Αν. Καθηγητής Π. Λουρίδας

Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας

Οικονομικό Πανεπιστήμιο Αθηνών

Ο Samuel Beckett και οι Κώδικες Gray

Το γνωστότερο έργο του Ιρλανδού συγγραφέα Samuel Beckett (1906–1989) βραβείο Νόμπελ λογοτεχνίας 1969), είναι το θεατρικό έργο «Περιμένοντας τον Γκοντό». Ο Beckett, ο οποίος έζησε στο Παρίσι το μεγαλύτερο κομμάτι της ζωής του, το έγραψε στα γαλλικά το 1952 (En attendant Godot) και στα αγγλικά το 1954 (Waiting for Godot). Σε αυτό οι δύο κύριοι χαρακτήρες του έργου, ο Βλαντιμίρ και ο Εστραγκόν, περιμένουν την έλευση του Γκοντό. Δεν ξέρουμε γιατί τον περιμένουν, περιμένοντας συζητούν, συναντούν άλλους, το έργο τελειώνει χωρίς να έχει έρθει ο Γκοντό· θα συνεχίσουν να τον περιμένουν.

Εδώ θα ασχοληθούμε όμως με ένα άλλο, λιγότερο γνωστό έργο του Beckett, το οποίο έγραψε για την τηλεόραση. Στο έργο αυτό, το οποίο ονομάζεται Quad, εμφανίζονται τέσσερεις χαρακτήρες. Ο Beckett ήθελε όμως να εμφανίζονται με συγκεκριμένο τρόπο στη σκηνή: διαδοχικά να μπαίνει ή να βγαίνει ένας μόνο χαρακτήρας και αν ένας χαρακτήρας βγαίνει να είναι αυτός που έχει μείνει περισσότερη ώρα στη σκηνή. Επιπλέον ήθελε με τον τρόπο αυτό να εμφανιστούν στη σκηνή μία φορά όλοι οι δυνατοί συνδυασμοί των τεσσάρων χαρακτήρων, μία φορά ακριβώς.

Ο Beckett δεν βρήκε τρόπο να το πετύχει αυτό. Πράγματι, δεν γίνεται να επιτευχθεί μια τέτοια αλληλουχία εισόδων και εξόδων για τέσσερα άτομα, χωρίς να υπάρχουν κάποιες επαναλήψεις στους συνδυασμούς που εμφανίζονται.

Στο σημείο αυτό ας επιστρέψουμε στην Πληροφορική. Ο κώδικας Gray (Gray code) n bits είναι μία κυκλική διάταξη των δυαδικών αριθμών με n bits έτσι ώστε δύο διαδοχικοί αριθμοί να διαφέρουν κατά ένα μόνο ψηφίο (bit). Για παράδειγμα, ο κώδικας Gray για $n = 4$ είναι ο:

0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000.

Προσέξτε ότι ο τελευταίος αριθμός διαφέρει όχι μόνο από τον προηγούμενό του, αλλά και από τον πρώτο κατά ένα bit, κατά συνέπεια ο κώδικας είναι πράγματι κυκλικός.

Είναι εύκολο να κατασκευάσουμε έναν τέτοιο κώδικα με n bits, δουλεύοντας αναδρομικά. Αν με Γ_n συμβολίσουμε τον κώδικα Gray με n bits, και Γ_0 είναι απλώς η κενή συμβολοσειρά, τότε, για να κατασκευάσουμε τον κώδικα Gray με $n + 1$ bits αρκεί να πάρουμε τον κώδικα Gray με n bits, να βάλουμε 0 πριν από κάθε συμβολοσειρά του, να τον αναστρέψουμε και να βάλουμε 1 πριν από συμβολοσειρά του στην ανάστροφη σειρά. Πράγματι:

$$\Gamma_1 = 0, 1$$

Απλώς βάλαμε 0 και 1 πριν από την κενή συμβολοσειρά. Στη συνέχεια:

$$\Gamma_2 = 00, 01, 11, 10$$

Βάλαμε 0 πριν από το $0, 1$ και 1 πριν από την ανάστροφη σειρά $1, 0$.

Ομοίως:

$$\Gamma_3 = 000, 001, 011, 010, 110, 111, 101, 100$$

Βάλαμε 0 πριν από το $00, 01, 11, 10$ και 1 πριν από το $10, 11, 01, 00$.

