$heap\_size$	Βήμα 1	Βήμα 2
5	[1 7 5 2 | 7]	[7 2 5 1 | 7]
4	[1 2 5 | 7 7]	[5 2 1 | 7 7]
3	[1 2 | 5 7 7]	[2 1 | 5 7 7]
2	[1 | 2 5 7 7]	[1 | 2 5 7 7]

Πίνακας 4.2: Εκτέλεση Αλγόριθμου 30 - Παράδειγμα 10

4.3 Αναδρομή ουράς

Όταν ένας αλγόριθμος υλοποιείται μέσω αναδρομικής συνάρτησης, σε κάθε κλήση της γίνεται δέσμευση ενός κομματιού της μνήμης προκειμένου να χρησιμοποιηθεί από τις τοπικές μεταβλητές, τις τιμές που επιστρέφει η συνάρτηση, τις τιμές των παραμέτρων της συνάρτησης και τη διεύθυνση στην οποία πρέπει να επιστρέψει ο έλεγχος όταν ολοκληρωθεί η εκτέλεση της συγκεκριμένης κλήσης. Επομένως όσο πιο βαθιά είναι η αναδρομή τόσο περισσότερος χώρος δεσμεύεται. Αυτό μπορεί εν' γένει να αποτελέσει πηγή προβλημάτων αφού η μνήμη είναι πεπερασμένος πόρος.
Υπάρχει όμως μία κατηγορία αναδρομικών αλγορίθμων στις οποίες ο χώρος που δεσμεύεται είναι σταθερός και δεν εξαρτάται από το πλήθος των αναδρομικών κλήσεων· αυτό συμβαίνει στην περίπτωση που η αναδρομική κλήση είναι η τελευταία λειτουργία που εκτελείται - δεν υπάρχει άλλη εντολή η οποία να εκτελείται μετά την αναδρομική κλήση. Στην περίπτωση αυτή έχουμε τη λεγόμενη *αναδρομή ουράς (tail recursion)*. Η ιδιότητα αυτή είναι επιθυμητή και επιτρέπει στους σύγχρονους μεταγλωττιστές να επιτύχουν διαφόρων ειδών βελτιστοποιήσεις στον παραγόμενο κώδικα.

Παραδείγματα διαδικασιών που δεν έχουν την ιδιότητα της αναδρομής ουράς αποτελούν οι Αλγόριθμοι 23, 24 - για τον πρώτο μετά την αναδρομική κλήση απομένει να εκτελεστεί μία πράξη πολλαπλασιασμού ενώ για τον δεύτερο μία πρόσθεση. Αντίθετα ο Αλγόριθμος 28 έχει την ιδιότητα της αναδρομής ουράς. Ας εστιάσουμε όμως στη λειτουργία του Αλγόριθμου 23 που δεν έχει αυτή την ιδιότητα. Η κλήση Factorial(4) πυροδοτεί την ακολουθία κλήσεων

$$\begin{aligned}\text{Factorial(4)} &= 4 \cdot \text{Factorial}(3)\\ &= 4 \cdot (3 \cdot \text{Factorial}(2))\\ &= 4 \cdot (3 \cdot (2 \cdot \text{ Factorial}(1)))\\ &= 4 \cdot (3 \cdot (2 \cdot (1 \cdot \text{Factorial}(0))))\\ &= 4 \cdot (3 \cdot (2 \cdot (1 \cdot (1)))) = 24.\end{aligned}$$

Παρατηρούμε ότι κάθε αναδρομική κλήση πρέπει να ολοκληρωθεί προκειμένου να υπολογιστεί το αποτέλεσμα. Με αυτό τον τρόπο ο χώρος που χρησιμοποιείται δίνεται από το γινόμενο της «ποσότητας» μνήμης (stack frame) που απαιτεί μία κλήση επί τον αριθμό των αναδρομικών κλήσεων. Στην προκειμένη περίπτωση η μνήμη που απαιτείται είναι τάξης $O(n)$.

