ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΠΟΛΥΠΛΟΚΟΤΗΤΑ

1η Σειρά Ασκήσεων

Ακαδημαϊκό έτος 2021-2022

Κοκκινάκης Παναγιώτης – Α.Μ.: 03118115

Άσκηση 1: Αναδρομικές Σχέσεις

1. $T(n) = 4\ T(n/2) + \Theta(n^2 \log n)$

Από Master Theorem ($T(n)=aT(n/b)+\Theta(n^k \log^p n)$) έχουμε ότι a=4, b=2, k=2, p=1

Είμαστε στη $2^η$ περίπτωση ($a=b^k$) οπότε εφόσον p>-1 ισχύει $T(n)=\Theta(n^{\log b(a)} \log^{p+1} n)=\Theta(n^2 \log^2 n)$

2. $T(n) = 5\ T(n/2) + \Theta(n^2 \log n)$

Από Master Theorem ($T(n)=aT(n/b)+\Theta(n^k \log^p n)$) έχουμε ότι a=5, b=2, k=2, p=1

Είμαστε στην $1^η$ περίπτωση ($a>b^k$) άρα $T(n)=\Theta(n^{\log b(a)})=\Theta(n^{\log 2(5)})$.

3. $T(n) = T(n/4) + T(n/2) + \Theta(n)$

$n/4+n/2<n$, οπότε υποπτευόμαστε $T(n)=\Theta(n)$. Κάνουμε επαγωγή:

Για n=1 =>$T(1)=\Theta(1)$ δηλαδή η υπόθεση μας επαληθεύεται. Έστω ότι έχουμε $n\geq 2$. Θεωρούμε ότι ισχύει για $1\leq k<n$ η υπόθεση οπότε θα ισχύει $T(n)=\Theta(n/4)+\Theta(n/2)+\Theta(n)=\Theta(n)$. Επομένως από επαγωγή έχουμε ότι $T(n)=\Theta(n)$.

4. $T(n)=2\ T(n/4) + T(n/2) + \Theta(n)$

$2n/4+n/2=n$, οπότε υποπτευόμαστε ότι $T(n)=\Theta(n\log n)$. Με δέντρο αναδρομής έχουμε:

Για Ύψος: $\Theta(\log n)$ (Το ύψος του δέντρου είναι $\log_3 n$), #κορυφών: $\Theta(n)$, Χρόνος/επίπεδο: $\Theta(n)$

Επομένως ο συνολικός χρόνος είναι $\Theta(n\log n)$.

5. $T(n)=T(n^{1/2}) + \Theta(\log n)$

Αντικαθιστούμε $n = 2^m$ οπότε η αναδρομική μας σχέση γίνεται:

$T(2^m)=T(2^{m/2})+\Theta(m)$

Αντικαθιστούμε $T(2^m)=S(m)$ οπότε η αναδρομική μας σχέση γίνεται:

$S(m)=S(m/2)+\Theta(m)$

Από Μ.Τ. (Ειδική μορφή-$3^{η}$ περίπτωση) => $S(m)=Θ(m)$

$n=2^m$=>$m=\log_2 n$ οπότε $S(m)=T(2^m)=Θ(m)$=> $T(n)=Θ(\log n)$

6. $T(n) = T(n/4) + Θ(\sqrt{n})$

Από Master Theorem ($T(n)=aT(n/b)+Θ(n^k \log^p n)$) έχουμε ότι a=1, b=3, k=1/2, p=0

Είμαστε στην $3^{η}$ περίπτωση ($a<b^k$) και $p>=0$ άρα $T(n)= Θ(n^k \log^p n) = Θ(\sqrt{n})$

Άσκηση 2: Προθεματική Ταξινόμηση

Α)

- Διατρέχουμε αρχικά όλο τον πίνακα και βρίσκουμε το μέγιστο στοιχείο του, έστω ότι είναι το i-οστό.
- Κάνουμε προθεματική περιστροφή στον υποπίνακα του Α από το 1 μέχρι το i ώστε να φέρουμε το μέγιστο στοιχείο του πίνακα στην πρώτη θέση.
- Ξανακάνουμε προθεματική περιστροφή σε ολόκληρο τον πίνακα, ώστε το μέγιστο στοιχείο να πάει στην τελευταία θέση.
- Επαναλαμβάνουμε τα 3 πρώτα βήματα για τον υποπίνακα του Α από το 1 μέχρι το n-1 (για να μην περιλαμβάνεται το ταξινομημένο μέγιστο στοιχείο).

