

«ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΠΟΛΥΠΛΟΚΟΤΗΤΑ»

ΕΡΓΑΣΙΑ

Α' ΜΕΡΟΣ (60%)

Υλοποίηση αλγορίθμων ταξινόμησης

Θα υλοποιήσετε τον αλγόριθμο ταξινόμησης σωρού, MergeSort και δύο παραλλαγές του βασικού αλγόριθμου, συγκεκριμένα τους αλγόριθμους TimSort και Adaptive ShiverSort. Οι δύο αυτές παραλλαγές βασίζονται στο γεγονός ότι ο πίνακας εισόδου ήδη μπορεί να περιέχει ταξινομημένα τμήματα κατά αύξουσα και κατά φθίνουσα σειρά και επομένως η διαδικασία των διαδοχικών συγχωνεύσεων (merge) δεν χρειάζεται να αρχίσει από στοιχειώδεις πίνακες ενός στοιχείου αλλά κατευθείαν από αυτά τα ταξινομημένα τμήματα του πίνακα εισόδου. Οι δύο παραλλαγές διαφέρουν στην στρατηγική που ακολουθούν για να καθορίσουν τη σειρά με την οποία θα συγχωνευτούν τα ταξινομημένα τμήματα. Για την υλοποίηση των αλγορίθμων TimeSort και ShiverSort, θα βασιστείτε αποκλειστικά στην περιγραφή που δίνεται στην ερευνητική εργασία [1].

Για τον έλεγχο της απόδοσης των αλγορίθμων θα χρησιμοποιήσετε δύο μετρητές compareCount και swapCount. Ο μετρητής compareCount θα αυξάνει κατά ένα κάθε φορά που γίνεται σύγκριση μεταξύ δύο αριθμών, ενώ ο μετρητής swapCount θα αυξάνει κάθε φορά που εναλλάσσονται δύο στοιχεία ή που ένα στοιχείο μετακινείται. Θα εκτελέσετε τους αλγορίθμους για διαφορετικό μέγεθος εισόδου. Συνεπώς, το πρόγραμμά σας, θα αποτελείται από έναν εξωτερικό βρόχο με τιμές στο διάστημα [50, 2000] με βήμα 50. Για κάθε μέγεθος εισόδου, θα γεμίζετε τον πίνακα με τυχαία στοιχεία και στη συνέχεια θα εκτελείτε τον αλγόριθμο. Θα επαναλάβετε δέκα φορές τη διαδικασία της τυχαίας επιλογής στοιχείων εισόδου και κατόπιν της εκτέλεσης του αλγορίθμου. Στο τέλος αυτών των δέκα επαναλήψεων για το ίδιο μέγεθος εισόδου, θα πρέπει να έχει υπολογιστεί επίσης η μέγιστη, η ελάχιστη και η μέση τιμή των μετρητών compareCount και swapCount.

Σχεδιάστε διαγράμματα που θα απεικονίζουν τη μεταβολή της ελάχιστης, μέγιστης και μέσης τιμής των μετρητών compareCount και swapCount συναρτήσει του μεγέθους εισόδου.

Δρομολόγηση Εργασιών

Έστω ένα σύνολο εργασιών και έστω μια μηχανή η οποία εκτελεί τις εργασίες. Κάθε εργασία i έχει ένα όφελος (βάρος), $w_i > 0$, χρόνο εκκίνησης s_i και χρόνο τερματισμού f_i . Ισχύει ότι η μηχανή μπορεί να εκτελέσει μια εργασία κάθε στιγμή και ότι η εκτέλεση μιας εργασίας είναι συνεχής, δηλαδή η μηχανή δεν μπορεί να σταματήσει την εκτέλεση μιας εργασίας ώστε να εκτελέσει μια άλλη και έπειτα να επιστρέψει στην πρώτη. Επίσης δυο εργασίες ορίζονται ως συμβατές μεταξύ τους αν δεν υπάρχει χρονική επικάλυψη, δηλαδή ισχύει ότι $s_i < s_j, f_i < s_j$ ή $s_i > s_j, f_i > f_j$.

