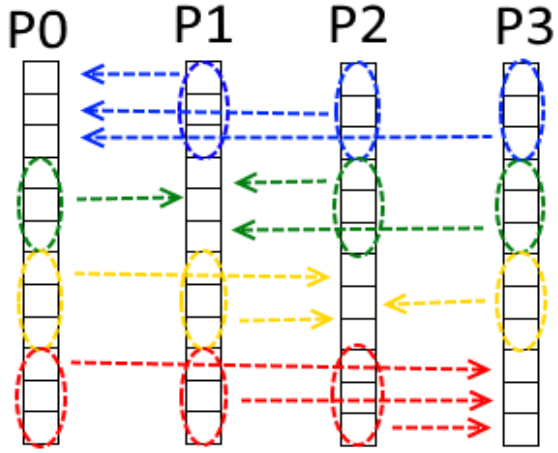


Εικόνα 5.*Διαχωρισμός δεδομένων για 2 επεξεργαστές.*



Εικόνα 6.*Ανταλλαγή μερικών αθροισμάτων μεταξύ δύο επεξεργαστών.*

Στην περίπτωση των **τεσσάρων** επεξεργαστών ακολουθείται η ίδια λογική ,ο πίνακας διαχωρίζεται σε τέσσερα block και ο κάθε επεξεργαστής λαμβάνει από τους υπόλοιπους τα μερικά αθροίσματα που χρειάζεται για να υπολογίσει το μέρος του c που του αναλογεί.

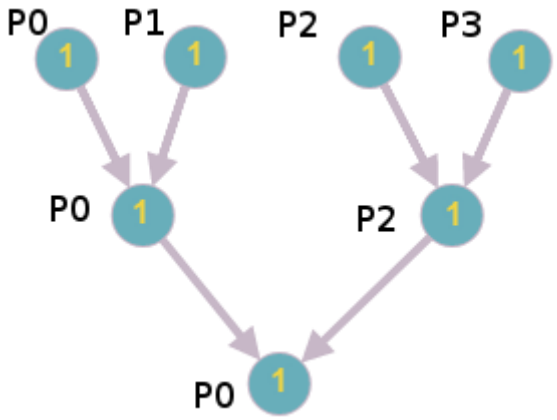


Εικόνα 6.*Ανταλλαγή μερικών αθροισμάτων μεταξύ τεσσάρων επεξεργαστών.*

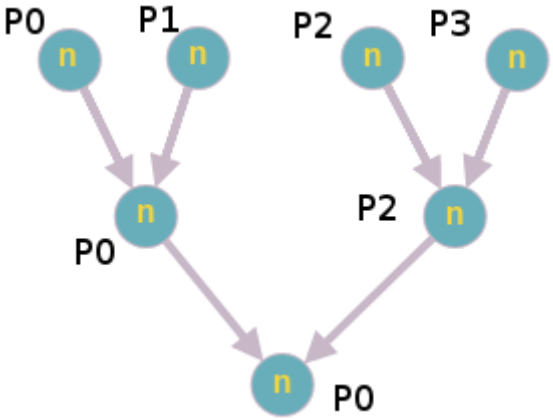
Ερώτηση 3

Στην πρώτη περίπτωση έχουμε n διεργασίες αφού έχουμε n πολλαπλασιασμούς για τα στοιχεία των δύο διανυσμάτων. Άρα μπορούνε να τρέξουνε παράλληλα n διεργασίες για τον υπολογισμό των γινομένων και στην συνέχεια να επικοινωνήσουν μεταξύ τους για να γίνει το ολικό άθροισμα. Η επικοινωνία γίνεται όπως στην εικόνα 7.

Στην δεύτερη περίπτωση ο διαχωρισμός γίνεται ανά στήλη δηλαδή η κάθε διεργασία υπολογίζει το γινόμενο μιας στήλης του πίνακα με το αντίστοιχο στοιχείο του διανύσματος. Στην συνέχεια επικοινωνούν μεταξύ τους και υπολογίζουν το νέο διάνυσμα. Ο τρόπος επικοινωνίας είναι ίδιος αλλά αντί για έναν πολλαπλασιασμό ο κάθε επεξεργαστής κάνει n πολλαπλασιασμούς. Το σχήμα είναι όμοιο με την πρώτη περίπτωση.



