

Ενότητα Γ: Φίλτρα Bloom

Άσκηση 1: Υπολογισμός τομής

Έστω ότι έχουμε μία συλλογή από τεράστια αρχεία που βρίσκονται κατανεμημένα σε δύο servers A και B. Ορισμένα αρχεία περιέχονται και στους δύο servers (ως ακριβή αντίγραφα), και θα θέλαμε να τα βρούμε. Μας δίνεται ως πληροφορία ότι αυτά τα κοινά αρχεία είναι πολύ λιγότερα απ' όσα περιέχονται στους A και B. Η επικοινωνία μεταξύ των servers κοστίζει πολύ, οπότε θα θέλαμε να υπολογίσουμε τα κοινά αρχεία των A και B ελαχιστοποιώντας την επικοινωνία μεταξύ τους. Για τον σκοπό αυτό, μπορούμε να χρησιμοποιήσουμε φίλτρα Bloom ως εξής.

Ο A θα υπολογίσει ένα φίλτρο Bloom για τα αρχεία που περιέχονται σε αυτόν, και θα το στείλει στον B για να τσεκάρει ποια από τα δικά του βρίσκονται στον A. Έπειτα, ο B θα στείλει στον A αυτά που πέρασαν το φίλτρο (δηλαδή αυτά που το φίλτρο ανέφερε ότι ανήκουν στον A), και ο A θα τσεκάρει ποια από αυτά ανήκουν όντως σε αυτόν. (Υποθέτουμε ότι ο A μπορεί να κάνει γρήγορο έλεγχο ύπαρξης, έχοντας οργανώσει τα αρχεία του σε μια αποδοτική δομή δεδομένων, π.χ., σε tries, ή, έχοντας ταξινομήσει τα αρχεία του, κάνοντας δυαδική αναζήτηση.)

Ας πούμε ότι η παραπάνω διαδικασία είναι “ένας γύρος επικοινωνίας”. Προφανώς κάποια αρχεία που ο B έστειλε στον A δεν ανήκουν στον A, αλλά στάλθηκαν λόγω του false positive bias του φίλτρου Bloom. Παρά ταύτα, κατά πάσα πιθανότητα πετύχαμε κάτι καλύτερο απ' το να στέλναμε όλα τα αρχεία του B στον A.

Υπάρχει όμως το ενδεχόμενο να μπορούμε να κάνουμε και κάτι πολύ καλύτερο. Αντί ο B στον πρώτο γύρο να στείλει τα αρχεία του, θα στείλει ένα δικό του φίλτρο Bloom στον A, που σχηματίζεται από τα αρχεία του B που πέρασαν από το φίλτρο του A. Έπειτα ο A θα στείλει ένα καινούριο φίλτρο στον B με βάση αυτό που του έστειλε ο B, και τώρα ο B θα στείλει στον A όσα αρχεία περνούν από αυτό το δεύτερο φίλτρο, όπως και πριν. Ας πούμε ότι αυτή η διαδικασία συνιστά “δύο γύρους επικοινωνίας”.

Αυτό το σχήμα επικοινωνίας γενικεύεται άμεσα σε “k γύρους επικοινωνίας”.

α) (Μονάδες: 1.75)

Υλοποιήστε την παραπάνω ιδέα ως εξής.

Πρώτα απ' όλα, θα φροντίσουμε οι A και B να περιέχουν 100.000 αρχεία ο καθένας, αλλά μόνο 10 θα είναι κοινά σε αυτούς. Τα "αρχεία" για εμάς θα είναι απλώς συμβολοσειρές των 100 bits.

Για να υλοποιήσουμε αυτήν την κατάσταση, πρώτα θα φτιάξουμε 10 τυχαία αρχεία. (Δηλαδή, 10 εντελώς τυχαίες συμβολοσειρές των 100 bits.) Έπειτα θα φτιάξουμε 99.990 τυχαία αρχεία για τον A, 99.990 τυχαία αρχεία για τον B, και θα προσθέσουμε στους A και B επιπλέον τα 10 αρχεία που φτιάξαμε στην αρχή.

Ισχυριζόμαστε ότι αυτή η διαδικασία εξασφαλίζει με πρακτικά απόλυτη βεβαιότητα ότι μόνο τα 10 αρχεία που φτιάξαμε στην αρχή θα ανήκουν στους A και B ταυτόχρονα. Τεκμηριώστε αυτόν τον ισχυρισμό, εκτιμώντας την πιθανότητα τα κοινά αρχεία στους A και B, μετά από την όλη αυτή διαδικασία, να είναι πάνω από 10. (Μονάδες: 0.125).