Από τον τρόπο κατασκευής του, αυτός ο κώδικας ονομάζεται, αν θέλουμε να είμαστε πιο ακριβείς, Κατοπτρικός Δυαδικός Κώδικας (Reflected Binary Code, RBC), ή Κατοπτρικός Δυαδικός (Reflected Binary, RB).

Μπορούμε όμως να παρατηρήσουμε ότι μπορεί να έχουμε έναν κώδικα που να πληρεί τη συνθήκη της αλλαγής ενός μόνο δυαδικού ψηφίου κάθε φορά χωρίς να κατασκευάζεται με αυτόν τον τρόπο. Για παράδειγμα, στον *ισορροπημένο κώδικα Gray* (balanced Gray code) θέλουμε να έχουμε τον ίδιο αριθμό αλλαγών για κάθε bit. Παρακάτω μπορείτε να δείτε έναν ισορροπημένο κώδικα Gray 4 bits:

0	1	1	1	1	1	1	0	0	0	0	0	0	1	1	0
0	0	1	1	1	1	0	0	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0
0	0	0	1	1	0	0	0	0	0	1	1	1	1	1	1

Κάθε στοιχείο του κώδικα αντιστοιχεί σε μία στήλη. Μπορείτε να διαπιστώσετε ότι σε κάθε γραμμή αλλάζουν ακριβώς τέσσερα bits.

Χρησιμοποιώντας άλλα κριτήρια μπορούμε να δημιουργήσουμε και άλλους κώδικες Gray. Μπορούμε επίσης να χαλαρώσουμε την απαίτηση να είναι ο κώδικας κυκλικός· αν δεν είναι, θα λέμε ότι ο κώδικας είναι ένα μονοπάτι, αλλά όχι ένας κύκλος.

Αν ως κριτήριο υιοθετήσουμε ότι θέλουμε όταν αλλάζουμε ένα bit από ένα σε μηδέν, αυτό να είναι το bit το οποίο έχει παραμείνει για περισσότερο χρόνο στην τιμή ένα, τότε ο κώδικας που προκύπτει πληροί το κριτήριο που ήθελε ο Beckett—και ως εκ τούτου, ο κώδικας αυτός ονομάζεται κώδικας Beckett-Gray.

Σε περίπτωση που έχουμε $n = 3$, δεν υπάρχει κώδικας Becket-Gray, γιατί δεν μπορεί να βρεθεί ένας κυκλικός κώδικας με τη συνθήκη που θέλουμε, υπάρχει όμως ένα και μοναδικό μονοπάτι:

$$
\begin{array}{cccccccc}
0 & \boxed{1} & 1 & \boxed{0} & 0 & 0 & 1 & 1 \\
0 & 0 & \boxed{1} & 1 & 1 & \boxed{0} & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
$$

Μπορείτε να επιβεβαιώσετε ότι το bit που σβήνει σε μία στήλη είναι αυτό που έχει μείνει τον περισσότερο χρόνο αναμμένο στη γραμμή του.

Για $n = 4$ και πάλι δεν υπάρχει κώδικας Beckett-Gray, και έτσι ο Beckett δεν μπόρεσε να βρει κάποιον για το έργο του. Υπάρχουν όμως τέσσερα μονοπάτια:

$$
\begin{array}{cccccccccccccccc}
0 & \boxed{1} & 1 & \boxed{0} & 0 & 0 & 0 & 0 & \boxed{1} & 1 & 1 & 1 & 1 & \boxed{0} & 0 & 1 \\
0 & 0 & \boxed{1} & 1 & 1 & \boxed{0} & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & \boxed{1} & 1 & 1 & \boxed{0} & 0 & \boxed{1} & 1 & 1 & 1 & 1 & \boxed{0} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 1 & 1 & 1 & \boxed{0} & 0 & 1 & 1 & 1 & 1
\end{array}
$$

$$
\begin{array}{cccccccccccccccc}
0 & \boxed{1} & 1 & \boxed{0} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & \boxed{1} & 1 & 1 & \boxed{0} & 0 & 0 & \boxed{1} & 1 & 1 & 1 & \boxed{0} & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & \boxed{1} & 1 & 1 & \boxed{0} & 0 & \boxed{1} & 1 & 1 & 1 & 1 & \boxed{0} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 1 & 1 & 1 & 1 & \boxed{0} & 0 & 1 & 1 & 1
\end{array}
$$

