Κεφάλαιο 1

Επαναληπτικοί αλγόριθμοι: Μέτρα προόδου και αναλλοίωτες συνθήκες

Η χρήση ενός *επαναληπτικού αλγόριθμου* για την επίλυση ενός υπολογιστικού προβλήματος είναι κάπως σαν να οδηγεί κάποιος σε έναν δρόμο, πιθανόν μακρύ και δύσκολο, από την αφετηρία ως τον προορισμό. Με κάθε επανάληψη, έχετε μια μέθοδο η οποία σας φέρνει ένα βήμα πιο κοντά στον προορισμό. Για να βεβαιωθείτε ότι προχωράτε προς τα εμπρός, πρέπει να έχετε ένα *μέτρο προόδου* που σας λέει πόσο απέχετε από τον προορισμό ή πόση απόσταση έχετε διανύσει από την αφετηρία. Δεν είναι δυνατόν να γνωρίζετε ακριβώς προς τα πού θα κινηθεί ο αλγόριθμος, γι' αυτό πρέπει να προετοιμαστείτε για παρακάμψεις και λοξοδρομήσεις. Από την άλλη πλευρά, δεν είστε αναγκασμένοι να γνωρίζετε πώς να χειριστείτε κάθε χαντάκι και κάθε αδιέξοδο σε όλο τον κόσμο. Ένας συμβιβασμός ανάμεσα σ' αυτά τα δύο είναι να έχετε μια *αναλλοίωτη συνθήκη*, η οποία ορίζει έναν δρόμο (ή μια περιοχή) που δεν πρέπει να εγκαταλείψετε. Όταν ταξιδεύετε, ας ανησυχείτε για ένα μόνο βήμα κάθε φορά. Πρέπει να γνωρίζετε πώς να μπείτε στη σωστή πορεία από οποιαδήποτε αφετηρία. Από κάθε σημείο στον δρόμο, πρέπει να γνωρίζετε ποιες ενέργειες θα εκτελέσετε προκειμένου να προχωρήσετε προς τα εμπρός χωρίς να εγκαταλείψετε τον δρόμο. Τέλος, όταν θα έχετε κάνει αρκετή πρόοδο στον δρόμο, πρέπει να γνωρίζετε πώς να βγείτε απ' αυτόν και να φτάσετε στον προορισμό σας σε εύλογο χρονικό διάστημα.

1.1 Αλλαγή κουλτούρας: Μια ακολουθία ενεργειών και μια ακολουθία ισχυρισμών

Η κατανόηση των επαναληπτικών αλγόριθμων απαιτεί την κατανόηση της διαφοράς ανάμεσα σε μια *αναλλοίωτη συνθήκη*, η οποία είναι ένας *ισχυρισμός* ή μια εικόνα του υπολογισμού σε συγκεκριμένο χρονικό σημείο, και στις ενέργειες που απαιτούνται προκειμένου να διατηρηθεί μια τέτοια αναλλοίωτη συνθήκη.

Ως εκ τούτου, θα ξεκινήσουμε με μια απόπειρα να κατανοήσουμε αυτή τη διαφορά.

Μία από τις πρώτες σημαντικές αλλαγές κουλτούρας που αγωνίστηκαν να επιτύχουν οι προγραμματιστές είναι να μη θεωρούν τους αλγόριθμους ακολουθίες από ενέργειες, αλλά ακολουθίες από στιγμιότυπα της κατάστασης του υπολογιστή. Οι προγραμματιστές έχουν την τάση να προτιμούν την ακολουθία από ενέργειες, επειδή ο κώδικας είναι μια ακολουθία από οδηγίες για ενέργειες και ένας υπολογισμός είναι μια ακολουθία από ενέργειες. Αν και αυτή η άποψη είναι σημαντική, υπάρχει και μια διαφορετική. Φανταστείτε ότι σταματάτε τον χρόνο σε καίρια σημεία κατά τη διάρκεια του υπολογισμού και φωτογραφίζετε την κατάσταση του υπολογιστή. Εάν το κάνετε αυτό, τότε μπορείτε να δείτε έναν υπολογισμό και ως μια σειρά από τέτοια στιγμιότυπα. Όταν καταφέρετε να έχετε στη διάθεσή σας δύο διαφορετικές οπτικές γωνίες για να βλέπετε το ίδιο πράγμα, τότε αποκτάτε πρόσβαση σε περισσότερα εργαλεία για να το χειριστείτε *και* είστε σε θέση να το κατανοήσετε καλύτερα. Δείτε ένα παράδειγμα του πώς ένας υπολογισμός μπορεί να αντιμετωπιστεί με δύο τρόπους: ως ισχυρισμοί για την τρέχουσα κατάσταση του υπολογισμού και ως ενέργειες που προωθούν τον υπολογισμό στην επόμενη κατάσταση.

```
Max(a, b, c)
   Προσυνθήκη: Η είσοδος έχει 3 αριθμούς.
   m = a
   ισχυρισμός: Το m είναι max στο {a}.
   εάν (b > m)
      m = b
   τέλος εάν
   ισχυρισμός: Το m είναι max στο {a,b}.
    εάν (c > m)
      m = c
   τέλος εάν
   ισχυρισμός: Το m είναι max στο {a,b,c}.
   επίστρεψε (m)
   Μετασυνθήκη: Επίστρεψε το max στο {a,b,c}.
τέλος αλγόριθμου
```

Η πρόκληση της θεώρησης ως ακολουθίας ενεργειών: Έστω ότι κάποιος σχεδιάζει έναν νέο αλγόριθμο ή εξηγεί έναν αλγόριθμο σε έναν φίλο. Εάν αυτός ο κάποιος σκέφτεται τον αλγόριθμο ως μια ακολουθία από ενέργειες, τότε μάλλον θα ξεκινήσει από την αρχή: Κάνε αυτό. Μετά κάνε το άλλο. Στη συνέχεια κάνε αυτό κ.λπ. Σύντομα, όμως, θα χαθεί και δεν θα γνωρίζει πού βρίσκεται. Για να αντιμετωπίσει αυτή την κατάσταση, θα πρέπει να παρακολουθεί παράλληλα και τις αλλαγές της κατάστασης του υπολογιστή με κάθε νέα ενέργεια. Προκειμένου να γνωρίζει ποια ενέργεια θα πρέπει να εκτελεστεί στη συνέχεια, πρέπει να έχει και ένα γενικό πλάνο που να δείχνει προς τα πού πρέπει να κατευθυνθεί ο υπολογισμός. Τα πράγματα γίνονται χειρότερα αν σκεφτούμε ότι ένας υπολογισμός έχει πολλά *αν* και πολλούς *βρόχους* και, συνεπώς, θα πρέπει να συνυπολογιστούν όλες οι δυνατές διαφορετικές διαδρομές που μπορεί να ακολουθήσει ο υπολογισμός, αν τον θεωρήσουμε ακολουθία ενεργειών.