Μπορούμε να μετατρέψουμε τον Αλγόριθμο 23 σε έναν αλγόριθμο που έχει την ιδιότητα της αναδρομής ουράς. Για να γίνει αυτό αρκεί να χρησιμοποιήσουμε μία επιπλέον παράμετρο στην οποία θα αποθηκεύονται τα γινόμενο των τιμών που παίρνει η πρώτη παράμετρο κατά τη διάρκεια των αναδρομικών κλήσεων. Στο τέλος του υπολογισμού η παράμετρος αυτή θα περιέχει την ζητούμενη τιμή $(n!)$. Η υλοποίηση της ιδέας παρουσιάζεται στον Αλγόριθμο 31.

Στην αρχική κλήση της συνάρτησης του Αλγόριθμου 31 η τιμή της δεύτερης παρα-

Αλγόριθμος 31 Αναδρομικός υπολογισμός του $n!$ με αναδρομή ουράς.

Απαιτείται: Ακέραιος $n \geq 0$, παράμετρος p με αρχική τιμή ίση με 1
Επιστρέφεται: n παραγοντικό ως τιμή της παραμέτρου p.

```
1: function Factorial(int n, int p)
2:     if n > 1 then
3:         Factorial(n − 1, n · p);
4:     end if
5: end function
```

μέτρου είναι ίση με 1. Για παράδειγμα, για να υπολογιστεί η τιμή $4!$ έχουμε διαδοχικά τις κλήσεις

$$\text{Factorial}(4, 1)$$
$$\text{Factorial}(3, 4)$$
$$\text{Factorial}(2, 12)$$
$$\text{Factorial}(1, 24)$$

Ένα βασικό πλεονέκτημα όταν ισχύει η αναδρομή ουράς είναι ότι μπορούμε πολύ εύκολα να μετατρέψουμε έναν αναδρομικό αλγόριθμο σε μη-αναδρομικό αντικαθιστώντας το σχήμα αναδρομής με ένα επαναληπτικό σχήμα.[3] Στον Αλγόριθμο 32 περιγράφεται ένα γενικό σχήμα αναδρομής ουράς. Το ισοδύναμο μη-αναδρομικό σχήμα παρουσιάζεται στον Αλγόριθμο 33.

Αλγόριθμος 32 Γενικό αλγοριθμικό σχήμα με αναδρομή ουράς.

```
1: function TailRecursive(int p_1, int p_2, ..., int p_k)
2:     if Λογική Έκφραση then
3:         Εντολές βάσης αναδρομής;
4:     else
5:         Εντολές αναδρομικής κλήσης;
6:         TailRecursive(q_1, q_2, ..., q_k);
7:     end if
8: end function
```

Αλγόριθμος 33 Ισοδύναμο αλγοριθμικό σχήμα με επανάληψη.

```
1: function Iterative(int p_1, int p_2, ..., int p_k)
2:     while not (Λογική Έκφραση) do
3:         Εντολές αναδρομικής κλήσης;
4:         p_1 ← q_1; p_2 ← q_2; ··· , p_k ← q_k;
5:     end while
6:     Εντολές βάσης αναδρομής;
7: end function
```

[3] Είναι αληθές ότι κάθε αναδρομικός αλγόριθμος μπορεί να μετατραπεί σε μη-αναδρομικό όμως αυτό δεν μπορεί να γίνει εύκολα σε κάθε περίπτωση π.χ. μη-αναδρομικός υπολογισμός της συνάρτησης Ackermann ή άλλων *μη-πρωτόγονων συναρτήσεων (non-primitive functions).*

4.4 Αναδρομικοί αλγόριθμοι και ΣΥΒ

Όπως και στους αλγορίθμους που παρουσιάσαμε στα προηγούμενα κεφάλαια, είναι κρίσιμο να μπορέσουμε να εκτιμήσουμε την ασυμπτωτική συμπεριφορά και στην περίπτωση των αναδρομικών αλγορίθμων. Για αυτό θα πρέπει να εκφράσουμε τον αριθμό των ΣΥΒ που εκτελεί ένας αλγόριθμος σαν μία συνάρτηση της παραμέτρου n που εκφράζει το μέγεθος του στιγμιότυπου. Η συνάρτηση αυτή σε έναν αναδρομικό αλγόριθμο περιέχει δύο συνιστώσες· η μία αφορά τα ΣΥΒ που εκτελεί ο αλγόριθμος προκειμένου να επιλύσει στιγμιότυπα του ιδίου προβλήματος μικρότερου μεγέθους (αναδρομική συνιστώσα) και η δεύτερη αφορά τον αριθμό των ΣΥΒ που εκτελούνται σε κάθε κλήση εκτός των αναδρομικών κλήσεων. Οι συναρτήσεις του τύπου αυτού ονομάζονται *αναδρομικές*.