Εικόνα 7. *Διάγραμμα με κόστη για την 1η περίπτωση.*



Εικόνα 8 *.Διάγραμμα με κόστη για την 2η περίπτωση.*

Στο εσωτερικό των κόμβων υπάρχουν το σύνολο των πράξεων(κόστος) για κάθε διεργασία.

Και στις δύο περιπτώσεις εκτελούνται παράλληλα n διεργασίες άρα ο Μ.β.π = n.

Το μ.β.π και στις δύο περιπτώσεις είναι $\frac{((2n-1)*κόστος\ κόμβου)}{\lceil log2(n)+1\rceil*κόστος\ κόμβου}$. Το 2n-1 προκύπτει επειδή έχουμε n πολλαπλασιασμούς και n-1 προσθέσεις για τον υπολογισμό του γινομένου και το κόστος του κόμβου από τον αριθμό των πράξεων που έχει να πραγματοποιήσει. Ο παρονομαστής υπολογίζει το κόστος του κρίσιμου μονοπατιού. Το **ceil** το χρησιμοποιούμε για να μας δοθεί ακέραιος αριθμός μονοπατιών. Για παράδειγμα αν έχουμε n=4 στην πρώτη περίπτωση : $\frac{((2n-1)*1)}{\lceil log2(n)+1\rceil*1}$ = 7/3. Με όμοιο τρόποι και για την δεύτερη περίπτωση προκύπτει το ίδιο αποτέλεσμα.

Ο ψευδοκώδικας για το εσωτερικό γινόμενο των δύο διανυσμάτων :

```
level = [0,1,.....,N-1]  //οι διεργασίες που τρέχουν σε κάθε βήμα
removeElements=[]  //στοιχεία που πρέπει να διαγραφούν σε κάθε βήμα
id  //το ID του κάθε thread

// Ο καθε επεξεργαστής γνωρίζει τα a,b που του αντιστοιχούν
data = a_id * b_id
while( level not includes only one element ) {
    if( id is included in level){
        pos=getPositionInLevel()
        if( id == last_element_of_level and count_of_elements_is_odd() ){
            wait_till_next_level()
        }
        else if( pos mod 2 != 0 ){
            send(data,pos-1)
            add_element_to_removeElements()
        }
        else{
            receive(recData,pos+1)
            data=data+recData
        }
        wait_for_threads_to_complete()
        updateLevel()
    }
}
print data_from_P0
```

Ο ψευδοκώδικας για το εσωτερικό γινόμενο του πίνακα με το διάνυσμα :

```
level = [0,1,.....,N-1]  //οι διεργασίες που τρέχουν σε κάθε βήμα
removeElements=[]  //στοιχεία που πρέπει να διαγραφούν σε κάθε βήμα
id  //το ID του κάθε thread

// Ο καθε επεξεργαστής γνωρίζει τα a,b που του αντιστοιχούν
for(i =0 , i< N-1 ,i=i+1){
    data[i] = a_id[i]*b_id
}
while( level not includes only one element ) {
    if( id is included in level){
        pos=getPositionInLevel()
        if( id == last_element_of_level and count_of_elements_is_odd() ){
            wait_till_next_level()
        }
        else if( pos mod 2 != 0 ){
            send(data,pos-1)
            add_element_to_removeElements()
        }
        else{
            receive(recData,pos+1)
            data=data+recData
        }
        wait_for_threads_to_complete()
        updateLevel()
    }
}
print data_from_P0
```