Τα πλεονεκτήματα της θεώρησης ως ακολουθίας στιγμιότυπων: Αυτή η νέα θεώρηση είναι ιδιαίτερα χρήσιμη για κάποιον που αναζητά τρόπους για να επινοήσει, να εξηγήσει ή να αναπτύξει έναν αλγόριθμο.

Προ- και μετασυνθήκες: Για να είναι σε θέση κάποιος να σχεδιάσει έναν αλγόριθμο που θα λύνει ένα υπολογιστικό πρόβλημα, πρέπει πρώτα να ορίσει προσεκτικά το πρόβλημα. Αυτό γίνεται με προ- και με μετασυνθήκες, οι οποίες παρέχουν την αρχική εικόνα, τον *ισχυρισμό*, δηλαδή, σχετικά με το στιγμιότυπο εισόδου, αλλά και μια αντίστοιχη εικόνα ή ισχυρισμό για την απαιτούμενη έξοδο.

Ξεκινήστε από τη μέση: Αντί να ξεκινήσετε με την πρώτη γραμμή του κώδικα, ένας εναλλακτικός τρόπος είναι η σχεδίαση ενός αλγόριθμου έτσι, ώστε να γίνεται μια μεταπήδηση στη μέση του υπολογισμού και να αποτυπώνεται μια στατική εικόνα –ή ισχυρισμός– της κατάστασης στην οποία θα θέλαμε να βρίσκεται ο υπολογισμός εκείνη τη χρονική στιγμή. Αυτή η εικόνα δεν χρειάζεται να περιλαμβάνει την ακριβή τιμή κάθε μεταβλητής που εμπλέκεται στον υπολογισμό. Αντίθετα, ο ισχυρισμός αυτός θα περιγράφει γενικές ιδιότητες και σχέσεις ανάμεσα στις διάφορες δομές δεδομένων που είναι σημαντικές για την κατανόηση του αλγόριθμου. Εάν αυτός ο ισχυρισμός είναι επαρκώς γενικός, δεν θα αποτυπώνει μόνο αυτό το συγκεκριμένο σημείο κατά τη διάρκεια του υπολογισμού, αλλά πολλά παρόμοια σημεία. Τότε, ίσως ο ισχυρισμός αυτός να μπορεί να γίνει κομμάτι ενός βρόχου (δηλαδή επαναληπτικής διαδικασίας).

Ακολουθία στιγμιότυπων: Όταν δημιουργήσουμε μια ακολουθία από ισχυρισμούς μ' αυτό τον τρόπο, τότε μπροστά μας θα απλωθεί όλη η διαδρομή του υπολογισμού.

Συμπληρώστε τις ενέργειες: Αυτοί οι ισχυρισμοί είναι απλά στατικά στιγμιότυπα του υπολογισμού (φωτογραφίες), όπως όταν σταματάμε τον χρόνο στα αντίστοιχα σημεία του. Ακόμα δεν έχουμε λάβει υπόψη καμία ενέργεια. Το τελικό βήμα είναι να συμπληρώσουμε τις ενέργειες (τον κώδικα) ανάμεσα σε διαδοχικούς ισχυρισμούς.

Ένα βήμα κάθε φορά: Κάθε τέτοιο κομμάτι ενεργειών μπορεί να εκτελεστεί εντελώς ανεξάρτητα από τα άλλα. Είναι πολύ πιο εύκολο να βλέπουμε κάθε ενέργεια ξεχωριστά παρά να ανησυχούμε για όλο τον υπολογισμό μονομιάς. Θα πρέπει μάλιστα να γνωρίζετε ότι μπορείτε να ολοκληρώσετε αυτά τα κομμάτια με όποια σειρά επιθυμείτε και να τροποποιείτε κάθε κομμάτι χωρίς να ανησυχείτε για τον αντίκτυπο που θα έχει στα άλλα.

Επιστροφή από τον Άρη: Έτσι ακριβώς, σαν να είχατε επιστρέψει από τον πλανήτη Άρη, θα πρέπει να συμπληρώσετε τον κώδικα ανάμεσα στον i και τον $i+1$ ισχυρισμό. Φανταστείτε ότι μόλις επιστρέψατε από τον Άρη και το μοναδικό πράγμα που γνωρίζετε για την τρέχουσα κατάσταση του υπολογισμού σας είναι αυτή που περιγράφει ο i ισχυρισμός. Ο υπολογισμός μπορεί μάλιστα να βρίσκεται σε μια κατάσταση στην οποία είναι αδύνατο να φτάσει ποτέ, λαμβάνοντας υπόψη τον αλγόριθμο που έχει σχεδιαστεί μέχρι αυτό το σημείο. Όμως, το να επιτρέπουμε ακόμη και κάτι τέτοιο μας οδηγεί στην επίτευξη του στόχου για την ανεξαρτησία μεταξύ των ξεχωριστών κομματιών των ενεργειών.

Εκτελέστε ένα βήμα: Ευρισκόμενοι σε μια κατάσταση στην οποία ισχύει ο ισχυρισμός i, αρκεί να γράψετε λίγο απλό κώδικα που θα εκτελεί κάποιες απλές ενέργειες που θα αλλάζουν την κατάσταση του υπολογισμού, ώστε να ισχύει, πλέον, ο ισχυρισμός $i+1$.

Απόδειξη ορθότητας κάθε βήματος: Η απόδειξη ότι ο αλγόριθμός σας λειτουργεί μπορεί να γίνει βήμα βήμα, εστιάζοντας σε ένα κομμάτι του κάθε φορά. Πρέπει να αποδείξετε ότι αν ο χρόνος σταματήσει και η κατάσταση του υπολογισμού είναι τέτοια που να ισχύει ο i ισχυρισμός και ξεκινήσετε τον χρόνο ξανά για αρκετή ώρα, τόση ώστε να εκτελεστεί το επόμενο κομμάτι κώδικα, τότε, όταν σταματήσετε ξανά τον χρόνο, η κατάσταση του υπολογισμού θα είναι τέτοια

που να ισχύει ο $i+1$ ισχυρισμός. Αυτή η απόδειξη μπορεί να είναι μια μαθηματική (αυστηρή) απόδειξη ή μπορεί να είναι ένα απλό, άτυπο (αλλά κατανοητό) επιχείρημα. Σε κάθε περίπτωση, η μαθηματική διατύπωση αυτού που πρέπει να αποδειχθεί είναι η εξής:

$$\langle i\text{-}\textit{οστός–ισχυρισμός}\rangle \& \textit{κώδικας}_i \Rightarrow \langle i+1\ \textit{αρχικός–ισχυρισμός}\rangle$$