Για κάθε στοιχείο που ταξινομούμε χρειάζονται 2 προθεματικές περιστροφές, επομένως για να ταξινομηθεί ολόκληρος πίνακα n στοιχείων απαιτούνται ακριβώς 2n προθεματικές περιστροφές.

Μπορούμε να μειώσουμε τις προθεματικές περιστροφές εάν αντί για όλα τα n στοιχεία ταξινομήσουμε τα n-1 αφού το τελευταίο θα είναι στην πρώτη θέση εφόσον όλα τα υπόλοιπα είναι στις κατάλληλες θέσεις τους. Έτσι κάνουμε 2(n-1) = 2n-2 προθεματικές περιστροφές. Εάν αντ' αυτού ταξινομήσουμε n-2 στοιχεία διακρίνουμε δύο περιπτώσεις. Είτε τα δύο στοιχεία που μένουν είναι στις σωστές ταξινομημένες τους θέσεις οπότε η διαδικασία ταξινόμησης ολοκληρώθηκε σε 2(n-2) = 2n-4 προθεματικές περιστροφές, είτε δεν είναι οπότε με μία προθεματική περιστροφή στα δύο πρώτα μη ταξινομημένα στοιχεία τα βάζουμε στην σωστή τους θέση. Επομένως με το πολύ 2n-3 προθεματικές περιστροφές μπορούμε να ταξινομήσουμε τα n στοιχεία.

Β)

Στον γενικό αλγόριθμο του προηγούμενου ερωτήματος δουλεύουμε με τις απόλυτες τιμές των αριθμών στον πίνακα για να βρίσκουμε κάθε φορά το μέγιστο στοιχείο. Όταν το μέγιστο του υποπίνακα με τον οποίο δουλεύουμε είναι στην αριστερότερη θέση, ελέγχουμε το πρόσημο του. Αν αυτό είναι θετικό τότε πραγματοποιούμε μία έξτρα προθεματική περιστροφή στο στοιχείο αυτό μόνο για να του αλλάξουμε το πρόσημο και συνεχίζουμε κανονικά τον αλγόριθμο. Έτσι θα

χρειαστούμε το πολύ 3n προθεματικές περιστροφές, αφού απαιτούνται οι 2 του κανονικού αλγορίθμου + μια για να αλλάξουμε το πρόσημα για κάθε ένα από τα n στοιχεία του πίνακα.

Γ)

1. Διατρέχουμε τον πίνακα και ελέγχουμε τα στοιχεία που δεν ανήκουν ήδη σε συμβατό ζεύγος. Διακρίνουμε τις εξής περιπτώσεις:

- Αν όλα τα στοιχεία είναι αρνητικά: Εφόσον (από υπόθεση) δεν έχουμε τον πίνακα [-1,-2…,-n], θα υπάρχει στοιχείο χ που είναι δεξιότερα του αμέσως προηγούμενου του χ-1. Επομένως με δύο προθεματικές περιστροφές μπορούμε τα φέρουμε το στοιχείο χ αυτό στην αρχή του πίνακα χωρίς να πειράξουμε το στοιχείο –(χ-1) που είναι δεξιότερα, και στη συνέχεια με μία ακόμη προθεματική περιστροφή να φέρουμε το -χ δίπλα στο –(χ-1).
- Αν υπάρχει θετικό στοιχείο: βρίσκουμε το μεγαλύτερο θετικό στοιχείο χ.

Αν αυτό είναι το n, με δύο προθεματικές περιστροφές το φέρνουμε στα δεξιά του πίνακα επομένως δημιουργείται συμβατό ζεύγος αφού η μετακίνηση του μέγιστου στοιχείου είναι συμβατό ζεύγος.

