Καθηγητής Π. Λουρίδας

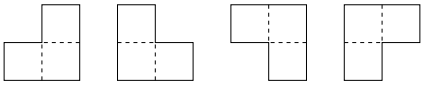
Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας

Οικονομικό Πανεπιστήμιο Αθηνών

Τρόμινο Ψηφιδωτό

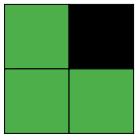
Έστω ότι θέλουμε να καλύψουμε μια επιφάνεια χρησιμοποιώντας πλακάκια, ή ψηφίδες, με συγκεκριμένα σχήματα. Θέλουμε να το πετύχουμε αυτό ώστε να μην υπάρχουν κενά στην επιφάνεια, και να μην υπάρχουν αλληλοκαλήψεις. Το πρόβλημα αυτό ονομάζεται *ψηφίδωση* ή *ψηφιδοθέτηση* (tesselation, tiling). Αναλόγως των σχημάτων των ψηφίδων, μπορούν να προκύψουν διαφορετικά ψηφιδωτά, μερικά από τα οποία δίνουν ένα κομψό αισθητικό αποτέλεσμα.

Εμείς θα ασχοληθούμε με την ψηφιδοθέτηση τετράγωνων επιφανειών, δηλαδή επιφανειών διαστάσεων $2^n \times 2^n$. Οι ψηφίδες μας θα έχουν το σχήμα L, που θα απαρτίζεται από τρία κομμάτια διαστάσεων 1×1, επομένως θα είναι τρόμινο. Τα τρόμινο θα μπορούμε να τα χρησιμοποιήσουμε σε τέσσερες προσανατολισμούς (που αντιστοιχούν σε περιστροφές 90 μοιρών):

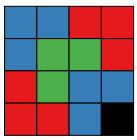


Αφού το κάθε τρόμινο αποτελείται από τρία τετράγωνα 1×1, δεν μπορεί η κάλυψη τετραγώνων $2^n \times 2^n$ να είναι τέλεια: στην καλύτερη περίπτωση, θα μένει ένα τετράγωνο κενό, θα έχουμε μια τρύπα. Αλλά δεν μπορούμε να το αποφύγουμε αυτό.

Στην περίπτωση του τετραγώνου $2^1 \times 2^1$, η κάλυψη είναι απλή: απλώς τοποθετούμε ένα τρόμινο πάνω στο τετράγωνο· με μαύρο σημειώνουμε την τρύπα.

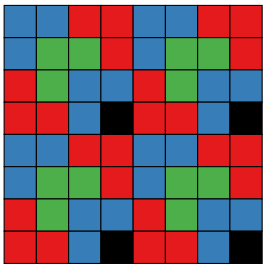


Στην περίπτωση του τετραγώνου $2^2 \times 2^2$, η κάλυψη επίσης μπορεί να γίνει εύκολα με πέντε τρόμινο.

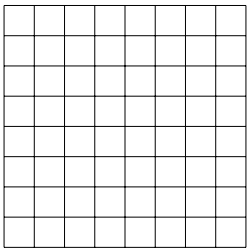


Πιο ενδιαφέρουσα είναι η γενικότερη περίπτωση του τετραγώνου $2^n \times 2^n$, με $n > 2$. Για να καλύψουμε το τετράγωνο αυτό, εργαζόμαστε αναδρομικά: διαιρούμε το τετράγωνο αυτό σε τέσσερα τεταρτημόρια μεγέθους $2^{n-1} \times 2^{n-1}$ το καθένα, και ξεκινάμε τη διαδικασία κάλυψης κάθε ενός από αυτά. Συνεχίζοντας τη διαδικασία αναδρομικά, θα φτάσουμε σε τεταρτημόρια μεγέθους $2^2 \times 2^2$, τα οποία ξέρουμε να τα καλύπτουμε.