Με χρήση δυναμικού προγραμματισμού βρείτε το υποσύνολο των αμοιβαία συμβατών εργασιών που μεγιστοποιεί το συνολικό όφελος των εργασιών που έχουν εκτελεστεί.

Επίσης για την επίλυση του συγκεκριμένου προβλήματος καλείστε να εφαρμόσετε έναν άπληστο αλγόριθμο και να συγκρίνετε τα αποτελέσματα με αυτά του δυναμικού προγραμματισμού. Συγκεκριμένα, ο άπληστος αλγόριθμος επιλέγει προς εκτέλεση σε κάθε βήμα την εργασία i που έχει το μέγιστο λόγο οφέλους ως προς χρονικής διάρκειας, $w_i/(f_i - s_i)$, από το υποσύνολο των διαθέσιμων προς εκτέλεση εργασιών.

Για την εκτέλεση των αλγορίθμων, θα δημιουργήσετε με τυχαίο τρόπο τις προς εκτέλεση εργασίες. Συγκεκριμένα, θα δημιουργήσετε εργασίες με χρονική διάρκεια, $(f_i - s_i)$, στο διάστημα $[5,20]$ και όφελος με τιμές στο διάστημα $[20,100]$. Επίσης, θα εκτελέσετε και θα συγκρίνετε τους αλγορίθμους για διαφορετικό μέγεθος εισόδου (πλήθος εργασιών), με τιμές στο διάστημα $[10,100]$ και βήμα 10 κάθε φορά. Η σύγκριση θα περιλαμβάνει τόσο τον χρόνο εκτέλεσης όσο και το συνολικό όφελος των προγραμματισμένων εργασιών όπως καθορίστηκαν από τους δύο αλγορίθμους.

Εύρεση Συντομότερου Μονοπατιού

Καλείστε να υλοποιήσετε τον αλγόριθμο εύρεσης συντομότερου μονοπατιού Dijkstra μεταξύ δύο κόμβων ενός κατευθυνόμενου γραφήματος.

Επίσης θα υλοποιήσετε μια παραλλαγή του αλγορίθμου Dijkstra (ARC-FLAGS). Συγκεκριμένα, κάθε ακμή του γραφήματος πέρα από το βάρος της θα περιέχει και ένα σύνολο ετικετών (label), μία για κάθε κόμβο του γραφήματος. Κάθε ετικέτα θα δείχνει την ύπαρξη ή μη συντομότερου μονοπατιού προς ένα συγκεκριμένο κόμβο το οποίο περιέχει την ακμή αυτή. Με βάση αυτή την πληροφορία, η αναζήτηση της συντομότερης διαδρομής προς ένα κόμβο μπορεί να περιοριστεί στις ακμές όπου η αντίστοιχη ετικέτα δείχνει ότι υπάρχει μονοπάτι προς τον κόμβο αυτό.

Επειδή η προσθήκη πληροφορίας σε κάθε ακμή για την ύπαρξη συντομότερου μονοπατιού προς κάθε κόμβο μέσω της ακμής αυτής αυξάνει κατά πολύ τον απαιτούμενο χώρο στη μνήμη, χωρίζουμε το γράφημα σε περιοχές και ορίζουμε για κάθε ακμή ένα πίνακα με μέγεθος όσο και το πλήθος των περιοχών αυτών.

Κάθε στοιχείο του πίνακα παίρνει την τιμή TRUE αν υπάρχει συντομότερο μονοπάτι προς τουλάχιστον ένα κόμβο της περιοχής που ξεκινάει από την ακμή αυτή και την τιμή FALSE αν δεν υπάρχει τέτοιο συντομότερο μονοπάτι.