Αλλιώς υπάρχουν δύο υποπεριπτώσεις εφόσον το στοιχείο χ+1 θα έχει αρνητικό πρόσημο:

1. Εάν το στοιχείο αυτό είναι δεξιά του χ κάνουμε προθεματική περιστροφή στον πίνακα μέχρι και το –(χ+1) και δημιουργούμε πίνακα της μορφής [χ+1,…,-χ…] (το χ αλλάζει πρόσημο). Με μία ακόμη προθεματική περιστροφή μέχρι και το προηγούμενο στοιχείο του -χ δημιουργούμε το συμβατό ζεύγος [–(χ+1),-χ].
2. Εάν το στοιχείο αυτό είναι αριστερά του χ, κάνουμε μία προθεματική περιστροφή στον πίνακα μέχρι και το χ και δημιουργείται πίνακας της μορφής [-χ,…χ+1…]. Με μία ακόμη προθεματική περιστροφή μέχρι και το προηγούμενο στοιχείο του χ+1, δημιουργούμε το συμβατό ζεύγος [χ,χ+1].

Επομένως σε κάθε περίπτωση, βλέπουμε ότι απαιτούνται το πολύ 2 προθεματικές περιστροφές στον πίνακα για να δημιουργηθεί συμβατό ζεύγος.

2) Δημιουργούμε αρχικά n/2 συμβατά ζεύγη στον πίνακα με n προθεματικές περιστροφές, ώστε κάθε στοιχείο να ανήκει σε ένα.

Αντικαθιστούμε αυτά τα ζεύγη με αριθμούς που είναι ίσοι με το μέσο όρο των δύο στοιχείων τους (με πρόσημο). Έχουμε έτσι έναν νέο πίνακα που χρειάζεται ταξινόμηση. Επαναλαμβάνουμε τη διαδικασία και δημιουργούμε με n/2 προθεματικές περιστροφές n/4 νέα συμβατά ζεύγη, τα οποία και πάλι αντικαθιστούμε με προσημασμένα στοιχεία. Συνεχίζουμε μέχρι να έχουμε εν τέλει ένα συμβατό ζεύγος το οποίο έχει προκύψει τελικά από n+n/2+n/4+…+1=2n προθεματικές περιστροφές και αντικαθιστούμε αναδρομικά τα ζεύγη με τα στοιχεία από τα οποία προέκυψαν. Έχει έτσι ταξινομηθεί ο πίνακας με 2n προθεματικές περιστροφές συνολικά.

Άσκηση 3: Υπολογισμός Κυρίαρχων Θέσεων

Θα δουλέψουμε με στοίβα στην οποία θα αποθηκεύουμε την τιμή στοιχείων καθώς και τη θέση τους στον πίνακα. Δημιουργούμε αρχικά έναν νέο πίνακα n θέσεων και ξεκινάμε τη διάσχιση του δοσμένου πίνακα.

- Κάνουμε push το ζεύγος $(0,\infty)$
- Προχωράμε στο επόμενο στοιχείο του πίνακα και το συγκρίνουμε με την τιμή του στοιχείου την κορυφή του πίνακα. Διακρίνουμε τις εξής υποπεριπτώσεις:
  - Αν η κορυφή της στοίβας είναι μεγαλύτερη από το στοιχείο του πίνακα, αποθηκεύουμε στον καινούριο πίνακα τη θέση της κορυφής και κάνουμε push στη στοίβα το στοιχείο του αρχικού πίνακα μαζί με τη θέση του. Στη συνέχεια επαναλαμβάνουμε το βήμα αυτό για το επόμενο στοιχείο του πίνακα.
  - Αν η κορυφή της στοίβας είναι μικρότερη από το στοιχείο του πίνακα, κάνουμε pop στη στοίβα μέχρι να βρούμε στοιχείο που να είναι μεγαλύτερο, οπότε και δουλεύουμε με την παραπάνω υποπερίπτωση.

Η πολυπλοκότητα του αλγορίθμου είναι O(n) καθώς κάθε στοιχείο της στοίβας θα γίνει Push/Pop μία φορά επομένως έχουμε το πολύ 2n βήματα.