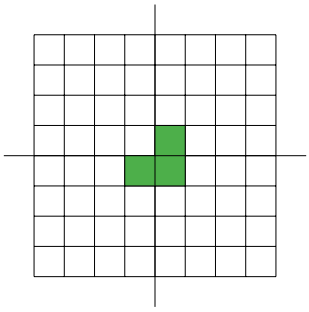
Αυτο σαν λογική στέκει, πλην όμως δεν μας εξασφαλίζει ότι τελικά στο τετράγωνο $2^n \times 2^n$ θα υπάρχει μόνο μια τρύπα. Για παράδειγμα, αν έχουμε ένα τετράγωνο $2^3 \times 2^3$, η αναδρομική διαδικασία που περιγράψαμε θα μας δώσει σαν αποτέλεσμα το παρακάτω:



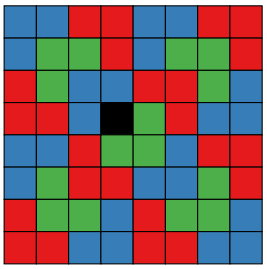
Επομένως χρειαζόμαστε κάτι επιπλέον στην αναδρομική διαδικαασία. Για να πετύχουμε την βέλτιστη κάλυψη, εργαζόμαστε ως εξής. Έστω ότι έχουμε ένα τετράγωνο $2^n \times 2^n$, όπως $2^3 \times 2^3$:



Τοποθετούμε ένα τρόμινο στη μέση.



Παρατηρούμε ότι τρία από τα τέσσερα τεταρτημόρια είναι *ελλειματικά* (deficient): τους λείπει ένα κομμάτι, ή μάλλον το κομμάτι αυτό είναι ήδη καλυμμένο. Επομένως κάθε ένα από αυτά μπορούμε να τα καλύψουμε πλήρως με τρόμινο, αρκεί να φροντίσουμε η τρύπα να πέφτει στο κομμάτι που είναι ήδη καλυμμένο! Έτσι θα μας μείνει μόνο μια τρύπα στο ένα τεταρτημόριο που δεν είναι ελλειματικό. Με τον τρόπο αυτό θα πάρουμε την παρακάτω κάλυψη:



Η κάλυψη μας έχει και ένα επιπλέον χαρακτηριστικό: χρησιμοποιούμε τρόμινο τριών χρωμάτων, έτσι ώστε κανένα τρόμινο να μην έχει κοινή πλευρά με τρόμινο του ίδιου χρώματος.

Η προσέγγιση αυτή δουλεύει όχι μόνο για τετράγωνα $2^3 \times 2^3$, αλλά για οποιοδήποτε τετράγωνο. Επιγραμματικά μπορούμε να την περιγράψουμε με τα ακόλουθα βήματα:

1. Αν $n = 1$, κάλυψε το τετράγωνο 2×2 και σταμάτα.
2. Αν $n = 2$, βάλε ένα πράσινο τρόμινο στη μέση, έτσι ώστε κάθε κομμάτι του να καλύπτει ένα μέρος του κάθε ενός 2×2 τετραγώνου που δεν είναι ελλειματικό. Βάλε γύρω-γύρω από το πράσινο τρόμινο εναλλάξ μπλε και κόκκινα τρόμινο, έτσι ώστε να καλυφθεί το ελλειματικό 2×2 τετράγωνο, πλην μιας τρύπας. Όπως είδαμε προηγουμένως:

   Σταμάτα.
3. Αν $n > 2$:
   - 3.1 Τοποθέτησε ένα πράσινο τρόμινο ώστε να αγκαλιάζει τη μέση.
   - 3.2 Χώρισε το $2^n \times 2^n$ τετράγωνο σε τέσσερα $2^{n-1} \times 2^{n-1}$ τετράγωνα που αντιστοιχούν σε τέσσερα τεταρτημόρια, αν πάρουμε τη μέση ως αρχή των αξόνων.

3.3 Κάλυψε αναδρομικά κάθε ένα $2^{n-1} \times 2^{n-1}$ τετράγωνο που προκύπτει από το αντίστοιχο τεταρτημόριο, λαμβάνοντας υπόψη ότι:

- Η τρύπα που αφήνει το πράσινο τρόμινο ορίζει ένα ελλειματικό $2^{n-1} \times 2^{n-1}$ τετράγωνο.
- Τα άλλα τρία $2^{n-1} \times 2^{n-1}$ τετράγωνα τα εκλαμβάνουμε επίσης ως ελλειματικά θεωρώντας ότι κάθε μέρος του τρόμινο αντιστοιχεί σε μία τρύπα.