Σαν πρώτο παράδειγμα, θεωρούμε τον Αλγόριθμο 23 ο οποίος δοθέντος ενός μη-αρνητικού ακέραιου n, υπολογίζει την τιμή $n!$. Ο αλγόριθμος εκτελεί μία σύγκριση και εφόσον $n \geq 1$, έναν πολλαπλασιασμό, μία αφαίρεση και μία κλήση στην ίδια συνάρτηση. Άρα, για $n \geq 1$, ο αριθμός των ΣΥΒ που εκτελείται σε κάθε κλήση είναι σταθερός (και ίσος με 4). Επιπλέον πρέπει να προσθέσουμε τον αριθμό των ΣΥΒ που εκτελούνται για τον υπολογισμό της τιμής $(n-1)!$ - αναδρομική συνιστώσα του αλγόριθμου. Ο συνολικός αριθμός των ΣΥΒ για τον Αλγόριθμο 23 δίνεται από την (αναδρομική) συνάρτηση

$$T(n) = \begin{cases} T(n-1) + 4, & n \geq 1, \\ 1, & n = 0. \end{cases} \tag{4.4}$$

Παρατηρούμε ότι η συνάρτηση περιέχει έναν κλάδο - αυτόν που ορίζεται για $n = 0$ - στον οποίο παύει να είναι αναδρομική. Η τιμή του n για την οποία συμβαίνει αυτό ορίζει μία *οριακή συνθήκη*. Σε κάθε συνάρτηση η οποία περιγράφει τον αριθμό των ΣΥΒ πρέπει να υπάρχει τουλάχιστον μία τιμή του n που να καθορίζει μία οριακή συνθήκη και επιπλέον πρέπει το μέγεθος του στιγμιότυπου (ή των στιγμιοτύπων) το οποίο καλείται να λύσει ο αλγόριθμος σε κάθε αναδρομική κλήση να συγκλίνει προς την τιμή αυτή.

Στην περίπτωση του Αλγόριθμου 24 που υπολογίζει το $n-$οστό όρο της ακολουθίας Fibonacci,η αναδρομική συνιστώσα της συνάρτησης που αφορά τον αριθμό των ΣΥΒ έχει δύο όρους: ο αλγόριθμος καλεί τον εαυτό του δύο φορές, μία με παράμετρο το $n-1$ και μία με το $n-2$. Συνολικά, στη Γραμμή 2 εκτελείται μία σύγκριση ενώ στη Γραμμή 5 δύο αφαιρέσεις μία πρόσθεση και δύο κλήσεις συνάρτησης Επομένως ο αριθμός των ΣΥΒ δίνεται από την αναδρομική σχέση

$$T(n) = \begin{cases} T(n-1) + T(n-2) + 6, & n \geq 2, \\ 1, & n \leq 1. \end{cases} \tag{4.5}$$

Η επίλυση των αναδρομικών συναρτήσεων συνίσταται στην εύρεση μίας εξίσωσης η οποία θα εκφράζει τον αριθμό των ΣΥΒ σαν συνάρτηση μόνο της παραμέτρου n. Μια τέτοια μαθηματική σχέση ονομάζεται *κλειστή μορφή* της αναδρομικής συνάρτησης. Για παράδειγμα, η κλειστή μορφή της (4.4) είναι $T(n) = 4n + 1$.

