1 Εισαγωγή

Το 4ο πρόβλημα του διαγωνισμού ζητούσε μία δομή δεδομένων που να μπορεί γρήγορα να ανανεώνεται (αλλάζει έναν αριθμό) και να απαντάει σε range queries (άθροισμα).
Στην περίπτωση που το range query αναφερόταν πάντα στην τελευταία έκδοση της δομής μας, η λύση θα ήταν η πιο τυπική εφαρμογή ενός segment tree (εάν δεν ξέρετε τι είναι ένα segment tree δείτε εδώ και κατόπιν παίξτε λίγο με αυτή την εφαρμογή ώστε να καταλάβετε οπτικά πώς λειτουργεί).

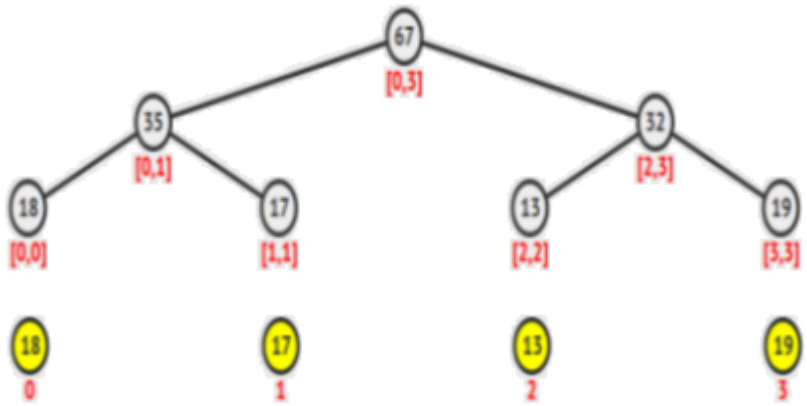
Το πρόβλημα που καλούμαστε να λύσουμε λέγεται Persistency.
Απαιτείται η τροποποίηση της δομής δεδομένων μας ώστε να θυμάται παλιότερες εκδόσεις της.

Εξαιρετική αναφορά στα persistent segment trees, με επεξήγηση δύσκολότερων προβλημάτων, μπορείται να βρείτε εδώ.

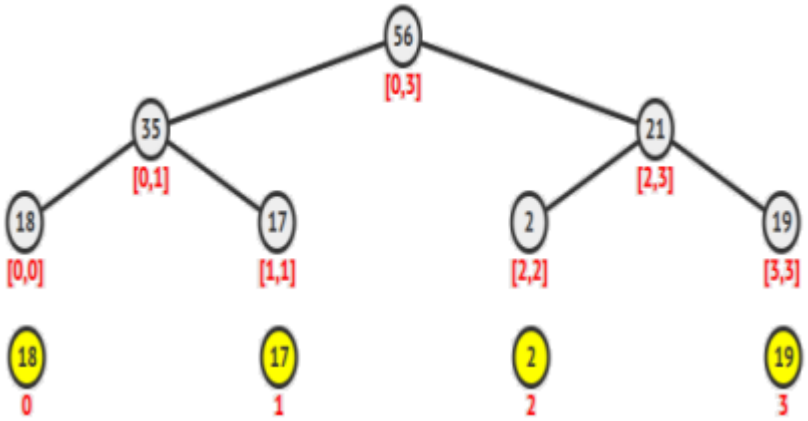
Όποιος θέλει να μάθει ακόμη περισσότερα για αυτή την τεχνική μπορεί να δει το σχετικό βίντεο από το MIT.

2 Απλές σκέψεις

Το πιο απλό πράγμα που μπορεί να σκεφτεί κανείς είναι να κάνει ένα ξεχωριστό segment tree κάθε φορά που γίνεται μια ανανέωση. Έτσι αν αρχικά είχαμε :