Απόδειξη ορθότητας του αλγόριθμου: Όλα αυτά τα μεμονωμένα βήματα μπορούν να συγκεντρωθούν σε έναν πλήρη, λειτουργικό αλγόριθμο. Θεωρούμε ότι το στιγμιότυπο εισόδου που δίνεται ικανοποιεί την προσυνθήκη. Σε κάποιο σημείο, αποδείξαμε ότι αν ισχύει η προσυνθήκη και εκτελεστεί το πρώτο κομμάτι κώδικα, τότε η κατάσταση του υπολογισμού θα είναι τέτοια που να ισχύει ο πρώτος ισχυρισμός. Σε κάποιο άλλο σημείο, αποδείξαμε ότι αν ισχύει ο πρώτος ισχυρισμός και εκτελεστεί το δεύτερο κομμάτι κώδικα, τότε η κατάσταση του υπολογισμού θα είναι τέτοια που να ισχύει ο δεύτερος ισχυρισμός. Αυτό γίνεται για κάθε κομμάτι κώδικα. Τότε όλες αυτές οι ανεξάρτητα αποδεδειγμένες προτάσεις μπορούν να συγκεντρωθούν έτσι ώστε να αποδείξουν το εξής: αν, αρχικά, το στιγμιότυπο εισόδου ικανοποιεί την προσυνθήκη και εκτελεστεί ολόκληρος ο κώδικας, τότε, στο τέλος, η κατάσταση του υπολογισμού θα είναι τέτοια που να ικανοποιείται η μετασυνθήκη. Είναι αυτό ακριβώς που απαιτείται προκειμένου να αποδειχθεί ότι ο αλγόριθμος λειτουργεί (είναι, δηλαδή, σωστός).

1.2 Τα βήματα ανάπτυξης ενός επαναληπτικού αλγόριθμου

Επαναληπτικοί αλγόριθμοι: Ένας καλός τρόπος για να σχεδιάζετε προγράμματα υπολογιστών σε πολλές περιπτώσεις είναι να αποθηκεύετε τις βασικές πληροφορίες που γνωρίζετε ήδη σε κάποια δομή δεδομένων έτσι ώστε, στη συνέχεια, κάθε επανάληψη του κύριου βρόχου υπολογισμού να σας μεταφέρει ακόμα ένα βήμα προς τον προορισμό σας, κάνοντας μια μικρή αλλαγή σε αυτές τις πληροφορίες.

Αναλλοίωτη συνθήκη βρόχου: Μια *αναλλοίωτη συνθήκη βρόχου* εκφράζει σημαντικές σχέσεις που πρέπει να ισχύουν ανάμεσα στις μεταβλητές στην αρχή κάθε επανάληψης και όταν ο βρόχος τερματίζεται. Εάν η συνθήκη ισχύει, τότε ο υπολογισμός βρίσκεται σε καλό δρόμο, αλλά αν δεν ισχύει, τότε ο αλγόριθμος έχει αποτύχει.

Η δομή του κώδικα: Η βασική δομή του κώδικα έχει ως εξής:

```
αρχή ρουτίνας
   ⟨προσυνθήκη⟩
   κώδικας_προ βρόχου        % Όρισε αναλλοίωτη συνθήκη βρόχου
   βρόχος
      ⟨αναλλοίωτη συνθήκη βρόχου⟩
      έξοδος όταν ⟨συνθήκη εξόδου⟩
      κώδικας_βρόχος        % Προχώρα όσο διατηρείται η αναλλοίωτη συνθήκη
   τέλος βρόχου
   κώδικας_μετά τον βρόχο        % Τακτοποίησε τις εκκρεμείς εργασίες
   ⟨μετασυνθήκη⟩
τέλος ρουτίνας
```

Απόδειξη ορθότητας: Θα πρέπει, φυσικά, να είστε βέβαιοι ότι ο αλγόριθμός σας θα λειτουργεί για όλες τις εισόδους που θα του δίνονται και θα παράγει τη σωστή, κάθε φορά, απάντηση.

Χρόνος εκτέλεσης: Θα πρέπει επίσης να είστε βέβαιοι ότι ο αλγόριθμός σας ολοκληρώνεται σε εύλογο χρονικό διάστημα.

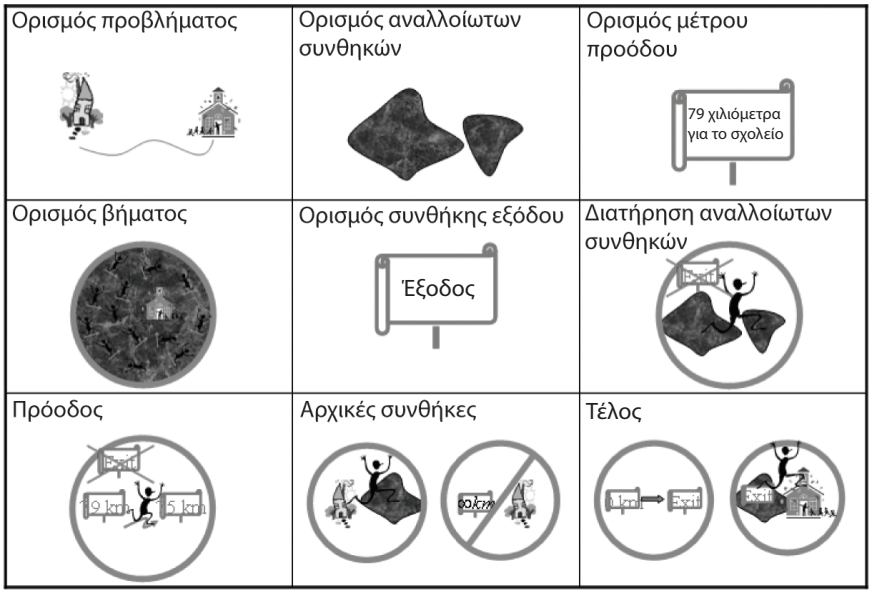
Τα πιο σημαντικά βήματα: Εάν πρέπει να σχεδιάσετε έναν αλγόριθμο, μην ξεκινήσετε πληκτρολογώντας κώδικα χωρίς να γνωρίζετε πραγματικά πώς ή γιατί λειτουργεί ο αλγόριθμος. Αντίθετα, προτείνω πρώτα να ολοκληρώσετε τις εργασίες που περιγράφω στην Εικόνα 1.1. Αυτές οι εργασίες πρέπει να συνδυάζονται με πολύ λεπτούς τρόπους. Ίσως χρειαστεί να τις επαναλάβετε αρκετές φορές, προσαρμόζοντας ό,τι έχετε κάνει, μέχρι όλα να ταιριάζουν ακριβώς όπως πρέπει.

