Παράλληλα και Διανεμημένα Συστήματα

Μάριος Πάκας

9498

# Εργασία 1

## Ανάλυση της δομής δεδομένων

Το πρώτο βήμα στην ανάπτυξη κώδικα για τη συγκεκριμένη εργασία ήταν η ανάγνωση Market Matrix. Επειδή στην αρχή δεν έβρισκα δυαδικούς πίνακες, μέσω του command line terminal υπάρχει επιλογή να δώσει ο χρήστης την τιμή 0 για μη δυαδικό πίνακα και 1 για δυαδικό πίνακα (default 1). Έτσι γνωρίζει το πρόγραμμα αν ο coo πίνακας έχει 3 ή 2 στοιχεία ανά γραμμή και με αυτόν τον τρόπο μπορεί να τα διαβάσει δίχως πρόβλημα μνήμης. Ωστόσο δεν δουλεύει για πίνακες με μη μηδενικά στοιχεία στη διαγώνιο.

Ακόμη, ήταν σημαντικό για το v3 να έχω τους πίνακες σε άνω τριγωνική μορφή, οπότε ήθελα να συγκρίνω την στήλη και τη γραμμή του συμμετρικού πίνακα ώστε, ανάλογα με το τι ήταν μεγαλύτερο, να το περάσω σωστά ως όρισμα στην συνάρτηση coo2csc και να βγει άνω τριγωνικός. Στο v4 με βόλεψε να δουλέψω με ολόκληρο τον πίνακα και όχι μόνο τον συμμετρικό, οπότε διπλασίαζα όλα τα στοιχεία.

## Ανάλυση αλγορίθμου V3

Για το V3, εφόσον έχουμε πλέον τον πίνακα σε δομή csc η σκέψη μου ήταν η εξής: Ξεκινούσα από ένα τυχαίο στοιχείο έστω row1, col1. Τότε γνωρίζω ότι:

(row1,col1) -> (col1,col2) -> (col2,row1)

Στη μορφή csc ο εύκολος τρόπος αναζήτησης ήταν ανά στήλη να βρίσκω ποια γραμμή είναι μη μηδενική, επομένως είχα στα χέρια μου το πρώτο στοιχείο (row1, col1). Στη συνέχεια (και επειδή, λόγω συμμετρίας του πίνακα, ίσχυε ότι στήλη = γραμμή και το αντίθετο) μπορούσα να ψάξω για το στήλη row1 και να βρω ποιες τιμές υπήρχαν ως γραμμές. Συνεπώς είχα πλέον στα χέρια μου και το στοιχείο (col2,row1). Προκειμένου να υπάρχει λοιπόν τρίγωνο έπρεπε να αναζητήσω αν στη στήλη row1 υπήρχε στοιχείο στη γραμμή με συντεταγμένη col1! Σε περίπτωση που υπήρχε αύξανα τον αριθμό τριγώνων και τον πίνακα c3 για τις αντίστοιχες κορυφές row1, col1, col2.

## Ανάλυση αλγορίθμου V4

Για τον αλγόριθμο στο V4 το πρώτο βήμα ήταν να υλοποιήσω τον πολλαπλασιασμό C=A⊙(AA). Στην αρχή έκανα ξεχωριστά τον κάθε πολλαπλασιασμό, αλλά στη συνέχεια το άλλαξα διότι ο A\*A έβγαινε πυκνός και ήταν πιο περίπλοκο στο χειρισμό της μνήμης. Η ιδέα πίσω από αυτόν τον πολλαπλασιασμό είναι ότι τα μόνα στοιχεία που θα υπάρχουν θα βρίσκονται στην ίδια ακριβώς θέση με τον αρχικό πίνακα Α. Επομένως ξεκινούσα περνώντας από κάθε στοιχείο (row, col). Στο επόμενο βήμα έπρεπε να υπολογίσω την τιμή (ΑΑ)[row,col]. Η τιμή αυτή ισούται με το άθροισμα των πολλαπλασιασμών ανάμεσα στις τιμές της γραμμής row και της στήλης col. Εκμεταλλευόμενοι την ιδιότητα της συμμετρίας όμως: γραμμή = στήλη. Επομένως κρατούσα σε έναν πίνακα k όλες τις συντεταγμένες των στηλών που υπήρχαν στην γραμμή row και σε έναν πίνακα l όλες τις συντεταγμένες των γραμμών στη στήλη col. Οι πίνακες αυτοί, λόγω της δομής csc ήταν και ταξινομημένοι, επομένως με ένα από merge sort σύγκρινα μεταξύ τους τις τιμές και εφόσον υπήρχε στήλη ίση με γραμμή σήμαινε ότι το (AA)[row,col] έπρεπε να αυξηθεί κατά 1 (αφού Α δυαδικός). Με αυτόν τον τρόπο έβρισκα τις τιμές του C, μιας και ισούταν με τις τιμές του (ΑΑ), απλώς μόνο για τις συντεταγμένες που ο Α είχε ήδη στοιχεία.

Το επόμενο βήμα ήταν να υλοποιήσω c3=C\*e όπου e ένα διάνυσμα μοναδιαίο. Υλοποιήθηκε με τη λογική ότι το i στοιχείο στο τελικό διάνυσμα είναι το άθροισμα της i-στης γραμμής (Hint: στήλης) με όλο το διάνυσμα c3. Ουσιαστικά το i-οστο στοιχείο του c3 ισούται με το άθροισμα όλων των στοιχείων που έχει ο C στην ι-στη γραμμή!

## Διαγράμματα που απεικονίζουν τους χρόνους εκτέλεσης κάθε προγράμματος