Για την εξαγωγή της πληροφορίας αυτής για κάθε μια ακμή θα πρέπει να εκτελέσετε μια «προπαρασκευαστική» διαδικασία (preprocessing) η οποία θα έχει τα εξής βήματα:

- Αρχικοποίηση των τιμών του πίνακα όλων των ακμών του γραφήματος $G = (V, A)$ σε FALSE, όπου V το σύνολο των κόμβων και A το σύνολο των ακμών.
- Για κάθε μια ακμή $a \in A$ του γραφήματος G
 - Εκτέλεση του αλγόριθμου Dijkstra με αφετηρία την ουρά (tail) της ακμής.
 - Για κάθε κόμβο του οποίου η συντομότερη διαδρομή διέρχεται από την ακμή a , καταχώριση της τιμής TRUE στη θέση του πίνακα της a που αντιστοιχεί στην περιοχή που ανήκει ο κόμβος.

Έχοντας εκτελέσει την προπαρασκευαστική διαδικασία, η εύρεση του συντομότερου μονοπατιού μεταξύ ενός ζεύγους κόμβων μπορεί να περιοριστεί στο υπογράφημα του οποίου οι ακμές έχουν την τιμή TRUE στη θέση του πίνακα που αντιστοιχεί στην περιοχή που ανήκει ο προορισμός. Εκτενέστερη παρουσίαση της παραλλαγής του αλγορίθμου Dijkstra θα βρείτε στην ερευνητική εργασία **Error! Reference source not found.**

Για την εκτέλεση του αλγορίθμου καλείστε να δημιουργήσετε το γράφημα με τυχαίο τρόπο. Συγκεκριμένα θεωρήστε τετραγωνική επιφάνεια στον δισδιάστατο χώρο που ορίζεται από τις κορυφές με συντεταγμένες τα σημεία $(0,0)$ και $(1000,1000)$. Για τον διαχωρισμό του γραφήματος σε περιοχές θα δημιουργήσετε ένα πλέγμα μεγέθους 10×10 . Δηλαδή, κάθε υποδιαμέριση θα είναι διάστασης 100×100 . Θέλουμε να εφαρμόσουμε τον αλγόριθμο σε αραιό γράφημα. Για το σκοπό αυτό, σε κάθε διαμέριση του πλέγματος επιλέξτε με πιθανότητα $pr_{node} = 0.6$ να δημιουργήσετε κόμβους. Στις διαμερίσεις που θα περιέχουν κόμβους δημιουργείστε με τυχαίο τρόπο πλήθος

κόμβων στο διάστημα $[1,20]$. Η δημιουργία ενός κόμβου αντιστοιχεί στη δημιουργία ενός ζεύγους συντεταγμένων, συνεπώς για κάθε κόμβο θα πρέπει να δημιουργήσετε ένα ζεύγος συντεταγμένων εντός της διαμέρισης που ανήκει ο κόμβος. Αφού δημιουργήσετε τους κόμβους, θα δημιουργήσετε με τυχαίο τρόπο τις ακμές του γραφήματος. Συγκεκριμένα για ένα ζεύγος κόμβων δημιουργείτε μια ακμή με πιθανότητα $pr_{arc} = 0.8$. Θα ορίσετε επίσης με τυχαίο τρόπο ($pr_{dir} = 0.5$) την κατεύθυνση της ακμής μεταξύ των δύο κόμβων. Από τη στιγμή που οι κόμβοι αντιστοιχούν σε σημεία στο διδιάστατο χώρο, το βάρος κάθε ακμής ορίζεται ως η ευκλείδεια απόσταση των δύο κόμβων που ενώνει.

Αφού δημιουργήσετε το γράφημα και εκτελέσετε την προπαρασκευαστική διαδικασία για να δημιουργηθούν οι πίνακες των ακμών, επιλέξτε 10 ζεύγη κόμβων και εκτελέστε και συγκρίνετε τον χρόνο εκτέλεσης του αλγορίθμου Dijkstra καθώς και την παραλλαγή του. Για το δεύτερο αλγόριθμο θα υπολογίσετε το χρόνο εκτέλεσης ξεχωριστά για την προπαρασκευαστική φάση και την κύρια φάση εκτέλεσης.

Β' ΜΕΡΟΣ (40%)

1) Δίνεται ένα σύνολο A N θετικών ακεραίων και ένας θετικός ακέραιος K . Υπάρχει ένα υποσύνολο αριθμών του A που το άθροισμά τους είναι ακριβώς K ; Δώστε έναν αλγόριθμο που επιλύει το συγκεκριμένο πρόβλημα σε χρόνο $O(NK)$.