1) ***Προδιαγραφές:*** Τι πρόβλημα επιχειρείτε να λύσετε; Ποιες είναι οι προ- και οι μετασυνθήκες – δηλαδή, από πού ξεκινάτε και πού βρίσκεται ο προορισμός σας;

2) ***Βασικά βήματα:*** Ποια βασικά βήματα θα σας οδηγήσουν, λίγο ως πολύ, στη σωστή κατεύθυνση;

3) ***Μέτρο προόδου:*** Πρέπει να ορίσετε ένα μέτρο προόδου: πού βρίσκονται οι χιλιομετρικοί δείκτες κατά μήκος της διαδρομής;

4) ***Η αναλλοίωτη συνθήκη βρόχου:*** Πρέπει να ορίσετε μια αναλλοίωτη συνθήκη βρόχου η οποία θα σας δίνει μια εικόνα για την κατάσταση του υπολογισμού σας όταν βρίσκεται στην αρχή του κύριου βρόχου. Με άλλα λόγια, καθορίστε τον δρόμο στον οποίο θα κρατηθεί ο αλγόριθμος.



Εικόνα 1.1: Οι απαιτήσεις ενός επαναληπτικού αλγόριθμου.

5) ***Κύρια βήματα:*** Για κάθε σημείο στον δρόμο του υπολογισμού, πρέπει να γράφετε τον ψευδοκώδικα *κώδικας*$_{βρόχος}$ που θα ολοκληρώνει ένα συγκεκριμένο βήμα. Δεν είστε υποχρεωμένοι να ξεκινήστε από το πρώτο σημείο. Θα πρότεινα να θεωρήσετε πρώτα ένα τυπικό βήμα υπολογισμού περίπου στο μέσο του συνολικού υπολογισμού σας.

6) ***Πρόοδος:*** Κάθε επανάληψη του κύριου βήματός σας πρέπει να προχωρά σύμφωνα με το μέτρο προόδου που έχετε ορίσει.

7) ***Διατήρηση αναλλοίωτης συνθήκης:*** Κάθε επανάληψη του κύριου βήματος πρέπει να διασφαλίζει ότι η αναλλοίωτη συνθήκη ισχύει ξανά όταν ο υπολογισμός επιστρέφει στην αρχή του βρόχου. (Με τη μέθοδο της επαγωγής τότε θα αποδειχθεί ότι η αναλλοίωτη ισχύει πάντα.)

8) ***Αρχικοποίηση της αναλλοίωτης συνθήκης:*** Τώρα που γνωρίζετε περίπου προς τα πού πηγαίνετε, καταλαβαίνετε καλύτερα το πώς πρέπει να ξεκινήσετε. Πρέπει να γράψετε τον ψευδοκώδικα *κώδικας*$_{προ\ βρόχου}$ ώστε να περιγράψετε, αρχικά, την αναλλοίωτη συνθήκη. Ποια διαδρομή θα ακολουθήσετε για να πάτε από το σπίτι σας στον σωστό δρόμο;

9) ***Συνθήκη εξόδου:*** Πρέπει να γράψετε τη συνθήκη ⟨*συνθήκη εξόδου*⟩ η οποία θα αναγκάζει τον υπολογισμό να εγκαταλείψει τον βρόχο.

10) ***Τερματισμός:*** Πώς διασφαλίζει η συνθήκη εξόδου μαζί με την αναλλοίωτη συνθήκη ότι το πρόβλημα λύθηκε; Όταν βρίσκεστε στο τέλος του δρόμου, αλλά ακόμα σ' αυτόν, πώς παράγετε την απαιτούμενη έξοδο; Πρέπει να γράψετε τον ψευδοκώδικα *κώδικας*$_{μετά\ τον\ βρόχο}$ που θα τακτοποιήσει τις εκκρεμείς εργασίες και θα επιστρέψει την απαιτούμενη έξοδο.

11) ***Τερματισμός και χρόνος εκτέλεσης:*** Πόση πρόοδο θα πρέπει να έχετε κάνει προκειμένου να καταλάβετε ότι έχετε φτάσει στην έξοδο; Αυτό αναφέρεται σε μια εκτίμηση του χρόνου εκτέλεσης του αλγόριθμού σας.

12) ***Ειδικές περιπτώσεις:*** Όταν επιχειρείτε αρχικά να σχεδιάσετε έναν αλγόριθμο, θα πρέπει να λαμβάνετε υπόψη σας μόνο έναν γενικό τύπο στιγμιότυπων εισόδου. Αργότερα, θα πρέπει να επαναλάβετε τα βήματα και για άλλους τύπους στιγμιότυπων και ειδικές περιπτώσεις. Θα πρέπει επίσης να δοκιμάσετε τον αλγόριθμό σας σε αρκετά διαφορετικά παραδείγματα.

13) ***Κωδικοποίηση και λεπτομέρειες υλοποίησης:*** Τώρα είστε έτοιμοι να ενώσετε όλα τα κομμάτια και να παραγάγετε τον ψευδοκώδικα για τον αλγόριθμο. Σε αυτό το σημείο, ίσως χρειαστεί να παρέχετε κάποιες πρόσθετες λεπτομέρειες υλοποίησης.

14) ***Αυστηρή (μαθηματική) απόδειξη:*** Εάν τα παραπάνω κομμάτια ταιριάζουν όπως πρέπει, τότε ο αλγόριθμός σας λειτουργεί.

ΠΑΡΑΔΕΙΓΜΑ 1.2.1

Ο αλγόριθμος δύο δαχτύλων εύρεσης μέγιστου για παρουσίαση αυτών των εννοιών

1) ***Προδιαγραφές:*** Ένα στιγμιότυπο εισόδου αποτελείται από μια λίστα

$L(1..n)$ με στοιχεία. Η έξοδος αποτελείται από έναν δείκτη i, ώστε το $L(i)$ να έχει μέγιστη τιμή. Εάν υπάρχουν πολλά στοιχεία με την ίδια μέγιστη τιμή, τότε ο αλγόριθμος επιστρέφει οποιοδήποτε απ' αυτά.

2) *Βασικά βήματα:* Αποφασίζετε να εφαρμόσετε τη μέθοδο δύο δαχτύλων. Το δεξί σας δάχτυλο διατρέχει τη λίστα από πάνω προς τα κάτω.

3) *Μέτρο προόδου:* Το μέτρο προόδου είναι πόσο προς τα κάτω έχει προχωρήσει στη λίστα το δεξί σας δάχτυλο.

4) *Η αναλλοίωτη συνθήκη:* Η αναλλοίωτη συνθήκη δηλώνει ότι το αριστερό σας δάχτυλο δείχνει ένα από τα μεγαλύτερα στοιχεία της λίστας που έχετε συναντήσει έως τώρα με το δεξί σας δάχτυλο.

5) *Κύρια βήματα:* Σε κάθε επανάληψη, μετακινείτε το δεξί σας δάχτυλο ένα στοιχείο πιο κάτω στη λίστα. Εάν το δεξί σας δάχτυλο δείχνει τώρα ένα στοιχείο της λίστας που είναι μεγαλύτερο από το στοιχείο του αριστερού δαχτύλου, μετακινήστε το αριστερό σας δάχτυλο εκεί όπου βρίσκεται το δεξί.

6) *Πρόοδος:* Πραγματοποιείτε πρόοδο, επειδή το δεξί σας δάχτυλο μετακινήθηκε κατά ένα στοιχείο πιο κάτω.