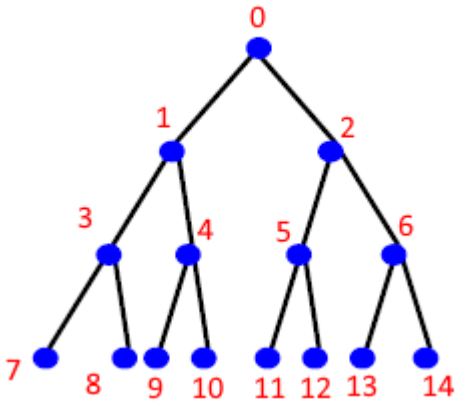
Ερώτηση 4

Οι βασικές ιδιότητες της ένθεσης δύο γραφημάτων είναι οι εξής :

- Συμφόρηση : μέγιστο πλήθος ακμών του Ε, που αντιστοιχούν σε μια ακμή του Ε'.
- Διαστολή: μέγιστο πλήθος ακμών/συνδέσεων στο Ε' για την αντιστοίχηση μιας ακμής του Ε.
- Επέκταση: ο λόγος του πλήθους των κορυφών του V' ως προς το πλήθος των κορυφών του V.

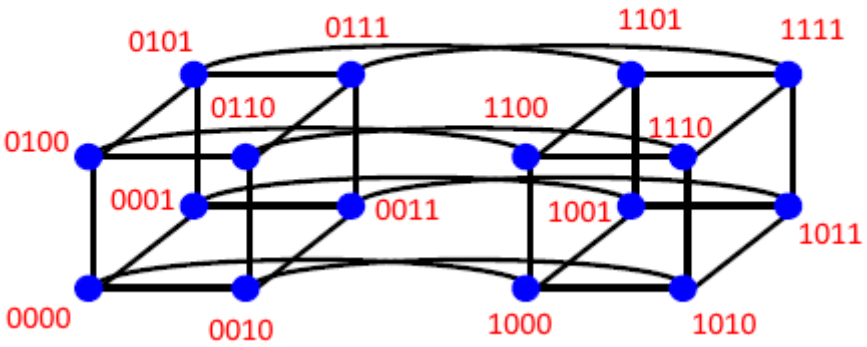
Ερώτηση 5

Καταρχάς αριθμίζουμε το δυαδικό δέντρο.



Εικόνα 9. *Αρίθμηση δυαδικού δέντρου.*

Στην συνέχεια αριθμίζουμε τον κύβο με τον εξής τρόπο :

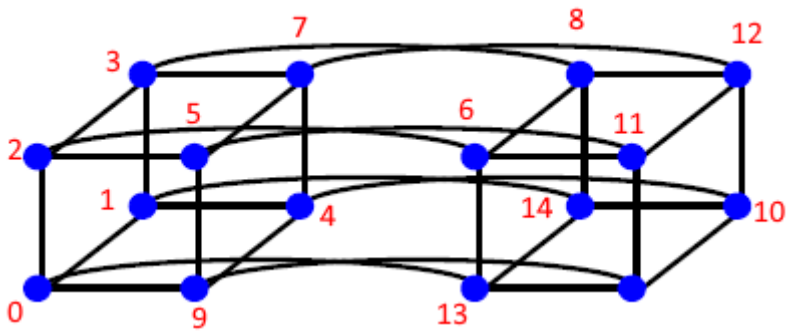


Εικόνα 10. *Αρίθμηση υπερκύβου.*

Η αντιστοίχηση μεταξύ των κόμβων των δύο γράφων είναι η εξής :

Binary Tree	4-D cube
0000	0000
0001	0001
0010	0100
0011	0101
0100	0011
0101	0110
0110	1100
0111	0111
1000	1101
1001	0010
1010	1011
1011	1010
1100	1111
1101	1000
1110	1001

Το σχήμα που προκύπτει από την παραπάνω αντιστοίχηση των κόμβων :



Εικόνα 11. *Αντιστοιχία κόμβων.*

Ακολουθώντας την παραπάνω αντιστοίχηση των κόμβων οι ιδιότητες της ένθεσης διαμορφώνονται ως εξής :

- Συμφόρηση(congestion) = 1
- Διαστολή(dilation) = 2 λόγω της διασύνδεσης των κορυφών 5 -> 12 και 6 -> 14
- Επέκταση(expansion) = 1