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-2024-1`.
- Μέσα στον παραπάνω κατάλογο το πρόγραμμα θα πρέπει να αποθηκευτεί με το όνομα `tromino_tiling.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 των παρακάτω βιβλιοθηκών ή τμημάτων τους όπως ορίζεται:
  - `sys.argv`
  - `argparse`
- Το πρόγραμμα θα πρέπει να είναι γραμμένο σε Python 3.
- Η εργασία είναι αποκλειστικά ατομική. Δεν επιτρέπεται συνεργασία μεταξύ φοιτητών στην εκπόνησή της, με ποινή το μηδενισμό. Επιπλέον η εργασία δεν μπορεί να είναι αποτέλεσμα συστημάτων Τεχνητής Νοημοσύνης (όπως ChatGPT). Ειδικότερα όσον αφορά το τελευταίο σημείο προσέξτε ότι τα συστήματα αυτά χωλαίνουν στην αλγοριθμική σχέση, άρα τυχόν προτάσεις που κάνουν σε σχετικά θέματα μπορεί να είναι λανθασμένες. Επιπλέον, αν θέλετε

να χρησιμοποιήσετε τέτοια συστήματα, τότε αν και κάποιος άλλος το κάνει αυτό, μπορεί οι προτάσεις που θα λάβετε να είναι παρόμοιες, οπότε οι εργασίες θα παρουσιάσουν ομοιότητες και άρα θα μηδενιστούν.

- Η έξοδος του προγράμματος θα πρέπει να περιλαμβάνει μόνο ό,τι φαίνεται στα παραδείγματα που παρατίθενται. *Η φλυαρία δεν επιβραβεύεται.*

Χρήση του GitHub

Όπως αναφέρθηκε το πρόγραμμά σας για να αξιολογηθεί θα πρέπει να αποθηκευθεί στο GitHub. Επιπλέον, θα πρέπει το GitHub να χρησιμοποιηθεί καθόλη τη διάρκεια της ανάπτυξης του.

Αυτό σημαίνει ότι *δεν θα ανεβάσετε στο GitHub απλώς την τελική λύση του προβλήματος* μέσω της λειτουργίας "Upload files". Στο GitHub θα πρέπει να φαίνεται *το ιστορικό της συγγραφής του προγράμματος*. Αυτό συνάδει και με τη φιλοσοφία του εργαλείου: λέμε "commit early, commit often". Εργαζόμαστε σε ένα πρόγραμμα και κάθε μέρα, ή όποια στιγμή έχουμε κάνει κάποιο σημαντικό βήμα, αποθηκεύουμε την αλλαγή στο GitHub. Αυτό έχει σειρά ευεργετικών αποτελεσμάτων:

- Έχουμε πάντα ένα αξιόπιστο εφεδρικό μέσο στο οποίο μπορούμε να ανατρέξουμε αν κάτι πάει στραβά στον υπολογιστή μας (μας γλιτώνει από πανικούς του τύπου: ένα βράδυ πριν από την παράδοση ο υπολογιστής μας ή ο δίσκος του πνέει τα λοίσθια, και εμείς τι θα υποβάλουμε στον Λουρίδα;).
- Καθώς έχουμε πλήρες ιστορικό των σημαντικών αλλαγών και εκδόσεων του προγράμματός μας, μπορούμε να επιστρέψουμε σε μία προηγούμενη αν συνειδητοποιήσουμε κάποια στιγμή ότι πήραμε λάθος δρόμο (μας γλιτώνει από πανικούς του τύπου: μα το πρωί δούλευε σωστά, τι στο καλό έκανα και τώρα δεν παίζει τίποτε;).
- Καθώς φαίνεται η πρόοδος μας στο GitHub, επιβεβαιώνουμε ότι το πρόγραμμα δεν είναι αποτέλεσμα της όποιας επιφοίτησης (μας γλιτώνει από πανικούς του τύπου: καλά, πώς το έλυσες το πρόβλημα αυτό με τη μία, μόνος σου;).
- Αν δουλεύετε σε μία ομάδα *που βεβαίως δεν είναι καθόλου η περίπτωση μας εδώ*, μπορούν όλοι να εργάζονται στα ίδια αρχεία στο GitHub εξασφαλίζοντας ότι ο ένας δεν γράφει πάνω στις αλλαγές του άλλου. Παρά το ότι η εργασία αυτή είναι ατομική, καλό είναι να αποκτάτε τριβή με το εργαλείο git και την υπηρεσία GitHub μιας και χρησιμοποιούνται ευρέως και όχι μόνο στη συγγραφή κώδικα.

Άρα επενδύστε λίγο χρόνο στην εκμάθηση των git / GitHub, των οποίων ο σωστός τρόπος χρήσης είναι μέσω γραμμής εντολών (command line), ή ειδικών προγραμμάτων (clients) ή μέσω ενοποίησης στο περιβάλλον ανάπτυξης (editor, IDE).