Κάθε συνάρτηση κατακερματισμού που θα χρησιμοποιήσουμε για τα φίλτρα Bloom κατασκευάζεται ως εξής. Έστω $p = 1048583$, και έστω N το πλήθος των bits του φίλτρου Bloom. (Ο p είναι ο μικρότερος πρώτος αριθμός που είναι μεγαλύτερος από το 2^{20} .) Αρχικά, επιλέγουμε τυχαία έναν αριθμό α μεταξύ 1 και $p - 1$, και έναν τυχαίο αριθμό β μεταξύ 0 και $p - 1$. Αυτές οι τέσσερις παράμετροι (α , β , p , N) καθορίζουν μια συνάρτηση κατακερματισμού h . Έστω τώρα ένα αρχείο X , και ο σκοπός είναι να περιγράψουμε πώς υπολογίζεται το $h(X)$. Αρχικά, χωρίζουμε το X σε 5 συνεχόμενα κομμάτια B_1, B_2, B_3, B_4, B_5 των 20 bits. Θεωρούμε κάθε ένα από αυτά τα κομμάτια ως έναν αριθμό μεταξύ 0 και $2^{20} - 1$, που η δυαδική του αναπαράσταση δίνεται από τα bits που συνιστούν το κομμάτι. (Π.χ., ένα κομμάτι 00000000000000001101 ερμηνεύεται ως ο αριθμός 13.) Αρχικοποιούμε μια τιμή $c_0 \leftarrow 0$, και υπολογίζουμε αναδρομικά την τιμή $c_i = (\alpha(B_i + c_{i-1}) + \beta) \bmod p$, για κάθε $i \in \{1, 2, 3, 4, 5\}$. Τελικά, επιστρέφουμε το $c_5 \bmod N$.

Για αρχή, ας υποθέσουμε ότι $N = 500.000$. (Οπότε, κάθε φίλτρο Bloom που στέλνεται από τον A στον B, ή αντίστροφα, αποτελείται από 500.000 bits. Παρατηρήστε όμως ότι ο κάθε ένας από τους A και B θα πρέπει να στέλνει στον άλλον και τις συναρτήσεις κατακερματισμού που χρησιμοποίησε.)

Υλοποιήστε την μέθοδο που περιγράψαμε για τον υπολογισμό των κοινών αρχείων των A και B. Το πρόγραμμά σας αρκεί να εκτυπώνει μονάχα το

πλήθος των positives που υπολόγισε ο B στο τελευταίο φίλτρο Bloom που του έστειλε ο A. Ας ονομάσουμε αυτήν την παράμετρο ύψιστου ενδιαφέροντος **count**. (Όμως τσεκάρετε κι όλες ότι στα αρχεία που ο B πήρε positive στο τέλος περιλαμβάνονται τα 10 που ανήκουν και στον A, και εκτυπώστε “true” σε αυτήν την περίπτωση. (Μονάδες για την υλοποίηση αυτού του ελέγχου: 0.25))

Η υλοποίησή σας θα πρέπει να αφήνει την ελευθερία στον χρήστη να επιλέξει: 1) το πλήθος των συναρτήσεων κατακερματισμού (που θα αφορά όλα τα φίλτρα Bloom που θα δημιουργηθούν), και 2) το πλήθος των γύρων επικοινωνίας.

Επιπλέον, δώστε σαν επιλογή, αντί όλα τα φίλτρα Bloom να αποτελούνται από 500.000 bits, το πλήθος των bits να αλλάζει με βάση το πόσα αρχεία εκτίμησε ο ένας να περιέχονται στον άλλον. Συγκεκριμένα, αν π.χ. πρόκειται να έχουμε πάνω από έναν γύρο επικοινωνίας, τότε ο B, αν στον πρώτο γύρο επικοινωνίας βρει ότι N_1 αρχεία του περνούν το φίλτρο που του έστειλε ο A, τότε θα στείλει ένα φίλτρο Bloom στον A με $5N_1$ bits. Έπειτα, αν ο A βρει N_2 αρχεία του να περνούν το φίλτρο που του έστειλε το B, τότε στέλνει ένα φίλτρο Bloom στον B με $5N_2$ bits, κ.ο.κ. Θεωρείτε ότι αυτό θα βελτιώσει ή θα χειροτερέψει το αποτέλεσμα **count**; (Για το πρόβλημά μας, προφανώς το “χειροτέρεμα” είναι το **count** να αυξάνεται.)