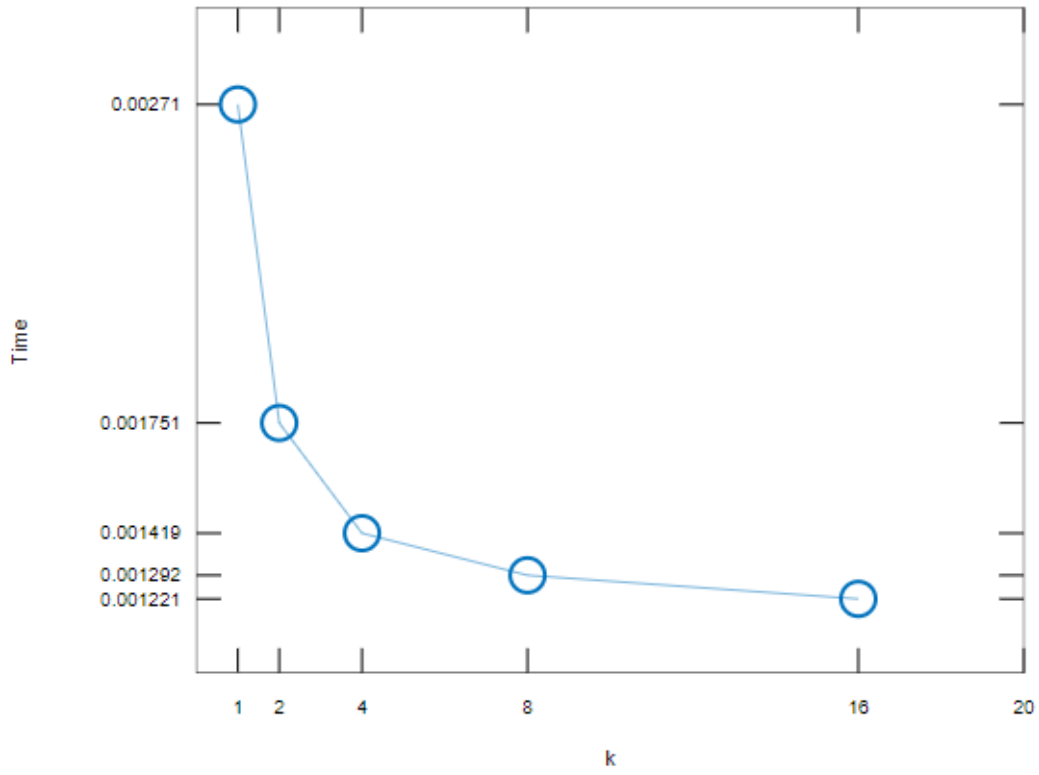
Επομένως όταν μας δωθεί ένα αντιστοίχο δέντρο με 15 κόμβους μπορούμε να το μετασχηματίσουμε σε ένα 4-D υπερκύβο ακολουθώντας την παραπάνω διαδικασία.

Ερώτηση 6

Τρέχοντας το πρόγραμμα askhsh6.c και λαμβάνοντας υπόψη το μέσο όρο για 4 τρεξίματα στο κάθε βήμα k καταλήξαμε στις παρακάτω μετρήσεις:

Βήμα	Χρόνος(sec)	Flops
1	0.002710	774019171.8
2	0.001751	1198821087.9
4	0.001419	1478552136.6
8	0.001292	1622822573.6
16	0.001221	1719367875.1

Η αντίστοιχη γραφική των χρόνων με τα βήματα είναι η εξής :



Το πλήθος των πράξεων όπως βλέπουμε και στο αρχείο εξόδου είναι 2N που προκύπτει από N-1 προσθέσεις, N πολλαπλασιασμούς και εξαιτίας του ότι το αποθηκευέουμε σε μια local μεταβλητή έχουμε αλλή μια πρόσθεση.