7) *Διατήρηση αναλλοίωτης συνθήκης βρόχου:* Θα καταλάβετε ότι η αναλλοίωτη συνθήκη διατηρείται ως εξής: Για κάθε βήμα, το νέο στοιχείο που δείχνει το αριστερό δάχτυλο είναι το μέγιστο ανάμεσα στο παλαιό στοιχείο του αριστερού δαχτύλου και στο νέο στοιχείο. Στην αναλλοίωτη συνθήκη, αυτό είναι το μέγιστο ανάμεσα στο μέγιστο της μικρότερης λίστας και στο νέο στοιχείο. Με μαθηματικούς όρους, πρόκειται για το μέγιστο της μεγαλύτερης λίστας.

8) *Επαλήθευση της αναλλοίωτης συνθήκης:* Ορίζετε αρχικά την αναλλοίωτη συνθήκη δείχνοντας με αμφότερα τα δάχτυλα το πρώτο στοιχείο.

9) *Συνθήκη εξόδου:* Έχετε τελειώσει όταν το δεξί σας δάχτυλο έχει διασχίσει όλη τη λίστα.

10) *Τερματισμός:* Θα γνωρίζετε τελικά ότι το πρόβλημα έχει λυθεί με τον εξής τρόπο: Από τη συνθήκη εξόδου, έπεται ότι το δεξί σας δάχτυλο έχει

συναντήσει όλα τα στοιχεία. Από την αναλλοίωτη συνθήκη, έπεται ότι το αριστερό σας δάχτυλο δείχνει το μέγιστο στοιχείο. Τότε θα πρέπει να επιστρέψετε αυτό το στοιχείο.

11) ***Τερματισμός και χρόνος εκτέλεσης:*** Ο χρόνος (αριθμός βημάτων) που απαιτείται για τον υπολογισμό είναι μια σταθερή ποσότητα (δηλαδή ανεξάρτητη από το μήκος της λίστας) επί το μήκος της λίστας.

12) ***Ειδικές περιπτώσεις:*** Ελέγξτε τι συμβαίνει όταν υπάρχουν πολλά στοιχεία με την ίδια τιμή ή όταν $n = 0$ ή $n = 1$.

13) ***Κωδικοποίηση και λεπτομέρειες υλοποίησης:***

αλγόριθμος *Find Max (L)*
⟨***προσυνθήκη***⟩**:** L είναι ένας πίνακας με n στοιχεία.
⟨***μετασυνθήκη***⟩**:** Επιστρέφει έναν δείκτη σε στοιχείο με τη μέγιστη τιμή.
αρχή
  $i = 1 \cdot j = 1$
  βρόχος
    ⟨***αναλλοίωτη συνθήκη***⟩**:** $L[i]$ είναι το μέγιστο στο $L[1..j]$.
    έξοδος όταν $(j \geq n)$
    % Προχώρα ενώ διατηρείται η αναλλοίωτη συνθήκη
    $j = j + 1$
    αν $(L[i] < L[j])$ τότε $i = j$
  τέλος βρόχου
  επίστρεψε(i)
τέλος αλγόριθμου

14) ***Αυστηρή απόδειξη:*** Η ορθότητα του αλγόριθμου προκύπτει από τα παραπάνω βήματα.

Ένας νέος τρόπος σκέψης: Ίσως μπείτε στον πειρασμό να σκεφτείτε ότι τα μέτρα προόδου και οι αναλλοίωτες συνθήκες είναι θεωρητικές φλυαρίες. Αλλά ο κλάδος, έπειτα από πολλά λάθη που στοίχισαν ακριβά, εκτιμά βαθύτερα την ανάγκη για ορθότητα. Η φιλοσοφία μας είναι να σας διδάξουμε πώς να σκέφτε-

στε, να αναπτύσσετε και να περιγράφετε αλγόριθμους με τέτοιον τρόπο ώστε η ορθότητά τους να είναι διάφανη. Γι' αυτό τον λόγο, τα μέτρα προόδου και οι αναλλοίωτες συνθήκες είναι ουσιώδη στοιχεία. Έτσι, η περιγραφή των προηγούμενων αλγόριθμων και των αποδείξεων ορθότητάς τους συνενώνονται σε μία και μοναδική οντότητα.

Μείνετε προσγειωμένοι: Οι αναλλοίωτες συνθήκες συνιστούν φιλοσοφία ζωής και μας οδηγούν σε ρεαλισμό. Το μεγαλύτερο μέρος του κώδικα που βαθμολογώ ως δάσκαλος μου δίνει μια αίσθηση ότι οι μαθητές μου πετούν στα σύννεφα. Υπάρχει μια εξέλιξη στον κώδικα, αλλά δεν καταλαβαίνω τι σημαίνουν οι μεταβλητές, πώς ταιριάζουν μεταξύ τους, προς τα πού κατευθύνεται ο αλγόριθμος ή πώς πρέπει να αρχίσω να σκέφτομαι γι' αυτόν. Οι αναλλοίωτες συνθήκες σημαίνουν ότι ξεκινώ τη μέρα μου στο σπίτι, όπου ξέρω τι ισχύει και ποια είναι η σημασία των πραγμάτων. Από εκεί, έχω όλα όσα απαιτούνται για να τολμήσω να βγω στο άγνωστο. Ωστόσο, οι αναλλοίωτες συνθήκες σημαίνουν επίσης ότι, αφού διανύσω έναν ολόκληρο κύκλο, θα επιστρέψω στην ασφάλεια του σπιτιού μου στο τέλος της ημέρας.

ΑΣΚΗΣΗ 1.2.1 *Ποιοι είναι οι αυστηροί μαθηματικοί ισχυρισμοί που αναφέρονται σε αναλλοίωτες συνθήκες και που πρέπει να αποδειχθούν, έτσι ώστε να αποδειχθεί ότι αν το πρόγραμμα τερματιστεί τότε επιτυγχάνεται η μετασυνθήκη;*

1.3 Περισσότερα για τα βήματα

Σ' αυτή την ενότητα δίνω περισσότερες λεπτομέρειες για τα βήματα που πρέπει να γίνουν προκειμένου να αναπτυχθεί ένας επαναληπτικός αλγόριθμος.

1) ***Προδιαγραφές:*** Για να σχεδιάσουμε έναν επαναληπτικό αλγόριθμο, πρέπει πρώτα να γνωρίζουμε ακριβώς τι είναι αυτό που υποτίθεται ότι πρέπει να κάνει.

> ***Προσυνθήκες:*** Ποια είναι τα έγκυρα στιγμιότυπα εισόδου; Οποιοσδήποτε ισχυρισμός ο οποίος πρέπει να ισχύει σε σχέση με το στιγμιότυπο εισόδου αναφέρεται ως *προσυνθήκη*.
>
> ***Μετασυνθήκες:*** Ποια είναι η απαιτούμενη έξοδος για κάθε έγκυρο στιγμιότυπο; Οποιοσδήποτε ισχυρισμός ο οποίος πρέπει να ισχύει σε σχέση με την έξοδο αναφέρεται ως *μετασυνθήκη*.

