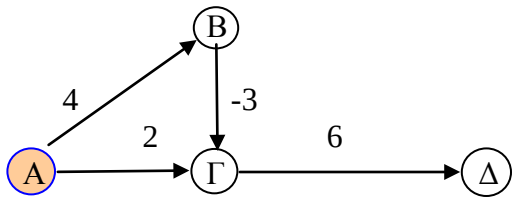
Φροντιστήριο 10 – Σκελετοί Λύσεων

Άσκηση 1

(α)

	d(s)	d(A)	d(B)	d(Γ)	d(Δ)	d(E)
∅	**0**	∞	∞	∞	∞	∞
{s}	**0**	**13**	∞	**2**	∞	∞
{s,Γ}	**0**	**13**	**3**	**2**	**7**	∞
{s,Γ,B}	**0**	**9**	**3**	**2**	**7**	**11**
{s,Γ,B,Δ}	**0**	**9**	**3**	**2**	**7**	**9**
{s,Γ,B,Δ,A}	**0**	**9**	**3**	**2**	**7**	**9**
{s,Γ,B,Δ,A,E}	**0**	**9**	**3**	**2**	**7**	**9**

(β) Ο πιο κάτω γράφος δείχνει την ακαταλληλότητα του αλγόριθμου του Dijkstra σε γράφους με αρνητικά βάρη. Το πρόβλημα οφείλεται στο γεγονός ότι η ύπαρξη αρνητικών βαρών επιτρέπει σενάρια στα οποία ενώ μια κορυφή έχει ήδη υποστεί επεξεργασίας (π.χ. η Γ στο παράδειγμα), να διαπιστωθεί βραχύτερο μονοπάτι προς αυτήν μέσω ακμών με αρνητικό βάρος (το ΑΒΓ στο παράδειγμα). Αυτό έχει ως δυνατή συνέπεια, βραχύτερα μονοπάτια προς τους γείτονες της συγκεκριμένης κορυφής να μην γίνουν αντιληπτά.



	d(A)	d(B)	d(Γ)	d(Δ)
∅	**0**	∞	∞	∞
{A}	**0**	**4**	**2**	∞
{A,Γ}	**0**	**4**	**2**	**8**
{A,Γ,B}	**0**	**4**	**1**	**8**
{A,Γ,B,Δ}	**0**	**4**	**1**	**8**

Έτσι ο αλγόριθμος δίνει d(Δ)=8 ενώ θα έπρεπε d(Δ)=7.

Άσκηση 2

Θεωρούμε τους αριθμούς του συνόλου και συγκεκριμένα την ακολουθία θέσεων που δοκιμάστηκαν πριν την εισαγωγή τους στον πίνακα:

9	:	9, 10
21	:	8, 9
8	:	8
25	:	12, 0, 1, 2
26	:	0, 1
12	:	12
13	:	0

Αυτό υποδεικνύει ότι τα στοιχεία 9, 21, 8 εισήχθησαν με τη σειρά

8, 21, 9

ενώ τα στοιχεία 25, 26, 12, 13 εισήχθησαν με τη σειρά

το 13 πριν από το 26

το 25 μετά από τα 26, 12, 13.

Κατά συνέπεια:

(α) Ο αριθμός 25 εισήχθηκε τελευταίος σε κάποιες αλλά όχι κατ'ανάγκη όλες τις σειρές εισαγωγής των στοιχείων που θα μπορούσαν να δημιουργήσουν τον εν λόγω πίνακα. (Για παράδειγμα, η σειρά εισαγωγών 26, 12, 13, 25, 8, 21, 9, δημιουργεί τον πίνακα αλλά το στοιχείο 25 δεν εισάγεται τελευταίο.) Επομένως, ισχύει η πρόταση (iii).

(β) Τουλάχιστον τρεις αριθμοί εισήχθηκαν πριν από τον αριθμό 25 σε όλες τις σειρές εισαγωγής στοιχείων που θα μπορούσαν να δημιουργήσουν τον εν λόγω πίνακα. Είναι οι 26, 12, 13. Επομένως ισχύει το (i).

(γ) Ο αριθμός 8 εισήχθηκε πριν από τον αριθμό 12 σε κάποιες αλλά όχι κατ'ανάγκη σε όλες τις σειρές στοιχείων που θα μπορούσαν να δημιουργήσουν τον εν λόγω πίνακα. Για παράδειγμα η σειρά εισαγωγών 26, 12, 13, 25, 8, 21, 9, δημιουργεί τον πίνακα αλλά το στοιχείο 12 εισάγεται πριν από το στοιχείο 8. Επομένως ισχύει η πρόταση (iii).

