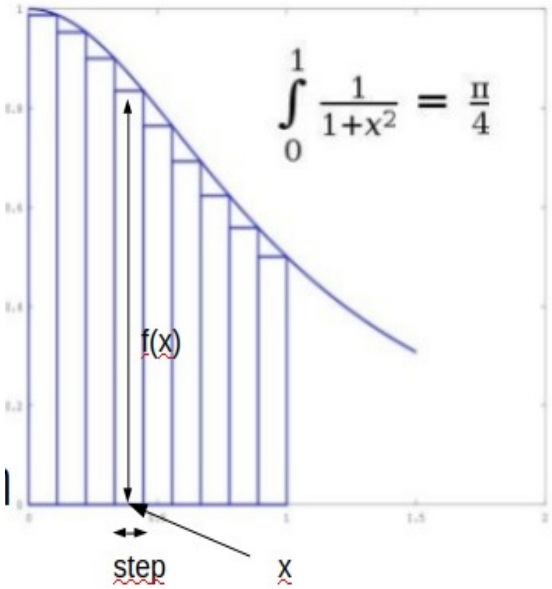
Παράλληλος και Κατανεμημένος Υπολογισμός

Εργαστήριο 6

1. Υπολογισμός του π με αριθμητική ολοκλήρωση

Αφού μελετήσετε τον Code/VecSum εφαρμόστε μερικές από τις αντίστοιχες μεθόδους σε ένα πρόβλημα αριθμητικής ολοκλήρωσης, στον υπολογισμό του π.

Η ιδέα του υπολογισμού του π με αριθμητική ολοκλήρωση φαίνεται παρακάτω. Χωρίζουμε το πεδίο τιμών σε numSteps μικρά τμήματα, με μήκος step. Στο μέσο κάθε τμήματος υπολογίζουμε τα x, f(x) και αθροίζουμε τις τιμές των εμβαδών step*f(x) για να υπολογίσουμε το π/4.




```
step = 1.0 / numSteps;
for (i=0; i < numSteps; ++i) {
            x = ((double)i+0.5)*step;
            sum += 4.0/(1.0+x*x);
}
pi = sum * step;
```

f(x)

Μελετήστε τον ακολουθιακό κώδικα στο φάκελλο NumInt. Εκτελέστε πειράματα με τον ακολουθιακό κώδικα και χρονομέτρηση για να καταλάβετε τη συμπεριφορά του κώδικα, δηλαδή να διαπιστώσετε ότι ο χρόνος εκτέλεσης είναι ευθέως ανάλογος της τιμής numSteps. Για παράδειγμα επιλέξτε numSteps ίσο με 1εκ 10εκ, 100εκ, 1δισεκ.

Στη συνέχεια με βάση το κώδικα στο Code/VecSum υλοποιήστε μερικά από τα σχήματα απεικόνισης-αναγωγής και συγκρίνετε τους χρόνους εκτέλεσης.

Δοκιμάστε τουλάχιστο τα παρακάτω – πάντα εννοείται με παραγωγή ορθού αποτελέσματος:
1. μια μέθοδο χωρίς αμοιβαίο αποκλεισμό
2. μια μέθοδο με αμοιβαίο αποκλεισμό, χρήση locks ή synchronized (αλλά όχι getClass)
3. το σχήμα με λανθασμένη χρήση του αμοιβαίου αποκλεισμού για να διαπιστώσετε τις διαφορές χρόνου με την ορθή χρήση του αμοιβαίου αποκλεισμού.

Ο σύνδεσμος https://en.wikipedia.org/wiki/Approximations_of_%CF%80
Παραθέτει δεκάδες αλγορίθμους υπολογισμού του π. Δείτε για παράδειγμα τη μέθοδο Monte Carlo (επιφάνεια τεταρτημορίου κύκλου, δείτε το σχετικό animation στον παραπάνω σύνδεσμο) και σκεφτείτε τη παραλληλοποίησή του. Θα βρείτε πολλές υλοποιήσεις στο Διαδίκτυο.

2. Μελετήσετε τον κώδικα Code/MapReduce/BruteForceStringMatch.java

Προσθέστε εντολές μέτρησης χρόνου.
Χρησιμοποιείστε το αρχείο E.coli (πλήρες DNA του βακτηριδίου Escherichia coli)από το Large Corpus στο σύνδεσμο https://corpus.canterbury.ac.nz/descriptions/#large (περίπου 4,6 Gbyets)

Μπορείτε για παράδειγμα να τετραπλασιάσετε το μέγεθος του αρχείου ως εξής:
Windows: type file.txt file.txt file.txt file.txt > fourtimes.txt
Linux: cat file.txt file.txt file.txt file.txt > fourtimes.txt

Διαλέξτε ένα pattern, για παράδειγμα “tacccagattatcgccatcaacgg” .
Χρησιμοποιείστε τις τεχνικές απεικόνισης και αναγωγής για να παραλληλίσετε τον κώδικα.
Μετρήστε το χρόνο εκτέλεσης της παράλληλης έκδοσης. Για να δείτε αξιόπιστους χρόνους χρειάζεται πολύ μεγάλο αρχείο, πχ 16 φορές το E.coli.

Η ταύτιση προτύπων (pattern matching) – εδώ ταυτίζουμε strings αλλά θα μπορούσαμε να το γενικεύσουμε και σε άλλα δεδομένα-- αποτελεί μια από τις θεμελιώδεις λειτουργίες στην ανάκτηση πληροφοριών (πχ grep), ασφάλεια υπολογιστών (πχ snort), υπολογιστική βιολογία (γονιδίωμα, πρωτεΐνες κλπ) και σε πολλές άλλες εφαρμογές. Παρακάτω δίνονται ορισμένα ενδεικτικά links για όσους ενδιαφέρονται.

https://www-igm.univ-mlv.fr/~lecroq/string/
https://algs4.cs.princeton.edu/50strings/
https://smart-tool.github.io/smart/

3. Η δεύτερη άσκηση του Εργαστηρίου 5 προτείνει την αποθήκευση των αποτελεσμάτων όλων των νημάτων σε μια συνδεδεμένη λίστα. Αυτό απαιτεί κάποιας μορφής αμοιβαίο αποκλεισμό.

Σκεφτείτε και να υλοποιήσετε μια λύση τύπου αναγωγής: Για παράδειγμα, κάθε νήμα μπορεί να έχει μια ξεχωριστή συνδεδεμένη λίστα (δηλαδή να έχουμε array of lists). Όταν τα νήματα τερματιστούν το main θα μπορούσε να διαχειριστεί όλα τα αποτελέσματα, ακόμη και να συνεννώσει όλες τις λίστες σε μία. Μπορείτε να σκεφτείτε και άλλες λύσεις..