Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms

ΑΛΜΑ
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Εθνικό Μετσόβιο Πολυτεχνείο

Ιανουάριος, 2022

Heaps

Σωρός είναι μία δομή δεδομένων που αποτελείται από ένα σύνολο αντικειμένων, κάθε ένα από τα οποία συνοδεύεται από έναν αριθμό κλειδί. Οι λειτουργίες του:

- Make heap: Επιστρέφουμε νέο, άδειο σωρό
- Insert (i, h): Εισάγουμε αντικείμενο i με καθορισμένο κλειδί στο σωρό h
- Find min (h): Βρίσκουμε το αντικείμενο με το ελάχιστο κλειδί στον h
- Delete min (h): Διαγράφουμε το αντικείμενο με το ελάχιστο κλειδί και το επιστρέφουμε.
- Delete (i, h): Διαγράφουμε το αντικείμενο i και το επιστρέφουμε.
- Meld (h_1, h_2): Συνδυάζουμε δύο ξένους σωρούς h_1, h_2.
- Decrease key (Δ, i, h): Μειώνουμε το κλειδί του i κατα Δ.

Heap-Ordered Trees

Για να υλοποιήσουμε τους σωρούς θα χρησιμοποιήσουμε δέντρα, τα heap-ordered trees:

Ορισμός

Ένα heap-ordered tree είναι ένα δέντρο με ρίζα, που περιλαμβάνει ένα σύνολο από αντικείμενα (ένα αντικείμενο σε κάθε κόμβο),έτσι ώστε:
αν ο x είναι ένας κόμβος, τότε το κλειδί του αντικειμένου μέσα στον x δεν μπορεί να είναι μικρότερο από το κλειδί του αντικειμένου στο γονέα $p(x)$ του x, δηλαδή στον κόμβο $p(x)$ (δεδομένου ότι ο x έχει γονέα στο δέντρο).

Heap-Ordered Trees

Παρατήρηση: Η ρίζα του δέντρου είναι ο κόμβος που περιέχει το αντικείμενο με το μικρότερο κλειδί.

Βασική Λειτουργία-Ένωση(Linking):

Η βασική λειτουργία των heap-ordered trees είναι η ένωση (linking) που ενώνει δύο ξένα (από πλευράς αντικειμένων) δέντρα σε ένα.

Δοσμένων δύο δέντρων με ρίζες x και y αντίστοιχα, τα ενώνουμε συγκρίνοντας τα κλειδιά των αντικειμένων στους κόμβους x και y. Αν το αντικείμενο στον κόμβο x έχει μικρότερο κλειδί, τότε κάνουμε τον y παιδί του x. (Αντίστοιχα αν το αντικείμενο στον y έχει το μικρότερο κλειδί, θα κάνουμε τον x παιδί του y).

Ορισμός

Ένας σωρός Fibonacci (F-Heap) είναι μία συλλογή από ξένα (όσον αφορά τα αντικείμενα) heap-ordered trees.

Σημείωση: Το πλήθος των παιδιών ενός κόμβου x είναι ο βαθμός του (rank) $r(x)$.

Αναπαράσταση F-Heap

Θεωρούμε την εξής αναπαράσταση των σωρών: Κάθε κόμβος περιέχει μία θέση μνήμης για το βαθμό του, ένα δείκτη προς το γονέα του και ένα δείκτη προς ένα από τα παιδιά του. Τα παιδιά ενός κόμβου είναι διπλά ενωμένα μεταξύ τους σχηματίζοντας μία κυκλική λίστα (κάθε παιδί έχει ένα δείκτη προς τον αριστερό "αδερφό" του και έναν δείκτη προς το δεξί "αδερφό" του. Οι ρίζες όλων των δέντρων είναι επίσης διπλά ενωμένες σχηματίζοντας μία κυκλική λίστα. Εμείς έχουμε πρόσβαση στο σωρό μέσω ενός δείκτη προς τη ρίζα που περιέχει το αντικείμενο με το μικρότερο κλειδί. Καλούμε αυτή τη ρίζα ελάχιστο κόμβο του σωρού (minimum node).