Παρόλο που η κλειστή μορφή αποτελεί ζητούμενο μερικές φορές είναι αρκετά δύσκολο να υπολογιστεί. Όμως - όπως είδαμε και σε προηγούμενο κεφάλαιο - στην ανάλυση των αλγορίθμων δεν ενδιαφέρει τόσο ο απόλυτος αριθμός των ΣΥΒ αλλά η τάξη μεγέθους του *(ασυμπτωτική εκτίμηση)*. Αυτό διευκολύνει αρκετά· όταν γράφουμε την αναδρομική συνάρτηση η συνιστώσα που αφορά τα ΣΥΒ που εκτελούνται σε κάθε

κλήση του αλγόριθμου αρκεί να περιγράφεται ως τάξη μεγέθους και όχι απαραίτητα ως μία ακριβής έκφραση. Για παράδειγμα, αντί της (4.4), μπορούμε να γράψουμε την αναδρομική σχέση που περιγράφει τον αριθμό των ΣΥΒ που εκτελεί ο Αλγόριθμος 23 ως

$$T(n) = \begin{cases} T(n-1) + \Theta(1), & n \geq 1, \\ 1, & n = 0. \end{cases} \tag{4.6}$$

Ο όρος $\Theta(1)$ υποδηλώνει ότι σε κάθε κλήση του αλγόριθμου με $n \geq 1$ εκτελείται σταθερός αριθμός ΣΥΒ. Ο αριθμός αυτός δεν χρειάζεται να προσδιοριστεί επακριβώς προκειμένου να εκτιμηθεί η ασυμπτωτική συμπεριφορά του αλγόριθμου: σε κλειστή μορφή, ο αριθμός των ΣΥΒ που εκτελούνται δίνεται από τη συνάρτηση $T(n) = cn + 1$, όπου c μία θετική σταθερά μεγαλύτερη του μηδενός. Οπότε έχουμε ότι $T(n) = \Theta(n)$. Παρατηρείστε το ίδιο αποτέλεσμα παίρνουμε και στην περίπτωση που έχουμε προσδιορίσει ότι $c = 4$.

Σε επόμενο κεφάλαιο θα παρουσιάσουμε μεθόδους επίλυσης αναδρομικών εξισώσεων και προσδιορισμού της ασυμπτωτικής συμπεριφοράς τους.

4.5 Ασκήσεις

1. Να εκπονήσετε αλγόριθμο ο οποίος να εξάγει το $k-$ιοστό στοιχείο από μία στοίβα με n στοιχεία, όπου $k \leq n$.

2. Το πλήθος των διαφορετικών ομάδων καθεμία αποτελούμενη από k αντικείμενα που μπορούν να σχηματιστούν από ένα σύνολο n διακριτών αντικειμένων συμβολίζεται με $C(n, k)$ για $0 \leq k \leq n$. Ισχύει ότι $C(n, 0) = C(n, n) = 1$, για κάθε ακέραιο $n \geq 0$. Επίσης, για $1 \leq k \leq n-1$, η τιμή $C(n, k)$ δίνεται από τον τύπο

   $$C(n, k) = C(n-1, k) + C(n-1, k-1).$$

   Να εκπονήσετε αλγόριθμο ο οποίος να δέχεται σαν είσοδο ακεραίους k, n για τους οποίους να ισχύει $1 \leq k \leq n-1$, και να υπολογίζει την τιμή $C(n, k)$ με τη χρήση του παραπάνω τύπου

3. Να εκπονήσετε έναν αναδρομικό αλγόριθμο ο οποίος δέχεται σαν είσοδο έναν ακέραιο και παράγει σαν έξοδο τον ακέραιο που προκύπτει αν αντιστρέψουμε τη σειρά των ψηφίων.

4. Μία συμβολοσειρά αποτελούμενη από ψηφία του συνόλου $\{0, 1\}$ ονομάζεται *καρκινική* αν ταυτίζεται με την αντίστροφή της. Να εκπονήσετε αλγόριθμο ο οποίος να δέχεται σαν είσοδο μία σειρά από αριθμούς και να αποφασίζει αν η συμβολοσειρά αυτή είναι καρκινική.

5. Να εκπονήσετε έναν αλγόριθμο ο οποίος να δέχεται σαν είσοδο έναν πίνακα ακεραίων A διάστασης $n \times n$ και να υπολογίζει την ορίζουσα του ($\det(A)$) από τον αναδρομικό τύπο

   $$\det(A) = \sum_{j=1}^{n} (-1)^{1+j} a_{1j} \cdot \det(A^{1,j}),$$

   όπου $\det(A^{1,j})$ είναι η ορίζουσα του πίνακα που προκύπτει αν από τον πίνακα A διαγράψουμε την πρώτη γραμμή και την j στήλη.