$$
\begin{array}{cccccccccccccccc}
0 & \boxed{1} & 1 & 1 & 1 & \boxed{0} & 0 & 0 & \boxed{1} & 1 & 1 & \boxed{0} & 0 & 0 & 0 & 1 \\
0 & 0 & \boxed{1} & 1 & 1 & 1 & \boxed{0} & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & \boxed{1} & 1 & 1 & 1 & \boxed{0} & 0 & \boxed{1} & 1 & 1 & 1 & \boxed{0} & 0 & 0 \\
0 & 0 & 0 & 0 & \boxed{1} & 1 & 1 & 1 & 1 & 1 & \boxed{0} & 0 & 0 & 0 & 1 & 1
\end{array}
$$

$$
\begin{array}{cccccccccccccccc}
0 & \boxed{1} & 1 & 1 & 1 & \boxed{0} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & \boxed{1} & 1 & 1 & 1 & \boxed{0} & 0 & \boxed{1} & 1 & 1 & \boxed{0} & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & \boxed{1} & 1 & 1 & 1 & \boxed{0} & 0 & 0 & \boxed{1} & 1 & 1 & 1 & \boxed{0} & 0 \\
0 & 0 & 0 & 0 & \boxed{1} & 1 & 1 & 1 & 1 & \boxed{0} & 0 & 0 & 0 & 1 & 1 & 1
\end{array}
$$

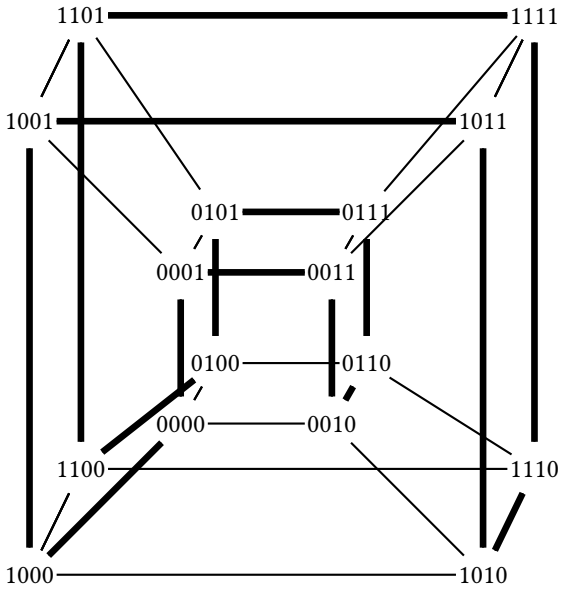
Για $n = 5$ υπάρχουν 16 κώδικες Beckett-Gray. Στο σημείο αυτό, ας εισάγουμε έναν πιο πρακτικό τρόπο αναπαράστασης κωδικών και μονοπατιών Gray. Αφού κάθε φορά μόνο ένα bit αλλάζει, για την αναπαράσταση ενός κώδικα ή ενός μονοπατιού αρκεί να δώσουμε τη σειρά των bits που αλλάζουν, όπου κάθε bit αναπαρίσταται από τη θέση του, μετρώντας από τα δεξιά. Αυτή ονομάζεται *σειρά δέλτα* (delta sequence) ή *σειρά μεταβάσεων* (transition sequence) και τη συμβολίζουμε με δ. Έτσι, για τον κύκλο Gray με 1 bit έχουμε $\delta = 00$, και για τον κύκλο Gray με 2 bits έχουμε $\delta = 0101$.

Προχωρόντας σε $n = 3$, για τον κύκλο Gray που προκύπτει κλείνοντας το Γ_3 έχουμε $\delta = 01020102$, ενώ υπάρχει και άλλος κύκλος Gray (που δεν είναι κατοπτρικός) με $\delta = 01210121$.

Χρησιμοποιώντας σειρές δέλτα, οι 16 κώδικες Beckett-Gray για $n = 5$ είναι οι:

0102013201043210434213234041230 4	0102031240302404123241401323 4013
0102031420302404123421410323 4103	0102031420324042103421413032 4103
0102034120234314232014320104 3104	0102341203240304123034101234 0124
0120132140231434023213402143 1041	0120304123031404321040320241 3241
0120310421304342131034210430 2402	0123012143021434023034142031 4121
0123012423414023141034320143 4204	0123040123134041323420234102 4212
0123040123243042313410143201 4121	0123041232043412034301431204 1323
0123401023243012431340143201 4121	0123401023243020143201413241 3141