Ορθότητα: Ένας αλγόριθμος για το πρόβλημα είναι *σωστός* αν για κάθε έγκυρο στιγμιότυπο εισόδου παράγεται η απαιτούμενη έξοδος. Εάν το στιγμιότυπο εισόδου δεν ικανοποιεί τις προσυνθήκες, τότε δεν υπάρχει καμία βεβαιότητα για τίποτα. Αυστηρά, αυτό εκφράζεται ως

$$\langle \textit{προσυνθήκη} \rangle \ \& \ \textit{κώδικας}_{\textit{αλγ}} \Rightarrow \langle \textit{μετασυνθήκη} \rangle$$

Αυτή η συνθήκη ορθότητας σχετίζεται μόνο με τις προδιαγραφές εισόδου-εξόδου.

Παράδειγμα: Το πρόβλημα *ταξινόμησης* ορίζεται ως εξής:

Προσυνθήκες: Η είσοδος είναι μια λίστα από n τιμές, συμπεριλαμβανομένων πιθανών επαναλήψεων σε αυτές.

Μετασυνθήκες: Η έξοδος είναι μια λίστα η οποία αποτελείται από τις ίδιες n τιμές σε μη φθίνουσα διάταξη.

Το συμβόλαιο: Οι προ- και οι μετασυνθήκες είναι, υπό μία έννοια, το συμβόλαιο ανάμεσα στον υλοποιητή και τον χρήστη του κωδικοποιημένου αλγόριθμου.

Υλοποιητής: Όταν γράφετε μια υπο-ρουτίνα, μπορείτε να θεωρήσετε ότι η είσοδος φτάνει στο πρόγραμμά σας με τη σωστή μορφή και ικανοποιεί όλες τις προσυνθήκες. Πρέπει να γράψετε την υπο-ρουτίνα έτσι, ώστε να εξασφαλίζει ότι όλες οι μετασυνθήκες θα ισχύουν μετά την εκτέλεση.

Χρήστης: Όταν χρησιμοποιείτε την υπο-ρουτίνα, πρέπει να εξασφαλίσετε ότι η είσοδος που δίνετε ικανοποιεί τις προσυνθήκες της υπο-ρουτίνας. Τότε θα μπορείτε να είστε βέβαιοι ότι η έξοδος ικανοποιεί τις μετασυνθήκες της.

2) *Βασικά βήματα:* Στην αρχή της διαδικασίας σχεδίασης του αλγόριθμου, θα ήταν χρήσιμο να σκεφτείτε τα βασικά βήματα ή τις βασικές ενέργειες που θα μπορούσατε να εκτελέσετε προκειμένου να προχωρήσετε προς την επίλυση του προβλήματος. Κάντε μερικά από αυτά τα βήματα για ένα απλό στιγμιότυπο εισόδου, ώστε να πάρετε μια ιδέα σχετικά με την πορεία που μπορεί να ακολουθήσει ο υπολογισμός σας. Πώς, όμως, μπορεί να αποδειχθεί ότι οι πληροφορίες που αποκτώνται με τη διαδικασία αυτή προχωρούν τον υπολογισμό;

3) *Μέτρο προόδου:* Πρέπει να ορίσετε μια λειτουργία η οποία, όταν λαμβάνει την τρέχουσα κατάσταση του υπολογισμού, επιστρέφει μια ακέραια τιμή που

εκφράζει την πρόοδο που έχει ήδη κάνει ο υπολογισμός ή την πρόοδο που υπολείπεται. Αυτή η τιμή αναφέρεται είτε ως *μέτρο προόδου* είτε ως *συνάρτηση δυναμικού*. Πρέπει να είναι τέτοια, ώστε η συνολική πρόοδος που απαιτείται για την επίλυση του προβλήματος να μην είναι άπειρη και σε κάθε επανάληψη ο υπολογισμός να προχωρά. Πέρα απ' αυτό, έχετε την απόλυτη ελευθερία να ορίσετε αυτό το μέτρο προόδου όπως θέλετε. Για παράδειγμα, το μέτρο σας μπορεί να δηλώνει την ποσότητα της εξόδου που παράγεται, την ποσότητα της εισόδου που λαμβάνεται υπόψη, τον βαθμό στον οποίο έχει περιοριστεί ο χώρος αναζήτησης, κάποια πιο δημιουργική συνάρτηση του έργου που έχει ολοκληρωθεί μέχρι το τρέχον σημείο ή πόσες περιπτώσεις έχουν ελεγχθεί μέχρι το τρέχον σημείο. Στην Ενότητα 1.4 περιγράφεται πώς αυτά τα διαφορετικά μέτρα οδηγούν σε διαφορετικούς τύπους επαναληπτικών αλγόριθμων.

4) ***Η αναλλοίωτη συνθήκη:*** Συχνά, η εύρεση της αναλλοίωτης συνθήκης είναι το πιο δύσκολο κομμάτι της σχεδίασης ενός αλγόριθμου. Απαιτεί εξάσκηση, επιμονή, δημιουργικότητα και βαθιά γνώση. Ωστόσο, έπειτα απ' αυτό, ο υπόλοιπος αλγόριθμος προκύπτει αβίαστα. Δείτε μερικές χρήσιμες συμβουλές.

Ορισμός: Μια *αναλλοίωτη συνθήκη βρόχου* είναι ένας ισχυρισμός που τοποθετείται στην αρχή ενός βρόχου και πρέπει να ισχύει κάθε φορά που ο υπολογισμός επιστρέφει στην αρχή του βρόχου.

Ισχυρισμοί: Πιο γενικά, ένας *ισχυρισμός* είναι μια δήλωση που γίνεται σε κάποιο συγκεκριμένο σημείο κατά τη διάρκεια της εκτέλεσης ενός αλγόριθμου σχετικά με την τρέχουσα κατάσταση των δομών δεδομένων του υπολογισμού, η οποία είναι είτε αληθής είτε ψευδής. Εάν είναι ψευδής, τότε υπάρχει κάποιο σφάλμα στη λογική του αλγόριθμου. Οι προ- και οι μετασυνθήκες είναι ειδικές περιπτώσεις ισχυρισμών, οι οποίες παρέχουν σαφή όρια μεταξύ συστημάτων, υπο-συστημάτων, ρουτινών και υπο-ρουτινών. Σε ένα τέτοιο πλαίσιο, οι ισχυρισμοί μπορούν επίσης να παρέχουν σημεία ελέγχου κατά τον υπολογισμό, ώστε να μπορούν όλοι να γνωρίζουν τι θα έπρεπε να έχει επιτευχθεί μέχρι το συγκεκριμένο σημείο. Οι *αναλλοίωτες συνθήκες* είναι το ίδιο, με τη διαφορά ότι εφαρμόζονται είτε σε έναν βρόχο που εκτελείται πολλές φορές είτε σε μια αντικειμενοστραφή δομή δεδομένων που είναι ενεργή καθ' όλη τη διάρκεια της «ζωής» του προγράμματος.