F-Heap Operations

- Make heap: Επιστρέφουμε δείκτη προς ειδικό κόμβο null.
- Find min(h): Επιστρέφουμε το αντικείμενο του ελάχιστου κόμβου.
- Meld (h_1, h_2): Ενώνουμε τις λίστες ριζών των σωρών h_1 και h_2 σε μία λίστα. Επιστρέφουμε ως ελάχιστο κόμβο (minimum node) του νέου σωρού, είτε τον ελάχιστο κόμβο του h_1, είτε αυτόν του h_2, κρίνοντας από το ποιος έχει το αντικείμενο με το μικρότερο κλειδί.
- Insert (i, h): Δημιουργούμε νέο σωρό αποτελούμενο από έναν κόμβο που θα περιέχει το αντικείμενο i, και έπειτα εκτελούμε την Meld ανάμεσα στο νέο κόμβο και στον h.

Όλες οι παραπάνω λειτουργίες εκτελούνται σε χρόνο $O(1)$.

F-Heap Operations

- Delete min (h):
  1. Αφαιρούμε τον ελάχιστο κόμβο x από τον h.
  2. Συνδέουμε τη λίστα των παιδιών του x με τη λίστα ριζών του h (χωρίς το x πλέον).
  3. Επαναλαμβάνουμε το ακόλουθο βήμα μέχρι να μην εφαρμόζεται πια:
     Linking step: Βρίσκουμε δύο δέντρα των οποίων οι ρίζες έχουν τον ίδιο βαθμό rank, και τα ενώνουμε (εκτελώντας την ένωση (linking) που ορίσαμε για τα heap-ordered trees).
  4. Όταν δεν υπάρχουν πια δέντρα με ρίζες ίδιου βαθμού, συνδέουμε τις ρίζες που έχουν απομείνει σε κυκλική λίστα, ενώ παράλληλα βρίσκουμε το νέο ελάχιστο κόμβο.
  5. Αποθηκεύουμε το αντικείμενο του κόμβου x, διαγράφουμε τον x και επιστρέφουμε το αποθηκευμένο αντικείμενο.

F-Heap Operations

- Decrease key (Δ, i, h): Αφαιρούμε Δ από το κλειδί του αντικειμένου i, βρίσκουμε τον κόμβο x του i, κόβουμε την ακμή που τον ενώνει με τον γονέα του $p(x)$, ενώ ταυτόχρονα αποκλείουμε τον x από τη λίστα των παιδιών του $p(x)$, μειώνοντας κατά 1 το βαθμό του $p(x)$. Το υπόδεντρο με ρίζα το x είναι ένα νέο δέντρο και ο x συνδέεται με τη λίστα των υπόλοιπων ριζών του h. (Αν ο x ήταν ρίζα κάποιου δέντρου στον h, απλώς θα αφαιρούσαμε Δ από το κλειδί του, και ίσως να χρειαζόταν να αντικαταστήσουμε τον ελάχιστο κόμβο με τον x).
- Delete (i, h): Όπως πάνω κόβουμε τον x (κόμβος που περιέχει το i) από τον γονέα του και τη λίστα με τα αδέρφια του. Συνδέουμε τα παιδιά του x με τη λίστα ριζών του h και καταστρέφουμε τον κόμβο x.

F-Heap Operations

Ο χρόνος που απαιτείται για τη λειτουργία Delete min είναι $O(logn)$.
Ο χρόνος που απαιτείται για το Decrease key και το Delete είναι $O(1)$, ενώ στην περίπτωση του Delete, αν ο κόμβος που πρόκειται να διαγράψουμε είναι ο ελάχιστος κόμβος και τότε μεταφερόμαστε στη λειτουργία Delete min με απαιτούμενο χρόνο $O(logn)$

Απαιτούμενος Χρόνος

Όλες οι λειτουργίες που ορίσαμε πάνω στους σωρούς Fibonacci είναι χρόνου $O(1)$, εκτός από τη λειτουργία Delete min, που είναι χρόνου $O(logn)$.

Cascading Cuts

Μία πρόσθετη λεπτομέρια, απαραίτητη για την απόκτηση των χρονικών φραγμάτων που θέλουμε → **Cascading Cuts**
Όταν ένας κόμβος x, που είναι ρίζα, γίνεται παιδί ενός άλλου κόμβου y μέσω κάποιας ένωσης (linking), τότε αν ο x χάσει δύο από τα παιδιά του (μέσω delete ή decrease key), κόβουμε και την ακμή ανάμεσα στο x και στον γονέα του, y, και ο x γίνεται μία νέα ρίζα (επιτυγχάνεται μέσω marking).
Σημείωση: Μία εκτέλεση Decrease key μπορεί να οδηγήσει σε μεγάλο αριθμό από cascading cuts.