Οι κώδικες Gray έχουν και μία ερμηνεία μέσω θεωρίας γράφων και γεωμετρίας. Αν φτιάξουμε ένα γράφο όπου οι κορυφές είναι τα στοιχεία του κώδικα και οι σύνδεσμοι συνδέουν δύο στοιχεία αν διαφέρουν κατά ένα μόνο bit, τότε τα δυνατά μονοπάτια και κύκλοι είναι οι δυνατοί τρόποι να εξερευνήσουμε όλο το γράφο περνώντας από κάθε κορυφή μία μόνο φορά. Ένα μονοπάτι που επισκέπτεται όλες τις κορυφές ενός γράφου μία φορά ονομάζεται *Χαμιλτόνιο μονοπάτι* (Hamiltonian path) και αντίστοιχα ένας κύκλος που επισκέπτεται όλες τις κορυφές ενός γράφου ονομάζεται *Χαμιλτόνιος κύκλος* (Hamiltonian cycle). Δεδομένου ότι κάθε κορυφή του γράφου αυτού έχει $n-1$ ακριβώς γείτονες, γεωμετρικά ένας τέτοιος γράφος είναι ένας *υπερκύβος* (hypercube). Παρακάτω μπορείτε να δείτε τον υπερκύβο για $n = 4$ και ένα Χαμιλτόνιο κύκλο πάνω σε αυτόν. Ο Χαμιλτόνιος κύκλος αντιστοιχεί στον κατοπτρικό κώδικα Gray.



Αν θέλουμε να βρούμε όλους τους κώδικες Gray, αρκεί να βρούμε όλα τα μονοπάτια και κύκλους πάνω στον αντίστοιχο γράφο (ή υπερκύβο). Πλην όμως, κάποιοι από τους κώδικες προκύπτουν από άλλους απλώς αλλάζοντας την θέση των bits μέσα σε αυτούς. Για να το δούμε αυτό, ας ονομάσουμε τα στοιχεία μιας σειράς δέλτα *συντεταγμένες*: για $n = 3$, οι συντεταγμένες είναι τα $0, 1, 2$. Επιπλέον ας συμβολίσουμε με $d(n)$ τον αριθμό των διαφορετικών σειρών δέλτα για ένα συγκεκριμένο n. Επίσης, ας συμβολίσουμε με $c(n)$ τον αριθμό των «κανονικών» (canonical) σειρών δέλτα, οι οποίες χαρακτηρίζονται από την ιδιότητα ότι κάθε συντεταγμένη k εμφανίζεται στη σειρά πριν από την πρώτη εμφάνιση της συντεταγμένης $k + 1$. Τότε αφού κάθε μετάθεση (permutation) των συντεταγμένων σε μία σειρά δέλτα παράγει μια άλλη σειρά δέλτα, έχουμε $d(n) = n!c(n)$. Πράγματι, είδαμε ότι για τον κώδικα Γ_3 έχουμε $\delta = 01020102$. Αν εφαρμόσουμε την μετάθεση:

$$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \end{pmatrix}$$

παίρνουμε $\delta = 10121012$ που μπορείτε να διαπιστώσετε ότι είναι επίσης κώδικας Gray. Αν ένας κώδικας Gray προκύπτει από έναν άλλο κώδικα Gray μέσω μετάθεσης, λέμε ότι οι δύο αυτοί κώδικες είναι *ισομορφικοί* (isomorphic). Για να βρούμε συνεπώς όλους τους μη ισομορφικούς κώδικες Gray, θα πρέπει να εξασφαλίσουμε ότι στην εξερεύνηση για την εύρεση των Χαμιλτόνιων μονοπατιών και κύκλων οι διαδρομές που επιλέγουμε ακολουθούν μόνο κανονικές σειρές δέλτα.

Ξέρουμε ότι για να εξερευνήσουμε έναν γράφο μπορούμε να χρησιμοποιήσουμε την κατά βάθος αναζήτηση. Εδώ λοιπόν θα χρησιμοποιήσουμε μια παραλλαγή της κατά βάθος αναζήτησης, η οποία θα έχει τα εξής χαρακτηριστικά:

- Η εξερεύνηση θα προχωράει από κόμβο σε κόμβο του υπερκύβου.
- Σε κάθε κόμβο που βρισκόμαστε, καταγράφουμε ποια είναι η μέγιστη συντεταγμένη που μπορούμε να αλλάξουμε.
- Οι γείτονες του κόμβου προκύπτουν από τον τρέχοντα κόμβο αλλάζοντας κάθε bit από το ελάχιστα σημαντικό (το πρώτο εκ των δεξιών, στη θέση μηδέν), μέχρι τη θέση της μέγιστης συντεταγμένης.
- Στην παραδοσιακή κατά βάθος αναζήτηση, σημειώνουμε κάθε κόμβο που επισκεπτόμαστε ώστε να μην τον επισκεπτούμε ξανά. Με τον τρόπο αυτό εξερευνούμε τον γράφο μία φορά. Εδώ όμως θέλουμε να τον εξερευνήσουμε εξονυχιστικά, βρίσκοντας όλα τα δυνατά μονοπάτια που πληρούν τις απαιτήσεις μας. Επομένως θα πρέπει να λάβουμε πρόνοια ώστε αφού έχουμε εξερευνήσει ένα μονοπάτι να μπορούμε να ξανα-επισκεφτούμε τους κόμβους του μονοπατιού (με άλλη σειρά πλέον).

Τα παραπάνω μπορούμε να τα επιτύχουμε με τον αλγόριθμο που ακολουθεί. Ο αλγόριθμος αυτός κτίζει διαδοχικά κώδικες Gray χρησιμοποιώντας τη στοίβα *gc* και τους συλλέγει όλους στη λίστα *all_codes*, η οποία αρχικά είναι άδεια. Για να τον καλέσουμε, πρέπει να έχει αρχικοποιηθεί ο πίνακας *visited* ώστε όλα τα στοιχεία του να είναι FALSE, εκτός του *visited*$[0]$ που θα είναι TRUE, και η στοίβα *gc* να περιέχει το στοιχείο 0. Στις παραμέτρους του αλγορίθμου βρίσκεται και το βάθος της αναδρομής, d, προκειμένου να ορίσουμε τη συνθήκη διακοπής της αναδρομής—όταν έχουμε επισκεφτεί όλους τους 2^n κόμβους του υπερκύβου. Ο αλγόριθμος χρησιμοποιεί τη συνάρτηση Flip(x, i), η οποία αλλάζει την τιμή του bit i μέσα στο x.

```
GC_DFS(d, x, max_coord, n, gc)
    Input: d, the recursion depth
           x, the current node in the hypercube
           max_coord, the maximum coordinate to be set in x
           n, the number of bits of the Gray code
           gc, a stack
    Data: visited, an boolean array of size 2^n
          all_codes, a list
    Result: all_codes contains all the Gray codes that could be found
1   if d = 2^n then
2       AppendToList(all_codes, gc)
3       return
4   for i = 0 to Min(n − 1, max_coord + 1) do
5       Flip(x, i)
6       if not visited[x] then
7           visited[x] ← TRUE
8           Push(gc, x)
9           GC_DFS(d + 1, x, Max(i + 1, max_coord))
10          visited[x] ← FALSE
11          Pop(gc)
12      Flip(x, i)
```

Ο αλγόριθμος αυτός θα παράξει όλους τους κώδικες Gray. Εμείς όμως θέλουμε να μπορούμε να βρούμε επιπλέον συγκεκριμένα τους κώδικες Beckett-Gray. Για να το πετύχουμε αυτό, αρκεί να τον αλλάξουμε λίγο, χρησιμοποιώντας μια ουρά, την οποία θα χρησιμοποιήσουμε για να ξέρουμε ότι όταν μηδενίζουμε ένα bit, αυτό είναι το bit που έχει μείνει για περισσότερο χρόνο στην τιμή ένα.

Με τον αλγόριθμο αυτό εξασφαλίζουμε ότι θα βρούμε κώδικες Gray που δεν είναι ισομορφικοί μέσω μετάθεσης. Μπορούμε όμως να επεκτείνουμε τον ορισμό του ισομορφισμού μεταξύ κωδικών Gray και για κώδικες οι οποίοι μπορούν να προκύψουν ο ένας από τον άλλο αν μπορούμε να χρησιμοποιήσουμε και αναστροφή εκτός από αντιμετάθεση. Για παράδειγμα, ας δούμε τον πρώτο κώδικα Beckett-Gray για $n = 5$ που είδαμε προηγουμένως:

$$01020132010432104342132340412304$$

Αυτός είναι ισομορφικός με τον:

$$01234010232430201432014132413141$$

Πράγματι, αν τον αντιστρέψουμε αυτόν, παίρνουμε:

14131423141023410203423201043210

Αλλά ο:

01020132010432104342132340412304

δίνει τον:

14131423141023410203423201043210

με τη μετάθεση:

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 & 0 \end{pmatrix}$$

Σκοπός της εργασίας είναι να φτιάξετε ένα πρόγραμμα το οποίο να κατασκευάζει διαφόρων ειδών κώδικες Gray και να βρίσκει και τυχόν ισομορφισμούς που προκύπτουν από αντιστροφές και μεταθέσεις.