Άσκηση 4: Φόρτιση Ηλεκτρικών Αυτοκινήτων

Παρατηρούμε ότι εφόσον ο αριθμός των αυτοκινήτων που φτάνουν σε μια χρονική βαθμίδα είναι μικρότερος ή ίσος από τον αριθμό των φορτιστών τότε δεν δημιουργείται συμφόρηση και επαρκεί ο χρόνος όποια και αν είναι η καθυστέρηση d. Επομένως πρέπει να βρούμε τη χρονική στιγμή που έχουμε τη μέγιστη εισροή αυτοκινήτων και να φροντίσουμε από αυτό το σημείο να κυλήσει ομαλά η φόρτιση. Οπότε:

- Δημιουργούμε αρχικά έναν πίνακα T θέσεων, τον οποίο γεμίζουμε διατρέχοντας τον πίνακα αφίξεων και αυξάνοντας κατά 1 την αντίστοιχη θέση για κάθε άφιξη σε μία χρονική στιγμή. Έχουμε με αυτόν τον τρόπο έναν πίνακα των T χρονικών βαθμίδων με τον αριθμό αυτοκινήτων που φτάνουν σε κάθε βαθμίδα.
- Διατρέχουμε τον πίνακα αυτόν και βρίσκουμε τη μέγιστη τιμή του καθώς και τη θέση της. Αυτή η τιμή αντιπροσωπεύει τον μέγιστο αριθμό αυτοκινήτων που φτάνουν σε μία χρονική στιγμή.
- Ελέγχουμε την μέγιστη τιμή και βρίσκουμε k τέτοιο ώστε να ισχύει: $(k-1)d < max <= kd$. Βρίσκουμε έτσι τους ελάχιστους φορτιστές που θα χρειάζονταν ώστε να ικανοποιείται η συνθήκη καθυστέρησης για τα αμάξια που έφτασαν αυτή τη χρονική στιγμή.

- Ξεκινάμε τον έλεγχο μας από την αρχή του πίνακα και προσομοιώνουμε τη διαδικασία φόρτισης με k φορτιστές, έτσι ώστε να βλέπουμε πόσα αμάξια βρίσκονται σε αναμονή κάθε φορά. Αφαιρούμε οπότε σε κάθε χρονική βαθμίδα k, αυτοκίνητα και προσθέτουμε όσα έφτασαν. Προσέχουμε και πάλι ώστε να ισχύει (k-1)d < συνολικά αμάξια <= kd. Ένα αυτό πάψει να ισχύει βρίσκουμε το νέο k', και συνεχίζουμε τη διάσχιση και την προσομοίωση με αυτό.
- Αφού φτάσουμε στο τέλος, συγκρίνουμε το k με το οποίο ξεκινήσαμε με αυτό που τελειώσαμε. Αν ταυτίζονται τότε ολοκληρώνεται ο αλγόριθμος και το k που βρήκαμε είναι ο ελάχιστος αριθμός φορτιστών που ζητάμε. Σε διαφορετική περίπτωση επαναλαμβάνουμε το προηγούμενο βήμα με δυαδική αναζήτηση ώστε να βρούμε το ελάχιστο k για το οποίο η διαδικασία ολοκληρώνεται χωρίς αλλαγές. Γνωρίζουμε ότι βρίσκεται στο διάστημα αυτό καθώς το αρχικό μας k είναι ο ελάχιστος αριθμός φορτιστών που απαιτούνται αν γνωρίζουμε τον μέγιστο αριθμό αφίξεων σε μία χρονική στιγμή, ενώ το k στο οποίο καταλήγουμε είναι σίγουρα επαρκές αφού ολοκληρώνεται ο πίνακας και ικανοποιείται η απαίτηση παρότι ξεκινάμε με λιγότερους φορτιστές. Συνεχίζουμε τη δυαδική αναζήτηση μέχρι να ολοκληρωθεί μία διάσχιση του πίνακα χωρίς να αλλάξει το k.

Η πολυπλοκότητα του παραπάνω αλγορίθμου είναι: n + Tlogn

- ❖ n λόγω της δημιουργίας του πίνακα μας, για τον οποίο διαβάζουμε τον χρόνο άφιξης των n αυτοκινήτων.
- ❖ T για κάθε διάσχιση του πίνακα που δημιουργήσαμε με τους ελέγχους που κάνουμε.
- ❖ logn για τη δυαδική αναζήτηση. (Στην πραγματικότητα η δυαδική μας αναζήτηση γίνεται σε πολύ λιγότερα από n στοιχεία αφού γίνεται το πολύ σε A_{max} (ο αριθμός μέγιστων αφίξεων σε μία μέρα) – το αρχικό k.

Επομένως αναλόγως τα n και T, έχουμε είτε O(n) είτε O(Tlogn).