Crucial Properties of F-Heaps

- Κάθε δέντρο στον $F - heap$ έχει μέγεθος τουλάχιστον εκθετικό ως προς το βαθμό $r(x)$ της ρίζας του, x.
- Το πλήθος των cascading cuts μέσα σε μία σειρά εκτελέσεων λειτουργιών σε έναν F-Heap φράσσεται από το πλήθος των εκτελέσεων Delete και Decrease key.

Η λειτουργία Cascading Cuts έχει εισαχθεί ακριβώς για την εξυπηρέτηση της πρώτης ιδιότητας.

Proof Of First Property

Λήμμα

Έστω x κόμβος του F-Heap. Κατατάσσουμε τα παιδιά του x στη σειρά με την οποία ενώθηκαν με τον x. Τότε το i-οστό παιδί του x έχει τουλάχιστον $i - 2$ παιδιά.

Proof: Έστω y το i-οστό παιδί του x. Πριν την ένωσή του με τον x, ο x είχε τουλάχιστον $i - 1$ παιδιά. Αφού ο x και ο y είχαν τον ίδιο βαθμό πριν την ένωση (διαφορετικά δε θα είχαν ενωθεί), τότε και ο y είχε τουλάχιστον $i - 1$ παιδιά πριν την ένωση. Μετά την ένωση, ο y μπορεί να έχει χάσει το πολύ ένα παιδί, άρα να έχει τουλάχιστον $i - 2$ (διαφορετικά θα είχε γίνει cascading cut και ο y δε θα ήταν πια παιδί του x).

Proof Of First Property

Συμπέρασμα

Ένας κόμβος βαθμού k στον F-Heap έχει τουλάχιστον $F_{k+2} \geq \phi^k$ απογόνους, συμπεριλαμβανόμενου του εαυτού του. (F_k ο k-οστός όρος Fibonacci και $\phi = \frac{1+\sqrt{5}}{2}$ ο golden ratio)

Proof: Έστω S_k ο ελάχιστος πιθανός αριθμός απογόνων ενός κόμβου με βαθμό k ($S_0 = 1, S_1 = 2$). Το λήμμα που δείξαμε μας δίνει $S_k \geq \sum_{i=0}^{k-2} S_i + 2$ για $k \geq 2$. Οι αριθμοί Fibonacci ικανοποιούν $F_{k+2} = \sum_{i=2}^{k} F_i + 2$ για $k \geq 2$, και τελικά με επαγωγή $S_k \geq F_{k+2}$. Η ανισότητα $F_{k+2} \geq \phi^k$ είναι γνωστή.

F-Heaps to implement Dijkstra's shortest path algorithm

- Έστω $G = (V(G), E(G))$ένα κατευθυνόμενο γράφημα, s μία κορυφή του που διακρίνεται ως η πηγή, και έστω ότι κάθε ακμή (v, w) έχει ένα μη αρνητικό μήκος $l(v, w)$.
- $|V(G)| = n$, $|E(G)| = m$ και υποθέτουμε ότι το γράφημα είναι συνεκτικό, άρα $m \geq n - 1$.
- Το μήκος ενός μονοπατιού στο G είναι το άθροισμα των μηκών των ακμών του.
- Η απόσταση μιας κορυφής v από μία w είναι το μονοπάτι ανάμεσα τους με το ελάχιστο μήκος.
- Το single source shortest path problem είναι αυτό της αναζήτησης του μονοπατιού ελαχίστου μήκους από την s ως την v, για κάθε κορυφή v του G.

F-Heaps to implement Dijkstra's shortest path algorithm

- Χρησιμοποιούμε τη συνάρτηση απόστασης $d : V(G) \to \mathbb{R}$ με $d(v) < \infty$ όταν υπάρχει μονοπάτι από την s στην v με μήκος $d(v)$.
- Κατά τη διάρκεια του αλγορίθμου κάθε κορυφή έχει μία κατάσταση: labeled, unlabeled και scanned.
- Στο πρώτο βήμα $d(s) = 0$ και $d(v) = \infty$ για $v \neq s$, η s είναι labeled και όλες οι υπόλοιπες κορυφές unlabeled.
- Ο ακόλουθος αλγόριθμος επαναλαμβάνετε ώσπου να μην υπάρχουν labeled κορυφές:

F-Heaps to implement Dijkstra's shortest path algorithm

Algorithm

1. Βρες μία labeled κορυφή v με $d(v)$ minimum.
2. Μετάτρεψε τη v από labeled σε scanned.
3. Για κάθε ακμή (v, w) με $d(v) + l(v, w) < d(w)$, κάνε $d(w) = d(v) + l(v, w)$.
4. Κάνε την w labeled.