Απαιτήσεις Προγράμματος

Κάθε φοιτητής θα εργαστεί σε αποθετήριο στο GitHub. Για να αξιολογηθεί μια εργασία θα πρέπει να πληροί τις παρακάτω προϋποθέσεις:

- Για την υποβολή της εργασίας θα χρησιμοποιηθεί το ιδιωτικό αποθετήριο του φοιτητή που δημιουργήθηκε για τις ανάγκες του μαθήματος και του έχει αποδοθεί. Το αποθετήριο αυτό έχει όνομα του τύπου `username-algo-assignments`, όπου `username` είναι το όνομα του φοιτητή στο GitHub. Για παράδειγμα, το σχετικό αποθετήριο του διδάσκοντα θα ονομαζόταν `louridas-algo-assignments` και θα ήταν προσβάσιμο στο https://github.com/dmst-algorithms-course/louridas-algo-assignments. *Τυχόν άλλα αποθετήρια απλώς θα αγνοηθούν.*
- Μέσα στο αποθετήριο αυτό θα πρέπει να δημιουργηθεί ένας κατάλογος `assignment-2021-3`.
- Μέσα στον παραπάνω κατάλογο το πρόγραμμα θα πρέπει να αποθηκευτεί με το όνομα `beckett_gray.py`.
- Δεν επιτρέπεται η χρήση έτοιμων βιβλιοθηκών γράφων ή τυχόν έτοιμων υλοποιήσεων των αλγορίθμων, ή τμημάτων αυτών, εκτός αν αναφέρεται ρητά ότι επιτρέπεται.
- Επιτρέπεται η χρήση δομών δεδομένων της Python όπως στοίβες, λεξικά, σύνολα, κ.λπ.
- Επιτρέπεται η χρήση της βιβλιοθήκης `itertools`.

- Επιτρέπεται η χρήση της βιβλιοθήκης `argparse` ή της βιβλιοθήκης `sys` (συγκεκριμένα, της λίστας `sys.argv`) προκειμένου να διαβάσει το πρόγραμμα τις παραμέτρους εισόδου.
- Το πρόγραμμα θα πρέπει να είναι γραμμένο σε Python 3.

Το πρόγραμμα θα καλείται ως εξής (όπου `python` η κατάλληλη εντολή στο εκάστοτε σύστημα):

```
python beckett_gray.py [-a | -b | -u | -c | -p] [-r] [-f] [-m] number_of_bits
```

Η σημασία των παραμέτρων είναι η εξής:

- `-a`: εύρεση και εμφάνιση όλων των κωδίκων (κύκλων και μονοπατιών).
- `-b`: εύρεση και εμφάνιση μόνο κωδίκων Beckett-Gray.
- `-u`: εύρεση και εμφάνιση μόνο Beckett-Gray.
- `-c`: εύρεση και εμφάνιση μόνο των κυκλικών κωδίκων.
- `-p`: εύρεση και εμφάνιση μόνο των μονοπατιών Gray.
- `-r`: εύρεση και εμφάνιση ανάστροφων ισομορφισμών.
- `-f`: εμφάνιση της πλήρους δυαδικής μορφής του κάθε κώδικα.
- `-m`: εμφάνιση του κάθε κώδικα με τη μορφή πίνακα.
- `number_of_bits`: ο αριθμός των bits του κώδικα.

Για περισσότερες λεπτομέρειες για τη σημασία των παραμέτρων και την εμφάνιση της εξόδου του προγράμματος, δείτε τα παραδείγματα που ακολουθούν.

Παραδείγματα

Παράδειγμα 1

Αν ο χρήστης του προγράμματος δώσει:

```
python beckett_gray.py -a 3
```

ή απλώς:

```
python beckett_gray.py 3
```

το πρόγραμμα θα εμφανίσει στην έξοδο:

```
C 01020102
P 0102101
C 01210121
```

δηλαδή, βρέθηκαν τρεις κώδικες, τους οποίους εμφανίζουμε με τη σειρά δέλτα, δύο από αυτοί είναι κυκλικοί (πρόθεμα `C`) και ένας είναι απλώς μονοπάτι (πρόθεμα `P`).