Σχεδίαση, κατανόηση και απόδειξη της ορθότητας: Γενικά, οι ισχυρισμοί δεν είναι εργασίες που πρέπει να εκτελέσει ο αλγόριθμος, αλλά

είναι μόνο σχόλια που προστίθενται προκειμένου να βοηθήσουν τον σχεδιαστή, τον υλοποιητή και τον αναγνώστη να κατανοήσουν τον αλγόριθμο και την ορθότητά του.

Αποσφαλμάτωση: Ορισμένες γλώσσες προγραμματισμού σάς επιτρέπουν να εισάγετε ισχυρισμούς ως γραμμές κώδικα. Εάν κατά τη διάρκεια της εκτέλεσης του προγράμματος ένας τέτοιος ισχυρισμός βρεθεί ψευδής, τότε το πρόγραμμα σταματά αυτόματα με ένα χρήσιμο μήνυμα σφάλματος – πράγμα χρήσιμο τόσο κατά τη διαδικασία αποσφαλμάτωσης του κώδικα αλλά και μετά την ολοκλήρωσή του. Είναι αυτό που συμβαίνει όταν ένα πλαίσιο με μήνυμα σφάλματος εμφανίζεται κατά την εκτέλεση ενός προγράμματος για να σας πει ότι πρέπει να επικοινωνήσετε με τον προμηθευτή αν το σφάλμα παραμείνει. Ωστόσο, δεν είναι δυνατόν να ελέγχονται όλοι οι ενδιαφέροντες ισχυρισμοί μέσα στον ίδιο τον υπολογισμό ενός προγράμματος.

Εικόνα από τη μέση: Μια αναλλοίωτη συνθήκη θα πρέπει να περιγράφει πώς θα θέλατε να μοιάζει η δομή δεδομένων όταν ο υπολογισμός βρίσκεται στην αρχή μιας επανάληψης. Από την περιγραφή σας, ο αναγνώστης θα πρέπει να αποκτήσει μια εικόνα. Σχεδιάστε και μια εικόνα αν θέλετε.

Χωρίς φόβο: Μια αναλλοίωτη συνθήκη βρόχου δεν χρειάζεται να αποτελείται από ακαταλαβίστικους μαθηματικούς τύπους και πολύπλοκες σχέσεις, εάν μια διαισθητική περιγραφή με ένα απλό κείμενο μεταφέρει την κύρια ιδέα καλύτερα. Από την άλλη μεριά, διατυπώσεις που γίνονται στη φυσική μας γλώσσα μπορεί να περιέχουν ασάφειες και να οδηγούν σε παρανοήσεις σχετικά με την αναλλοίωτη που πρέπει να ισχύει, οπότε η υιοθέτηση ενός πιο αυστηρού συμβολισμού με χρήση της μαθηματικής γλώσσας μπορεί να βοηθήσει κάποιες φορές. Αν χρειάζεται, πείτε δύο φορές το ίδιο πράγμα. Προτείνω να προσποιηθείτε ότι εξηγείτε τον αλγόριθμό σας σε έναν πρωτοετή φοιτητή.

Στον δρόμο: Μια αναλλοίωτη συνθήκη πρέπει να εξασφαλίζει ότι ο υπολογισμός εξακολουθεί να κινείται στην πορεία του προς τον προορισμό και ότι δεν γλίστρησε σε κάποιο χαντάκι ούτε έπεσε πάνω σε δέντρο.

Ένας φαρδύς δρόμος: Ο υπολογισμός ενός συγκεκριμένου αλγόριθμου σε μια συγκεκριμένη είσοδο ακολουθεί μια συγκεκριμένη γραμμή. Όταν ο

σχεδιαστής του αλγόριθμου γνωρίζει ακριβώς πού θα κατευθυνθεί αυτή η γραμμή, μπορεί να χρησιμοποιήσει μια πολύ σφιχτή αναλλοίωτη συνθήκη για να ορίσει έναν πολύ στενό δρόμο για την πορεία του αλγόριθμου. Από την άλλη πλευρά, επειδή ο αλγόριθμός σας πρέπει να λειτουργεί για άπειρο αριθμό στιγμιότυπων εισόδου και επειδή μπορεί να συναντήσετε και να υπερπηδήσετε πολλά εμπόδια στη διαδρομή, θα ήταν δύσκολο να προβλέψετε πού μπορεί να βρίσκεται ο υπολογισμός στη μέση της εκτέλεσης. Σε τέτοιες περιπτώσεις, η χρήση μιας πολύ χαλαρής αναλλοίωτης συνθήκης για τον ορισμό ενός πολύ φαρδιού δρόμου είναι απολύτως αποδεκτή. Η πορεία που ακολουθεί στην πραγματικότητα ο υπολογισμός πιθανόν να μην είναι ευθεία, αλλά όλα είναι καλά αν παραμένει εντός των ορίων του δρόμου και συνεχίζει να προχωρά προς τα μπροστά. Ένα πλεονέκτημα ενός φαρδιού δρόμου είναι ότι δίνει μεγαλύτερη ευελιξία στον τρόπο με τον οποίο υλοποιείται ο κύριος βρόχος. Ένα μειονέκτημα είναι ότι θα υπάρχουν πολύ περισσότερα μέρη όπου μπορεί να βρίσκεται ο αλγόριθμος και για κάθε τέτοιο μέρος θα πρέπει να καθορίσετε πώς θα προχωρήσει στη συνέχεια ο αλγόριθμος.

> **Παράδειγμα:** Ως παράδειγμα μιας χαλαρής αναλλοίωτης συνθήκης, στον αλγόριθμο δύο δαχτύλων για την εύρεση του μέγιστου, η αναλλοίωτη συνθήκη δεν υπαγορεύει απολύτως ποια καταχώρηση πρέπει να δείχνει το αριστερό σας δάχτυλο όταν υπάρχουν πολλές καταχωρήσεις με την ίδια μέγιστη τιμή.

Κατανοητή και εφικτή: Η αναλλοίωτη συνθήκη πρέπει να είναι *κατανοητή*, δηλαδή να βγάζει νόημα και να είναι αρκετά ισχυρή ώστε, με μια κατάλληλη συνθήκη εξόδου, να μπορεί να εγγυηθεί τη μετασυνθήκη. Η αναλλοίωτη συνθήκη πρέπει επίσης να είναι *εφικτή*, δηλαδή να μπορεί να ισχύσει αρχικά και η ισχύς της να διατηρηθεί καθ' όλη τη διάρκεια του υπολογισμού.