Δοκιμάστε τις εξής παραλλαγές (σε όλους τους δυνατούς συνδυασμούς): i) το πλήθος των συναρτήσεων κατακερματισμού να είναι 1, 2, 3, 5, 10, και ii) το πλήθος των γύρων επικοινωνίας να είναι 1, 2, 3, 4, 5. Για ποιον συνδυασμό βλέπετε να έχουμε το βέλτιστο **count**;

β) (Μονάδες: 0.25)

Ίσως παρατηρήσατε (ή υποψιάζεστε) ότι όσο το πλήθος των γύρων επικοινωνίας αυξάνεται, τόσο περισσότερο αναμένεται να βελτιώνεται (δηλαδή να μειώνεται) το πλήθος των αρχείων που πρέπει τελικά ο B να στείλει στον A (δηλαδή το **count**). Δώστε μία πειστική μαθηματική εξήγηση για αυτήν την παρατήρηση (ή υποψία), ή καταρρίψτε αυτόν τον ισχυρισμό.

Άσκηση 2: Ιχνηλάτηση πακέτου (Packet Tracing)

Έστω ότι έχουμε ένα δίκτυο από routerάκια, τα οποία προωθούν μηνύματα το ένα στο άλλο. Ανά πάσα ώρα και στιγμή, ένα routerάκι

μπορεί να δημιουργήσει ένα μήνυμα, το οποίο θα το στείλει σε κάποιο από τα routerάκια με τα οποία είναι συνδεδεμένο ως πομπός. (Το ποιος μπορεί να στείλει μήνυμα σε ποιον εξαρτάται από μία προκαθορισμένη τοπολογία.) Ένα routerάκι που δέχεται μήνυμα, μπορεί να αναρωτιέται ποιος ήταν αυτός που το δημιούργησε αρχικά. Σε αυτήν την περίπτωση, μπορεί να ρωτήσει ένα-ένα τα routerάκια τα οποία συνδέονται με αυτό (ως πομποί του), ώστε να μάθει ποιο του έστειλε το μήνυμα, και έπειτα αυτό να ρωτήσει με την σειρά του τους πομπούς του. Το πρόβλημα είναι ότι τα routerάκια έχουν πολύ περιορισμένη μνήμη, οπότε δεν μπορούν να κρατούν αυτούσια τα μηνύματα που δέχονται.

Σε ένα τέτοιο σενάριο, τα φίλτρα Bloom μπορούν να φανούν πολύ χρήσιμα. Υποθέτουμε ότι κάθε routerάκι κρατάει ένα φίλτρο Bloom για τα μηνύματα που δέχτηκε, και άρα η ιχνηλάτηση του μηνύματος θα γίνει χρησιμοποιώντας τα φίλτρα Bloom που είναι αποθηκευμένα στα routerάκια. Παρατηρήστε ότι αυτό θα κάνει ορισμένα routerάκια να ισχυρίζονται ότι ένα μήνυμα πέρασε από αυτά, ή δημιουργήθηκε από αυτά, χωρίς να ισχύει κάτι τέτοιο. Αν όμως καταλήξουμε σε μοναδικό routerάκι που ισχυρίζεται ότι δημιούργησε το μήνυμα, τότε θεωρούμε την ιχνηλάτηση πετυχημένη.

α) (Μονάδες: 2.25)

Ο σκοπός είναι να προσομοιώσουμε την διαδικασία προώθησης μηνυμάτων και ιχνηλάτησής τους, χρησιμοποιώντας φίλτρα Bloom, και να υπολογίσουμε το πλήθος των πετυχημένων ιχνηλατήσεων.

Συγκεκριμένα, έχουμε διάφορα δίκτυα από routerάκια, τα οποία έχουν την μορφή κατευθυνόμενου γραφήματος. Τα routerάκια είναι αριθμημένα ως 1, 2, 3, Υποθέτουμε ότι μόνο τα 1, 2, . . . , 10 δημιουργούν μηνύματα, και το τελευταίο router (δηλαδή αυτό με το μεγαλύτερο νούμερο) είναι ο παραλήπτης των μηνυμάτων που εκκινεί την διαδικασία ιχνηλάτησης.

Τα γραφήματα περιέχονται στον φάκελο Networks, στην σελίδα του μαθήματος στο ecourse, και δίνονται σε μορφή αρχείου κειμένου που έχει την εξής μορφή. Η πρώτη γραμμή έχει την μορφή “n m”, όπου n είναι το πλήθος των κόμβων (δηλαδή των router), και m είναι το πλήθος των ακμών (συνδέσεων). Κάθε μια από τις επόμενες m γραμμές έχει την μορφή “x y”, που σημαίνει ότι ο x μπορεί να στέλνει μηνύματα στον y. Τα γραφήματα αυτά δεν περιέχουν κύκλους (οπότε η προώθηση μηνύματος δεν πρόκειται να πέσει σε ατέρμονα βρόχο), και από κάθε έναν από τους

κόμβους 1, 2, . . . , 10 υπάρχει (κατευθυνόμενο) μονοπάτι προς τον n (τον τελικό κόμβο-παραλήπτη). Επιπλέον, τα routerάκια 1, 2, . . . , 10 έχουν μόνο εξερχόμενες συνδέσεις προς άλλα routerάκια (δηλαδή λειτουργούν μονάχα ως πομποί, και όχι ως δέκτες).