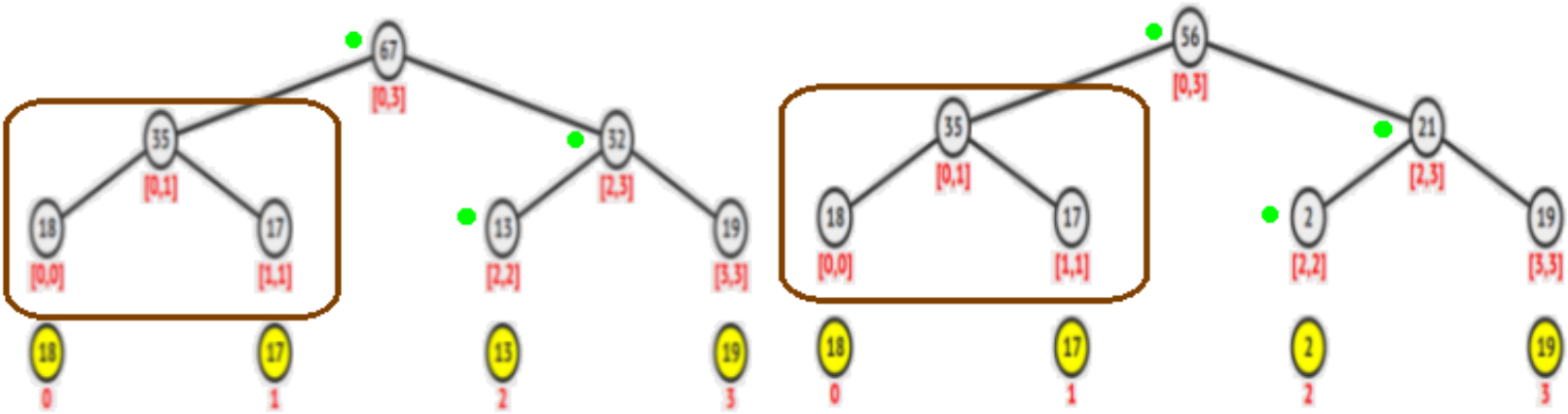


και κατόπιν κάναμε την ανανέωση (Θέση 2 - τιμή 2), θα παίρναμε :

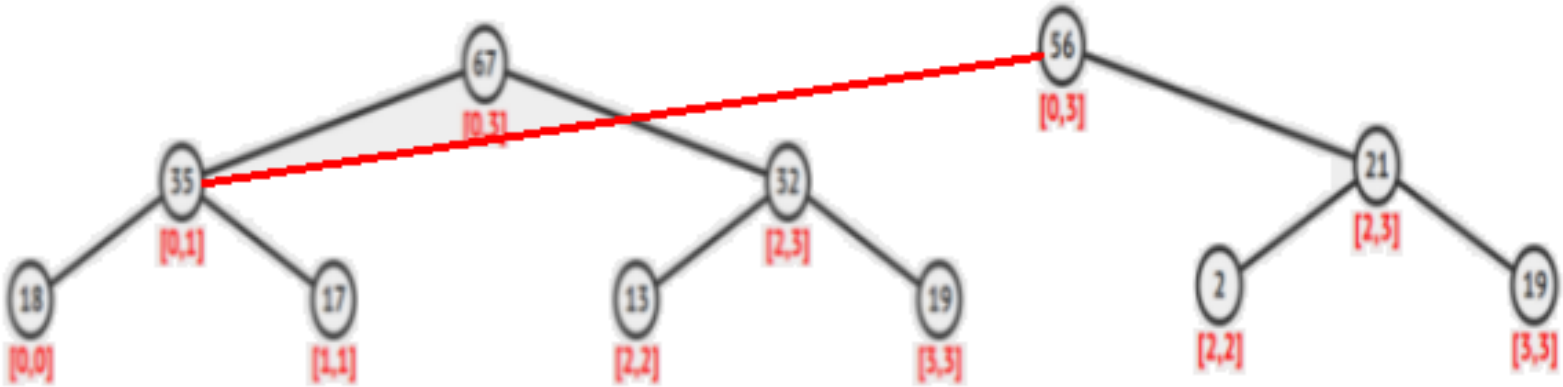


Προφανώς τώρα τα ερωτήματα απαντιούνται πολύ γρήγορα. Όμως η δουλειά που κάνουμε για να δημιουργήσουμε ένα segment tree από την αρχή είναι πολύ μεγάλη. Το ερώτημα είναι, πώς θα κάνουμε λιγότερη δουλειά για την δημιουργία του επόμενου segment tree; Ποιές ομοιότητες με το προηγούμενο μπορώ να εκμεταλλευτώ ώστε να μη γράφω πολλές φορές το ίδιο πράγμα;