Να γνωρίζετε τι είναι μια αναλλοίωτη συνθήκη: Θα πρέπει να είστε ξεκάθαροι ως προς το ποια είναι η αναλλοίωτη συνθήκη. Δεν είναι κώδικας, προσυνθήκη, μετασυνθήκη ή κάποια άλλη ακατάλληλη, για τους στόχους μας, πληροφορία. Για παράδειγμα, το να δηλώσετε κάτι που ισχύει πάντα, όπως «$1 + 1 = 2$» ή «Η ρίζα περιέχει το μέγιστο στοιχείο σε οποιοδήποτε δέντρο-σωρό», μπορεί να είναι μια χρήσιμη πληροφορία για την απάντηση

στο πρόβλημα, αλλά δεν θα πρέπει σε καμία περίπτωση να αποτελεί μέρος μιας αναλλοίωτης συνθήκης.

Ομαλή ροή: Η αναλλοίωτη συνθήκη θα πρέπει να «ρέει» ομαλά από την αρχή ως το τέλος του αλγόριθμου.

- Στην αρχή, θα πρέπει να προκύπτει αβίαστα από τις προσυνθήκες.
- Θα πρέπει να υπάρχει πρόοδος με μικρά, φυσικά βήματα.
- Όταν ικανοποιηθεί η συνθήκη εξόδου, οι μετασυνθήκες θα πρέπει να προκύπτουν εύκολα.

Ζητήστε το 100%: Μια καλή φιλοσοφία στη ζωή είναι να ζητάτε το 100% των επιθυμιών σας, αλλά να μη θεωρείτε βέβαιο ότι θα το πάρετε.

Ονειρευτείτε: Μην είστε ντροπαλοί. Τι θα θέλατε να ισχύει στη μέση του υπολογισμού σας; Αυτό μπορεί να είναι μια εύλογη αναλλοίωτη συνθήκη, αλλά, πάλι, μπορεί και να μην είναι.

Προσποιηθείτε: Φανταστείτε ότι ένα τζίνι πραγματοποίησε μια ευχή σας. Βρίσκεστε τώρα στη μέση του υπολογισμού σας και η αναλλοίωτη συνθήκη των ονείρων σας ισχύει πραγματικά.

Διατηρήστε την αναλλοίωτη συνθήκη: Απ' αυτό το σημείο, μπορείτε να κάνετε κάποια βήματα στον υπολογισμό σας διατηρώντας ταυτόχρονα την αναλλοίωτη συνθήκη; Εάν ναι, τότε όλα είναι τέλεια. Εάν όμως όχι, γενικά υπάρχουν δύο αιτίες.

Πολύ ασθενής: Εάν η αναλλοίωτη συνθήκη σας είναι πολύ ασθενής, τότε το τζίνι δεν σας έδωσε όλα όσα θα χρειαζόσασταν για να προχωρήσετε.

Πολύ ισχυρή: Εάν η αναλλοίωτη συνθήκη σας είναι πολύ ισχυρή, τότε δεν θα μπορέσετε να επιτύχετε την ισχύ της αρχικά, ούτε να τη διατηρήσετε στη συνέχεια.

Καμία αδήλωτη παραδοχή: Οι αναλλοίωτες συνθήκες δεν πρέπει να στερούνται λεπτομέρειας, ούτε να είναι τόσο ασθενείς που δεν θα μπορούν να προχωρήσουν στο επόμενο βήμα. Δεν πρέπει να υπάρχουν παραδοχές που δεν δηλώνετε. Για να είστε βέβαιοι, κάντε το γνωστό κόλπο και προσποιηθείτε ότι είστε εξωγήινοι και μόλις προσγειωθήκατε στην αρχή του βρόχου

χωρίς να γνωρίζετε *τίποτε* άλλο πέρα απ' όσα δηλώνονται στην αναλλοίωτη συνθήκη.

Παράδειγμα: Στον αλγόριθμο δύο δαχτύλων για την εύρεση του μέγιστου, η αναλλοίωτη συνθήκη περιέχει πράγματι κάποιες παραδοχές που δεν δηλώνονται. Θεωρεί ότι το δεξί σας δάχτυλο έχει ήδη συναντήσει τους αριθμούς που βρίσκονται από πάνω του, στη λίστα, αντίθετα απ' αυτούς που βρίσκονται από κάτω του. Ίσως ακόμα πιο σημαντικό είναι να είναι ξεκάθαρο αν θα θεωρήσουμε ότι το δάχτυλό σας έχει ήδη συναντήσει τον αριθμό που δείχνει τη δεδομένη στιγμή. Η αναλλοίωτη συνθήκη θεωρεί επίσης ότι οι αριθμοί στη λίστα δεν έχουν μεταβληθεί από τις αρχικές τιμές τους.

Έναστρη νύχτα: Πώς εμπνεύστηκε ο Βαν Γκονγκ το διάσημο έργο του *Έναστρη νύχτα*; Η απάντηση δεν είναι εύκολη. Ομοίως, η εύρεση των αναλλοίωτων συνθηκών και των αλγόριθμων είναι μια μορφή τέχνης.

Εφαρμόστε αυτή τη διαδικασία: Μην καταλήξετε στην αναλλοίωτη συνθήκη κατόπιν εορτής. Χρησιμοποιήστε τη, εξαρχής, για να σχεδιάσετε τον αλγόριθμό σας.

5) ***Κύρια βήματα:*** Ο ψευδοκώδικας $κώδικας_{βρόχος}$ πρέπει να ορίζεται έτσι ώστε να μπορείτε να τον ακολουθήσετε από το σημείο όπου πιστεύετε ότι βρίσκεται ο υπολογισμός, αλλά και από οποιαδήποτε άλλη κατάσταση της δομής δεδομένων για την οποία ισχύει η αναλλοίωτη συνθήκη και η συνθήκη εξόδου δεν έχει ικανοποιηθεί ακόμα.

Να ανησυχείτε για *ένα βήμα κάθε φορά*. Μην παρασύρεστε από την έντονη επιθυμία να κατανοήσετε όλο τον υπολογισμό από την αρχή. Γενικά, αυτό είναι κάτι που μπορεί μόνο να σας τρομάξει και να σας απογοητεύσει. Θα αναφέρω εδώ τα σοφά λόγια των βουδιστών και των προγραμμάτων των δώδεκα βημάτων: Σήμερα, θα νιώσετε σαν να βρεθήκατε, με κάποιον άγνωστο σ' εσάς τρόπο, σε μια ξένη πόλη. Μην ανησυχείτε για το παρελθόν ή το μέλλον. Είστε όμως στον σωστό δρόμο. Στόχος σας πρέπει να είναι να κάνετε ένα βήμα κάθε φορά ώστε να προχωράτε προς τα εμπρός και να παραμείνετε στον δρόμο. Μπορείτε επίσης να φανταστείτε ότι συμμετέχετε σε μια σκυταλοδρομία και ένας συναθλητής σας σας δίνει τη σκυτάλη. Εσείς απλώς πρέπει να τη μεταφέρετε κάνοντας έναν γύρο στον στίβο και να τη δώσετε στον επόμενο συμπαίκτη σας.