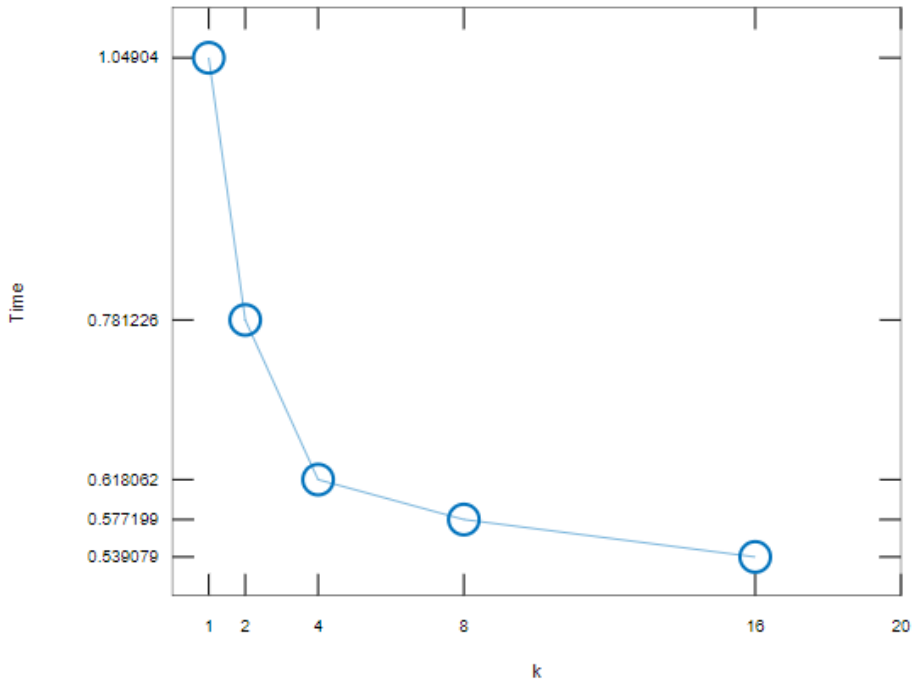
Η μέγιστη ταχύτητα των FLOPS παρατηρείται για k=16 και είναι 1719367875.1 πράξεις το δευτερόλεπτο.Δεν παρατήρησαμε κάποια άνοδο στον χρόνο με την αύξηση των βημάτων , επομένως δεν καταλήξαμε σε κάποιο κρίσιμο σημείο.Ο λόγος που βλέπουμε βελτίωση στο χρόνο οφείλεται στο γεγονός πως εκμεταλευόμαστε το prefetching που κάνει η cache και δεν χρείαζεται να τραβήξουμε δεδομένα από την μνήμη ούτε να εκτελέσουμε περισσότερα loops.

Ερώτηση 7

Τρέχοντας το πρόγραμμα askhsh7.c και λαμβάνοντας υπόψη το μέσο όρο για 4 τρεξίματα στο κάθε βήμα k καταλήξαμε στις παρακάτω μετρήσεις:

Βήμα	Χρόνος(sec)	Flops
1	1.049043	257453565.5
2	0.781226	343704654.3
4	0.618062	434318614.4
8	0.577199	465067307.8
16	0.539079	497958637.1

Η αντίστοιχη γραφική των χρόνων με τα βήματα είναι η εξής :



Το πλήθος των πράξεων όπως βλέπουμε και στο αρχείο εξόδου είναι $2N^3$ που προκύπτει από N-1 προσθέσεις και N πολλαπλασιασμούς για N^2 στοιχεία.

Η μέγιστη ταχύτητα των FLOPS παρατηρείται για k=16 και είναι 497958637.1 πράξεις το δευτερόλεπτο.Δεν παρατήρησαμε κάποια άνοδο στον χρόνο με την αύξηση των βημάτων , επομένως δεν καταλήξαμε σε κάποιο κρίσιμο σημείο.Ο λόγος που βλέπουμε βελτίωση στο χρόνο οφείλεται στο γεγονός πως εκμεταλευόμαστε το prefetching που κάνει η cache και δεν χρείαζεται να τραβήξουμε δεδομένα από την μνήμη ούτε να εκτελέσουμε περισσότερα loops.

Τα αρχεία MatrixMultiplyResult.txt και VectorsMultiplyResult.txt είναι οι μετρήσεις από τα τρεξίματα μας.