F-Heaps to implement Dijkstra's shortest path algorithm

Για να υλοποιήσουμε τον αλγόριθμο χρησιμοποιώντας τους F-Heaps:
Αποθηκεύουμε σε ένα σωρό το σύνολο των labeled κορυφών, και για κάθε v στο σύνολο αυτό έχουμε ως κλειδί το $d(v)$.

- 1 Make heap, n insert, n delete min και το πολύ m decrease key.
- Άρα απαιτούμενος χρόνος $O(nlogn + m)$, βελτιωμένος από $O(mlog_{\frac{m}{n+2}} n)$ (φράγμα του Johnson βασισμένο σε μία υλοποίηση με σωρούς).

F-Heaps to implement minimum spanning trees algorithms

Έστω μη κατευθυνόμενο γράφημα G με n κορυφές, m ακμές και ένα κόστος $c(v, w)$ για κάθε ακμή (v, w).
Minimum Spanning Tree αλγόριθμος: Greedy approach
Το δάσος με τις ακμές που θα έχουν επιλεγεί μέχρι στιγμής θα ανήκει στο minimum spanning tree.
Αρχικοποιούμε ένα δάσος που περιλαμβάνει κάθε μία από τις n κορυφές ως δέντρο.Επαναλαμβάνουμε $n - 1$ φορές:

Algorithm

1. Διάλεξε ένα δέντρο T στο δάσος.
2. Βρες την ακμή με το ελάχιστο κόστος που έχει ακριβώς το ένα άκρο της στο T και πρόσθεσε τη στο δάσος.

F-Heaps to implement minimum spanning trees algorithms

Για να υλοποιήσουμε τον αλγόριθμο χρησιμοποιώντας τους F-Heaps: Αποθηκεύουμε τις κορυφές με πεπερασμένο κλειδί key σε έναν F-Heap.
Ξεκινάμε με μία αυθαίρετη κορυφή-πηγή με key(s)=0 και key(v)=∞ για κάθε κορυφή $v \neg s$. Επαναλαμβάνουμε μέχρι να μην υπάρχει κορυφή με πεπερασμένο κλειδί:

1. Διαλέγουμε μία κορυφή v με ελάχιστο key(v) ανάμεσα στις κορυφές πεπερασμένου βαθμού.
2. Μετατρέπουμε το key(v) σε $-\infty$.
3. Για κάθε ακμή (v, w) με $c(v, w) <$ key(w) μετατρέπουμε το key(w) σε $c(v, w)$ και ορίζουμε $e(w) = (v, w)$.

Το σύνολο των e(w) με $w \neq s$ καθορίζουν το minimum spanning tree.

F-Heaps to implement minimum spanning trees algorithms

Ο αλγόριθμος θα εκτελέσει n φορές τη λετουργία Delete min σε χρόνο $O(nlogn)$, και κάποιες λειτουργίες πάνω στις ακμές σε χρόνο $O(m)$.
Συνολικά ο απαιτούμενος χρόνος είναι $O(nlogn + m)$, βελτιωμένος από τον μέχρι τότε καλύτερο.
Μπορούμε να πετύχουμε και καλύτερη βελτίωση χρησιμοποιώντας τους F-Heaps, με έναν πιο περίπλοκο στην περιγραφή αλγόριθμο. Συγκεκριμένα καταφέρνουμε ένα φράγμα $O(m\beta(m,n))$, όπου
$\beta(m,n) \leq min\{i|log^i n \leq \frac{2m}{n}\} + 1.$

Improvements

- Τα φράγματα με τη χρήση F-Heaps είναι ασυμπτωτικές βελτιώσεις για γραφήματα με ενδιάμεση πυκνότητα ($n << m << n^2$).
- Σημαντικές βελτιώσεις στους αλγορίθμους που περιλαμβάνουν εύρεση μονοπατιού ελαχίστου μήκους ως υπορουτίνα: π.χ. all pairs shortest path problem (σε χρόνο $O(n^2 logn + nm)$), the assignment problem (bipartite weighted matching) (επίσης σε χρόνο $O(n^2 logn + nm)$) κ.α.
- Σημαντική βελτίωση στο minimum spanning tree problem (σε χρόνο $O(m\beta(m, n))$).

THE END

Σας ευχαριστώ πολύ για την προσοχή σας!!! ★