2) Εξηγείστε πως πρέπει να τροποποιηθεί ο αλγόριθμος του Dijkstra για να παράγει το πλήθος των διαφορετικών συντομότερων διαδρομών από το κόμβο αφετηρία v προς ένα κόμβο w . Επίσης, πως τροποποιούμε τον αλγόριθμο ώστε αν υπάρχουν περισσότερα του ενός ελάχιστα μονοπάτια από τον κόμβο αφετηρία προς ένα άλλο κόμβο, να επιλέγεται το μονοπάτι με το ελάχιστο πλήθος ακμών.

3) Δίνονται δύο ταξινομημένοι πίνακες A και B N στοιχείων ο καθένας. Δώστε ένα αλγόριθμο χρονικής πολυπλοκότητας $O(\log N)$ για την εύρεση του διάμεσου (median) του συνόλου $A \cup B$.

4) Έστω ταξινομημένος πίνακας A N στοιχείων. Τροποποιήστε τη διαδικασία δυαδικής αναζήτησης, ώστε να αναζητεί ένα στοιχείο x εντός χρόνου $O(\log k)$, όπου k η θέση του πίνακα όπου είναι τοποθετημένο είτε το x είτε το μεγαλύτερο στοιχείο μικρότερου του x .

5) Έστω πίνακας A $n \times n$ που περιέχει ακέραιους αριθμούς. Δώστε ένα αλγόριθμο χρονικής πολυπλοκότητας $O(n)$ που θα ελέγχει αν υπάρχει στοιχείο $a_{i,j}$ του πίνακα A που είναι μικρότερο από το άνω, κάτω, αριστερό και δεξιό γειτονικό του στοιχείο, δηλ. από τα στοιχεία $a_{i-1,j}$, $a_{i+1,j}$, $a_{i,j-1}$, $a_{i,j+1}$.

6) Ένα σύνολο ακμών $S \subseteq E$ ενός γραφήματος $G(V,E)$ καλείται σύνολο ακμών αναδρομής (feedback edge set) εάν κάθε απλός κύκλος περιέχει κάποια ακμή του S . Δοθέντος ενός γραφήματος $G(V,E)$ με θετικά βάρη στις ακμές του, δώστε ένα αλγόριθμο εύρεσης του ελαφρύτερου συνόλου ακμών αναδρομής.

Παράδοση Εργασίας

Η παράδοση της εργασίας θα γίνει μέσω της πλατφόρμας Thales. Η προθεσμία παράδοσης της εργασίας ορίζεται η **26/3/2025**. Για το πρώτο μέρος, θα πρέπει να παραδώσετε τον πηγαίο κώδικα, αναλυτικές οδηγίες εκτέλεσης του κώδικα, αναλυτική τεκμηρίωση της υλοποίησης και των αποτελεσμάτων των πειραμάτων σας. Για το δεύτερο μέρος, όπου απαιτείται αλγόριθμος θα πρέπει στην απάντησή σας να περιλαμβάνεται ψευδοκώδικας ο οποίος θα συνοδεύεται από αναλυτική τεκμηρίωση. Ιδιαίτερη έμφαση θα πρέπει να δοθεί στην απόδειξη της ορθότητας των αλγορίθμων

και της σχετικής χρονικής πολυπλοκότητας. Η εργασία μπορεί να εκπονηθεί από ομάδα μέχρι **δύο ατόμων**.

Αναφορές

- [1] Jugé, V. (2024). Adaptive Shivers sort: an alternative sorting algorithm. *ACM Transactions on Algorithms*, 20(4), 1-55.
- [2] Möhring, R.H., Schilling, H., Schütz, B., Wagner, D., Willhalm, T. (2005). Partitioning Graphs to Speed Up Dijkstra's Algorithm. In: Nikolettseas, S.E. (eds) *Experimental and Efficient Algorithms. WEA 2005. Lecture Notes in Computer Science*, vol 3503. Springer