Άσκηση 5: Επιλογή

Α)

Θα χρησιμοποιήσουμε δυαδική αναζήτηση στην συνάρτηση κατανομής Fs. Ο αλγόριθμος έχει τα εξής βήματα:

- Θεωρούμε αρχικά ότι Fs(0)=0.
- Αν έχουμε M στοιχεία κάνουμε δυαδική αναζήτηση στην συνάρτηση κατανομής οπότε ελέγχουμε την τιμή του Fs(M/2). Εάν η τιμή αυτού είναι μικρότερη από το k που αναζητάμε συνεχίζουμε την δυαδική αναζήτηση με το Fs(M/4) αλλιώς με το Fs(3M/4) κοκ.

- Εάν πετύχουμε ακριβώς για κάποιο i να ισχύει Fs(i)=k, ελέγχουμε πάλι με δυαδική αναζήτηση τα προηγούμενα στοιχεία για να βρούμε εάν έχουμε επαναλήψεις της τιμής του k. Για το μικρότερο i' που βρίσκουμε ότι Fs(i')=k ισχύει η ζητούμενη συνθήκη δηλαδή ότι είναι το k-οστό μικρότερο στοιχείο.
- Εάν με τη δυαδική αναζήτηση δεν βρεθεί το ίδιο το στοιχείο k και καταλήξουμε σε δύο διαδοχικά στοιχεία της συνάρτησης κατανομής να έχουμε μία τιμή μικρότερη και μία τιμή μεγαλύτερη του k αντίστοιχα, τότε η ζητούμενη απάντηση είναι το πρώτο i που μας δίνει την αμέσως μεγαλύτερη τιμή. (π.χ. για $Fs(i_1)=1$, $Fs(i_2)=3$ και $k=2$ με $i_1=i_2-1$ θα έχουμε ότι το ζητούμενο k-οστό στοιχείο είναι i_2).

Η πολυπλοκότητα του αλγορίθμου είναι O(logM) καθώς κάνουμε δυαδική αναζήτηση σε M στοιχεία, δηλαδή στις τιμές της συνάρτησης Fs (αν θεωρήσουμε ότι οι κλήσεις της συνάρτησης F_S έχουν πολυπλοκότητα O(1)).

B)

Έχουμε ότι τα στοιχεία του S είναι οι θετικές διαφορές ζευγών στοιχείων του πίνακα A, n στοιχείων με μέγιστο το M. Χρησιμοποιήσουμε τον αλγόριθμο του προηγούμενου ερωτήματος, αφού πρόκειται για διαφορές οπότε το μέγιστο στοιχείο του S θα είναι το πολύ M, όμως θα πρέπει πρώτα να βρούμε την συνάρτηση F_S ώστε να είναι αυτό εφικτό.

Από το προηγούμενο ερώτημα γνωρίζουμε ότι η συνάρτηση Fs θα κληθεί logM φορές. Αρχικά ταξινομούμε τον πίνακα A με quicksort. Κάθε φορά που θα κληθεί η Fs με είσοδο l ελέγχουμε ένα-ένα τα n στοιχεία του ταξινομημένου A. Για κάθε ένα αναζητούμε με δυαδική αναζήτηση μέσα στον ταξινομημένο A το πρώτο στοιχείο για το οποίο ισχύει ότι η διαφορά του με το στοιχείο το οποίο ελέγχουμε είναι ίση με l. Για το στοιχείο αυτό προσθέτουμε σε counter τη διαφορά των δύο δεικτών (του στοιχείου που ελέγχουμε και του στοιχείου με βρήκαμε με τη δυαδική αναζήτηση), αφού η διαφορά και όλων των μικρότερων στοιχείων με το αρχικό δε θα ξεπερνά το l. Μόλις τελειώσουμε αυτή τη διαδικασία για κάθε στοιχείο του A η τιμή του counter θα είναι η τιμή της συνάρτησης Fs για είσοδο l. Η εύρεση αυτής της τιμής χρειάζεται n δυαδικές αναζητήσεις σε n στοιχεία επομένως έχει πολυπλοκότητα O(nlogn).

Εφόσον καλούμε τη συνάρτηση Fs logM φορές η συνολική πολυπλοκότητα της διαδικασίας είναι O(n*logn*logM).