Η διαδικασία δημιουργίας και προώθησης μηνύματος λειτουργεί ως εξής. Επιλέγουμε να παραχθούν 100.000 μηνύματα, όπου κάθε μήνυμα συνιστά μια συμβολοσειρά των 100 bits. Κάθε μήνυμα παράγεται τυχαία, και με τυχαίο τρόπο επιλέγεται ένα από τα routerάκια 1, 2, . . . , 10 ως αυτό που το δημιουργήσει. Έπειτα, δεδομένου ότι το μήνυμα βρίσκεται σε ένα router x που δεν είναι το τερματικό (δηλαδή το n), επιλέγουμε τυχαία ένα από τα router-δέκτες του x (με άλλα λόγια: έναν κόμβο y για τον οποίο υπάρχει ακμή (x,y)), και στέλνουμε το μήνυμα στο y . Κάθε router έχει το δικό του φίλτρο Bloom, και κάθε φορά που περνάει μήνυμα από αυτό (ή που δημιουργείται από αυτό), το φίλτρο Bloom του καταγράφει ότι πέρασε από εκεί αυτό το μήνυμα.

Μόλις σταλθούν και τα 100.000 μηνύματα, το τελευταίο routerάκι αρχίζει να τα ιχνηλατεί ένα προς ένα. Για κάθε μήνυμα, ρωτάει τους πομπούς του αν το δέχτηκαν. Όσοι είπαν “NAI”, κάνουν το ίδιο, μέχρι κάποιο από τα αρχικά routerάκια (δηλαδή κάποιο από τα 1, 2, . . . , 10) να πει “NAI”. Σε περίπτωση που μόνο ένα από τα αρχικά routerάκια καταλήξει να έχει πει “NAI”, θεωρούμε την ιχνηλάτηση πετυχημένη.

Στο τέλος, θα πρέπει να εκτυπώνουμε το πλήθος των πετυχημένων ιχνηλατήσεων. (Προφανώς αυτό θα ποικίλλει, ανάλογα με την τυχαιότητα στα φίλτρα Bloom, αλλά αναμένεται ίσως να υπάρχει κάποια ξεκάθαρη τιμή για κάθε γράφημα, γύρω απ’ την οποία συγκεντρώνονται οι τιμές από κάθε πείραμα ιχνηλάτησης.)

Θα δώσουμε σε κάθε routerάκι 10.000 bits μνήμης για το φίλτρο Bloom του, και για αρχή ας δοκιμάσουμε μόνο μία συνάρτηση κατακερματισμού (μία διαφορετική, για κάθε φιλτράκι). Οι συναρτήσεις κατακερματισμού θα είναι ακριβώς του ίδιου τύπου όπως και στην Άσκηση 1. Έπειτα δοκιμάστε περισσότερες συναρτήσεις κατακερματισμού (για κάθε φιλτράκι), και αναφέρετε για κάθε γράφημα ποιο πλήθος συναρτήσεων κατακερματισμού σάς φαίνεται ότι δίνει την καλύτερη απόδοση. (Η απόδοση εδώ μετράται ως το αναμενόμενο πλήθος των επιτυχημένων ιχνηλατήσεων.)

Ίσως μπορείτε να αντιληφθείτε ότι, κατά την διαδικασία της ιχνηλάτησης, ορισμένα routerάκια μπορεί να δεχτούν πάρα πολλά requests για έλεγχο ύπαρξης του μηνύματος στο φίλτρο Bloom τους. (Εξηγήστε γιατί μπορεί να συμβεί αυτό, και δώστε και ένα θεωρητικό παράδειγμα.) Αυτό θα ήταν αναπόφευκτο σε μια πραγματική εφαρμογή, και θα πρέπει να κρατήσετε αυτήν την υλοποίηση.

Παρά ταύτα, στο πλαίσιο της προσομοίωσης, μπορείτε να υλοποιήσετε επιπλέον και έναν πολύ πιο αποδοτικό αλγόριθμο για τον υπολογισμό των πετυχημένων ιχνηλατήσεων. Οι 0.5 μονάδες αυτής της άσκησης θα δοθούν για μία τέτοια αποδοτική υλοποίηση.

β) (Μονάδες: 0.75)

Ίσως παρατηρήσατε ότι σε ορισμένα γραφήματα η ιχνηλάτηση είναι γενικά πιο πετυχημένη απ' ό,τι σε κάποια άλλα. Πώς εξηγείτε την διαφορά στην επίδοση μεταξύ των διαφόρων γραφημάτων; Στην επόμενη σελίδα μπορείτε να δείτε μία αναπαράσταση όλων των γραφημάτων που βρίσκονται στον φάκελο Networks.

