Εισαγωγή

Το παρόν έγγραφο αφορά την παρουσίαση της ομαδικής εργασίας που υλοποιήθηκε στο πλαίσιο του μαθήματος Δομές Δεδομένων του χειμερινού εξαμήνου 2024-25 στο Οικονομικό Πανεπιστήμιο. Αποτελείται από τέσσερα μέρη.

1. 1.Την υλοποίηση των ΑΤΔ που χρειάστηκαν
2. Η υλοποίηση του αλγορίθμου Greedy
3. Η υλοποίηση του αλγορίθμου Heapsort
4. Εκτέλεση, πειραματισμός, αξιολόγηση

Την εργασία επιμελήθηκαν οι φοιτήτριες Ελένη Αντωνιάδη Τσαραμπουλίδη και Έλενα Μαρίνα Ρούσσου Γουμένου.

Περιεχόμενο

Μέρος Α: Υλοποίηση των ΑΤΔ

1. Κλάση Processor

Η κλάση Processor αναπαριστά έναν επεξεργαστή και περιλαμβάνει:

- Ένα μοναδικό ακέραιο id
- Λίστα processedJobs: περιέχει τις διεργασίες που έχει εκτελέσει ο επεξεργαστής
- Μέθοδος getTotalProcessingTime(): επιστρέφει τον συνολικό χρόνο που έχει δουλέψει ο επεξεργαστής
- Comparable<Processor>: έτσι μπορεί να χρησιμοποιηθεί σε ουρά προτεραιότητας. Η μέθοδος compareTo() συγκρίνει τον χρόνο του άλλου επεξεργαστή και σε περίπτωση ισοβαθμίας, επιλέγει τον επεξεργαστή με το μικρότερο id.

2. Κλάση Job

Η κλάση Job αναπαριστά μια διεργασία και περιλαμβάνει:

- Ένα μοναδικό ακέραιο id
- Time: αντιπροσωπεύει τον χρόνο της διεργασίας

3. Υλοποίηση Ουράς Προτεραιότητας (MaxPQ)

Ουρά προτεραιότητας με μεγιστοστρεφή σωρό (max-heap). Η ουρά αυτή αποθηκεύει αντικείμενα τύπου Processor και υποστηρίζει τις εξής λειτουργίες:

- isEmpty(): Επιστρέφει true αν η ουρά είναι κενή, αλλιώς false.
- size(): Επιστρέφει το πλήθος των στοιχείων στην ουρά.
- insert(Processor x): Προσθέτει έναν επεξεργαστή στο σωρό. Αν η ουρά έχει γεμίσει κατά 75%, διπλασιάζεται το μέγεθος του πίνακα.
- max(): Επιστρέφει τον επεξεργαστή με το μεγαλύτερο διαθέσιμο χρόνο χωρίς να τον αφαιρεί.
- getmax(): Επιστρέφει και αφαιρεί τον επεξεργαστή με τον μεγαλύτερο διαθέσιμο χρόνο από την ουρά.

Η MaxPQ είναι σχεδιασμένη έτσι ώστε να υποστηρίζει τους αλγορίθμους των επόμενων μερών της εργασίας.

Μέρος Β: Υλοποίηση του αλγορίθμου Greedy

Ο αλγόριθμος Greedy αναθέτει τις διεργασίες στους επεξεργαστές. Σε κάθε βήμα, η διεργασία τοποθετείται στον επεξεργαστή που έχει συγκεντρώσει τον λιγότερο χρόνο εργασίας μέχρι εκείνη τη στιγμή. Αν υπάρχουν ισοβαθμίες, επιλέγεται ο επεξεργαστής με το μικρότερο id.

1. **Δομή του αλγορίθμου**

Ο αλγόριθμος αποτελείται από τα εξής βήματα:

- Ανάγνωση του αρχείου εισόδου και αποθήκευση σε λίστα.
- Δημιουργία μιας MaxPQ που θα αποθηκεύει τους επεξεργαστές.
- Εισαγωγή όλων των επεξεργαστών στην ουρά.
- Για κάθε διεργασία:
    - Ανάθεση της στον επεξεργαστή με το μικρότερο συνολικό χρόνο επεξεργασίας.
    - Προσθήκη της διεργασίας στη λίστα processedJobs του επεξεργαστή.
    - Ενημέρωση του συνολικού χρόνου εργασίας του επεξεργαστή.
    - Επανατοποθέτηση του επεξεργαστή στην ουρά.