3 Πολύ λίγοι κόμβοι αλλάζουν

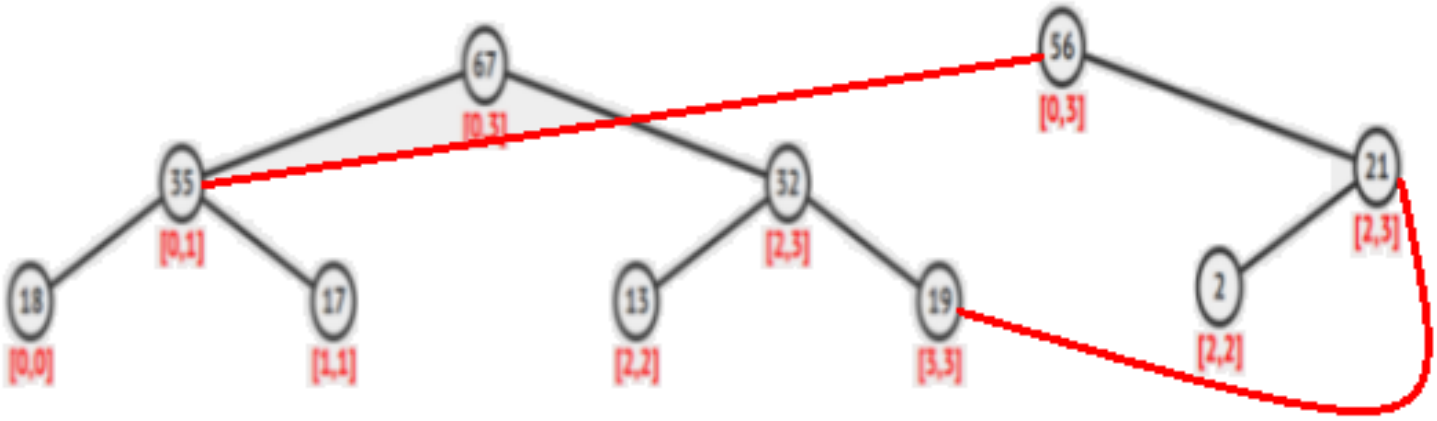


Παραπάνω βλέπουμε τα 2 segment trees δίπλα δίπλα. Με πράσινο είναι σημειωμένοι οι κόμβοι από την ρίζα μέχρι τον ανανεωμένο κόμβο 2.
Αρκεί μία και μόνο παρατήρηση για να απλοποιηθεί δραματικά το πρόβλημα. Μετά τη ρίζα, στρίψαμε δεξιά για να πάμε στη θέση 2, την οποία ανανεώσαμε. Συνεπώς το υπόδεντρο που βρίσκεται αριστερά της ρίζας, σημειωμένο σε καφέ πλαίσιο, παραμένει όπως έχει (ποτέ δε το επισκεφθήκαμε, άρα ποτέ δε το αλλάξαμε). Αντί λοιπόν να το ξαναδημιουργήσω, θα φερθώ έξυπνα :



Τι συνέβη παραπάνω; Το παλιό segment tree έμεινε όπως ήταν. Για το καινούριο όμως δε δημιούργησα καθόλου το αριστερό υπόδεντρο, χρειάστηκε απλά να προσθέσω μία σύνδεση, από την ρίζα προς το αριστερό της υπόδεντρο στο παλιό segment tree.

Δεν έχω κανένα λόγο να σταματήσω. Με την ίδια λογική, ο κόμβος με την τιμή 19 (κι ό,τι θα υπήρχε από κάτω του αν το segment tree είχε περισσότερα από 4 στοιχεία) μένει απαράλλακτος.



4 Πρόχειρη ανάλυση πολυπλοκότητας

Οι μόνοι κόμβοι που θα δημιουργηθούν είναι αυτοί που βρίσκονται στο μονοπάτι από τη ρίζα προς τον ανανεωμένο κόμβο. Το πλήθος τους είναι πολύ μικρό, όσο και το ύψος του segment tree, το οποίο γνωρίζουμε ότι είναι μόλις logN.

Χρονικά η απάντηση σε ερώτημα παραμένει $O(logN)$ αφού, ο κώδικάς της μένει ολόιδιος.
Η ανανέωση κι αυτή παραμένει $O(logN)$ αφού οι μόνες αλλαγές που χρειάζεται είναι η προσθήκη μίας ακμής κάθε φορά στο κατάλληλο υπόδεντρο του παλιού segment tree (περισσότερες λεπτομέρειες στον κώδικα).
Ο χώρος που θα χρειαστούμε είναι ελαφρώς περισσότερος. Από $O(N)$ που χρειαζόμασταν για ένα segment tree θα χρειαστούμε $O(NlogN)$ για τους $logN$ κόμβους που θα προσθέτει κάθε ένα από τα N segment trees που θα δημιουργηθούνε.