Παράδειγμα 2

Αν ο χρήστης του προγράμματος δώσει:

```
python beckett_gray.py -b 5
```

το πρόγραμμα θα εμφανίσει στην έξοδο:

```
B 0102013201043210434213234041230 4
B 0102031240302404123241401323 4013
B 0102031420302404123421410323 4103
B 0102031420324042103421413032 4103
B 0102034120234314232014320104 3104
B 0102341203240304123034101234 0124
B 0120132140231434023213402143 1041
B 0120304123031404321040320241 3241
B 0120310421304342131034210430 2402
B 0123012143021434023034142031 4121
B 0123012423414023141034320143 4204
B 0123040123134041323420234102 4212
B 0123040123243042313410143201 4121
B 0123041232043412034301431204 1323
B 0123401023243012431340143201 4121
B 0123401023243020143201413241 3141
```

δηλαδή, βρέθηκαν 16 κώδικες Beckett-Gray (πρόθεμα `B`).

Παράδειγμα 3

Αν ο χρήστης του προγράμματος δώσει:

```
python beckett_gray.py -b 5 -r
```

το πρόγραμμα θα εμφανίσει στην έξοδο τα περιεχόμενα του αρχείου `bgc_5_isomorphic.txt`, δηλαδή όπως στο προηγούμενο παράδειγμα αλλά στο τέλος θα παραθέσει και τους ανάστροφους ισομορφισμούς.

Παράδειγμα 4

Αν ο χρήστης του προγράμματος δώσει:

```
python beckett_gray.py -c 4
```

το πρόγραμμα θα εμφανίσει στην έξοδο τα περιεχόμενα του αρχείου `gc_cycles_4.txt`.

Παράδειγμα 5

Αν ο χρήστης του προγράμματος δώσει:

```
python beckett_gray.py -b 3
```

το πρόγραμμα θα εμφανίσει στην έξοδο:

```
U 0102101
```

δηλαδή, ένα μονοπάτι (το μοναδικό όπως είδαμε για $n = 3$) που ικανοποιεί τη συνθήκη του Beckett, εξ ου και το πρόθεμα `U` (unfinished, αφού τελειώνει χωρίς να κλείσει τον κύκλο).

Παράδειγμα 6

Αν ο χρήστης του προγράμματος δώσει:

```
python beckett_gray.py -u 4
```

το πρόγραμμα θα εμφανίσει στην έξοδο:

```
U 010213202313020
U 010213212031321
U 012301202301230
U 012301213210321
```

Παράδειγμα 7

Αν ο χρήστης του προγράμματος δώσει:

```
python beckett_gray.py -b -f 5
```

το πρόγραμμα θα εμφανίσει στην έξοδο τα περιεχόμενα του αρχείου `bgc_5_full.txt`, δηλαδή εκτός από τη σειρά δέλτα θα δώσει και την πλήρη δυαδική αναπαράσταση για κάθε κώδικα.

Παράδειγμα 8

Αν ο χρήστης του προγράμματος δώσει:

```
python beckett_gray.py -u -m 4
```

το πρόγραμμα θα εμφανίσει στην έξοδο:

```
U 010213202313020
0 1 1 0 0 0 0 0 1 1 1 1 1 0 0 1
0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1
0 0 0 0 1 1 1 0 0 1 1 1 1 1 0 0
0 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1
U 010213212031321
0 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1
0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 1
0 0 0 0 1 1 1 0 0 1 1 1 1 1 0 0
0 0 0 0 0 0 1 1 1 1 1 0 0 1 1 1
U 012301202301230
0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 1
0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1
0 0 0 1 1 1 1 0 0 1 1 1 1 0 0 0
```

```
0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1
U 012301213210321
0 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1
0 0 1 1 1 1 0 0 1 1 1 0 0 0 0 1
0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0
0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1
```

δηλαδή, θα εμφανίσει κάτω από κάθε κώδικα την αναπαράστασή του σε μορφή πίνακα.

Στην προτελεταία νουβέλλα, Westward Ho, που έγραψε ο Beckett υπάρχει η πιο διάσημη φράση του συγγραφέα:

> Ever tried. Ever failed. No matter. Try again. Fail again. Fail better.

Καλή Επιτυχία!