**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)**

Oυρά 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.

1. **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.

1. **Aποτελέσματα**

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.