(δ) Οποιαδήποτε σειρά εισαγωγής των στοιχείων όπου ο αριθμός 9 εισάγεται πριν από τον αριθμό 8 θα τοποθετήσει το στοιχείο 9 στη θέση 9 και όχι στη θέση 10. Αυτό οφείλεται στο γεγονός ότι ακόμα και αν η εισαγωγή προβλέπει εισαγωγή του 21 πριν από το 9, το 21 θα καταλάβει τη θέση 8, και η θέση 9 θα παραμείνει στη διάθεση του στοιχείου 8. Επομένως ισχύει η πρόταση (ii).

Άσκηση 3

Θεωρήστε τις εισαγωγές των στοιχείων 13, 1, 3, 4, 9, 10, και 12. Μετά επιχειρήστε εισαγωγή του στοιχείου 0. Για την εισαγωγή αυτή γίνεται μία αλυσίδα ανεπιτυχών προσπαθειών για εύρεση κενής θέσης ως εξής:

f(0,0) = 0	f(0,1) = 1
f(0,2) = 4	f(0,3) = 9
f(0,4) = 3	f(0,5) = 12
f(0,6) = 10	f(0,7) = 10
f(0,8) = 12	f(0,9) = 3
…	

Αυτή η αλυσίδα δημιουργεί την υποψία ότι η διαδικασία εισαγωγής δεν θα καταφέρει να εντοπίσει μια από τις κενές θέσεις του πίνακα, δηλαδή, ότι, για κάθε i >= 0,

$f(0,i) \in \{0, 1, 3, 4, 9, 10, 12\}$, ή
$i^2 \bmod 13 \in \{0, 1, 3, 4, 9, 10, 12\}$, ή
υπάρχουν d και r τέτοια ώστε $i^2 = 13d + r$, όπου $r \in \{0, 1, 3, 4, 9, 10, 12\}$.

Για την απόδειξη της εικασίας αρχικά παρατηρούμε ότι κάθε θετικός ακέραιος i μπορεί να γραφτεί ως:

$$i = 13k + a, \qquad 0 \le a \le 12$$

Επομένως,

$$i^2 = 13\cdot(13k^2 + 2ak) + a^2.$$

Τέλος, υπολογίζοντας την τιμή a^2 για $0 \le a \le 12$, παρατηρούμε ότι

$$a^2 \bmod 13 \in \{0, 1, 3, 4, 9, 10, 12\}$$

που δίνει το επιθυμητό αποτέλεσμα.

(Για παράδειγμα, για a = 9:

$i^2 = 13\cdot(k^2 + 2ak) + 81$
$= 13\cdot(k^2 + 2ak) + 13\cdot 6 + 3$
$= 13\cdot(k^2 + 2ak + 6) + 3$

που σημαίνει πως για κάθε $i = 13k + 9$, $i^2 \bmod 13 = 3$.
)

Άσκηση 4

Χρησιμοποιούμε τις δομές:

```
struct hashtable                          struct node{
     struct node table[maxsize];               int data;
     int maxsize;                              int status;
     int size;                            }
}
```

Χρησιμοποιούμε το πεδίο status για να σημάνουμε την "κατάσταση" κάθε θέσης. Αν η θέση είναι κενή έχουμε status = -1, αν είναι deleted έχουμε status = 0, και, αν είναι γεμάτη θέτουμε την τιμή του status ως την τιμή 1. Οι πράξεις υλοποιούνται ως εξής:

```
Insert(struct hashtable h, int x){
 if h->maxsize == h->size
            report "Table is full"
 i = 0;
 t = x mod h->maxsize;
 k = t;
 while (h->table[k]->status == 1)
     i++;
     k = (t + 3i + 5i²) mod h->maxsize;

 h->table[k]->data = x;
 h->table[k]->status = 1;
 h->size++;
}

int Find(struct hashtable h, int x){
 i = 0;
 t = x mod h->maxsize;
 k = t;
 while (h->table[k]->status != 1 && i < MAX)
     temp =  h->table[k]->data;
     if (temp == x)
          return k;
     else
          i++;
          k = (t + 3i + 5i²) mod h->maxsize;
 return -1;
}

Delete(struct hashtable h, int x){
 k = Find(h, x);
 if (k == -1) return;
 else h->table[k]->status = 0;
}
```