2. **Υλοποίηση του Comparator**

Για την εύρεση του επεξεργαστή με το μικρότερο χρόνο εκτέλεσης, η MaxPQ θα χρησιμοποιήσει έναν Comparator που θα συγκρίνει τους συνολικούς χρόνους των επεξεργαστών και σε περίπτωση ισοβαθμίας θα επιλέγει τον επεξεργαστή με το μικρότερο id.

3. **Αποτελέσματα**

Το πρόγραμμα εκτυπώνει:

- Το makespan.
- Αν ο αριθμός των διεργασιών είναι μικρότερος από 50, τυπώνονται τα στοιχεία των επεξεργαστών με βάση το συνολικό τους χρόνο.

Μέρος Γ: Υλοποίηση του αλγορίθμου Heapsort

Η Heapsort λειτουργεί ως εξής:

- Δημιουργία ενός min-heap από τη λίστα των διεργασιών.
- Ανταλλαγή του ριζικού στοιχείου (μικρότερη διεργασία) με το τελευταίο στοιχείο και αφαίρεσή του από το heap.
- Αναδόμηση του heap.
- Επανάληψη μέχρι η λίστα να ταξινομηθεί.

Με την ταξινόμηση αυτή, η πιο χρονοβόρα διεργασία τοποθετείται πρώτη και αποφεύγεται η δημιουργία μεγάλου makespan.

Μέρος Δ: Υλοποίηση της εφαρμογής σύγκρισης αλγορίθμων

1. Δομή της Εφαρμογής

Η εφαρμογή αποτελείται από:

- Την main η οποία εκτελεί την Comparisons
- Τη μέθοδο generateRandomTxt, η οποία δημιουργεί αρχεία τύπου `.txt` με τυχαίους χρόνους για τις διεργασίες, με ανώτατο όριο το 1000 για κάθε τυχαίο αριθμό.

2. Λειτουργία της Comparisons

- Δημιουργεί 10 αρχεία τύπου .txt για κάθε N χρησιμοποιώντας την μέθοδο generateRandomTxt.
- Για κάθε αρχείο διαβάζει τις διεργασίες και τις αποθηκεύει σε μια λίστα jobs.
- Δημιουργεί αντίγραφο της jobs, την jobs2 και την ταξινομεί τις διεργασίες σε φθίνουσα σειρά με την χρήση της Sort.
- Υπολογίζει το makespan γιά τους 2 αλγορίθμους για κάθε αρχείο.
- Υπολογίζει μέσο όρο των makespan των 2 αλγορίθμων για κάθε N ξεχωριστά και συνολικό μέσο όρο από όλα τα αρχεία.
- Συγκρίνει τους δύο αλγορίθμους τόσο για κάθε N ξεχωριστά όσο και συνολικά και εκτυπώνει τα αποτελέσματα. .

3. Συμπέρασμα από την Πειραματική Αξιολόγηση των Αλγορίθμων

Δοκιμάζοντας τους δύο αλγορίθμους για **N = 50, 100, 150, 200, 400, 500, 600**, φαίνεται πως ο αλγόριθμος 2 (Greedy Decreasing) είναι πιο αποδοτικός για κάθε μία από τις παραπάνω τιμές του N, καθώς και συνολικά. Σε κάθε δοκιμή, το Average Makespan του δεύτερου αλγορίθμου είναι μικρότερο από αυτό του πρώτου, όπως φαίνεται στον παρακάτω πίνακα, που δημιουργήθηκε με χρήση του Excel. Το αποτέλεσμα αυτό είναι αναμενόμενο, δεδομένου του τρόπου λειτουργίας του ίδιου του αλγορίθμου. Συγκεκριμένα, ο αλγόριθμος δίνει προτεραιότητα στον επεξεργαστή με τον μικρότερο συνολικό χρόνο διεργασιών για να αναλάβει την επόμενη διεργασία. Επειδή στον δεύτερο αλγόριθμο η λίστα των διεργασιών είναι ταξινομημένη σε φθίνουσα σειρά, επιτυγχάνεται καλύτερη κατανομή του φόρτου εργασίας και αποτρέπεται το ενδεχόμενο ένας επεξεργαστής να επιβαρυνθεί υπερβολικά με διεργασίες μεγάλου μεγέθους. Ως αποτέλεσμα, το makespan παραμένει χαμηλότερο, καθιστώντας τον δεύτερο αλγόριθμο πιο αποδοτικό.

N	Average Makespan Greedy	Average Makespan Greedy Decreasing
50	4005.9	3629.3
100	5497.6	5064.1
150	6691	6260.6
200	7548.8	7073.5
400	10266.8	9798.5
500	11717.9	11209.8
600	12963.9	12442
Total Average	8384.557	7925.4