Τελικό Πρόγραμμα

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

```
python tromino_tiling.py n
```

Η παράμετρος `n` υποδεικνύει το μέγεθος του τετραγώνου που θέλουμε να καλύψουμε, είναι δηλαδή ο εκθέτης n στην έκφραση $2^n \times 2^n$.

Σκοπός της εργασίας είναι η συγγραφή προγράμματος που θα εκτελεί την ψηφιδοποίηση τετραγώνου με τον αλγόριθμο που περιγράψαμε. *Υλοποιήσεις άλλων αλγορίθμων ψηφιδιποίησης δεν είναι αποδεκτές.*

Παραδείγματα

Παράδειγμα 1

Αν ο χρήστης του προγράμματος δώσει:

```
python tromino_tiling.py 1
```

το πρόγραμμά σας θα πρέπει να εμφανίσει στην έξοδο ακριβώς τα παρακάτω:

```
G X
G G
```

Το γράμμα `G` σημαίνει πράσινο, το γράμμα `X` σημαίνει μαύρο. Η έξοδος περιγράφει την κάλυψη ενός τετραγώνου 2×2 από ένα τρόμινο, αφήνοντας μια τρύπα.

Παράδειγμα 2

Αν ο χρήστης του προγράμματος δώσει:

```
python tromino_tiling.py 2
```

το πρόγραμμά σας θα πρέπει να εμφανίσει στην έξοδο ακριβώς τα παρακάτω:

```
B B R R
B G G R
R G B B
R R B X
```

Εδώ χρησιμοποιούνται και τα δύο επιπλέον χρώματα, με `B` το μπλε και με `R` το κόκκινο.

Παράδειγμα 3

Αν ο χρήστης του προγράμματος δώσει:

```
python tromino_tiling.py 3
```

το πρόγραμμά σας θα πρέπει να εμφανίσει στην έξοδο ακριβώς τα παρακάτω:

```
B B R R B B R R
B G G R B G G R
R G B B R R G B
R R B X G R B B
```

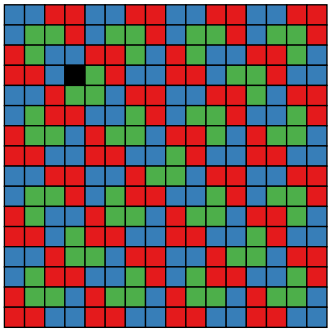
```
B B R G G B R R
B G R R B B G R
R G G B R G G B
R R B B R R B B
```

Παράδειγμα 4

Αν ο χρήστης του προγράμματος δώσει:

```
python tromino_tiling.py 4
```

το πρόγραμμά σας θα πρέπει να εμφανίσει στην έξοδο τα περιεχόμενα του αρχείου `tiling_16x16.txt`. Αυτό αντιστοιχεί στην κάλυψη:

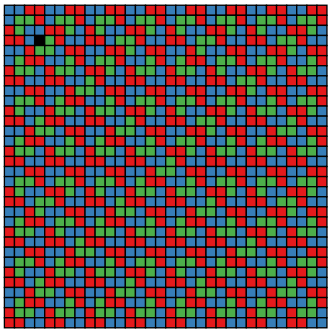


Παράδειγμα 5

Αν ο χρήστης του προγράμματος δώσει:

```
python tromino_tiling.py 5
```

το πρόγραμμά σας θα πρέπει να εμφανίσει στην έξοδο τα περιεχόμενα του αρχείου `tiling_32x32.txt`. Αυτό αντιστοιχεί στην κάλυψη:



Περισσότερες Πληροφορίες

Για την ψηφίδωση ή ψηφιδοθέτηση, μπορείτε να δείτε το σχετικό άρθρο στη Wikipedia. Ο αλγόριθμος που περιγράψαμε παρουσιάστηκε από τους Chu και Johnsonbaugh στο [1]. Τα τρόμινο είναι ειδική περίπτωση των πολυόμινο (polynimos), τα οποία εισήγαγε ο Solomon W. Golomb το 1954 [2], ο οποίος συνέγραψε στη συνέχεια και σχετικό βιβλίο [3].

Βιβλιογραφία

[1] I-Ping Chu and Richard Johnsonbaugh. "Tiling and recursion". In: *Proceedings of the Eighteenth SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '87. St. Louis, Missouri, USA: Association for Computing Machinery, 1987, pp. 261–263. ISBN: 0897912179. DOI: 10.1145/31820.31770. URL: https://doi.org/10.1145/31820.31770.

[2] S. W. Golomb. "Checker Boards and Polyominoes". In: *The American Mathematical Monthly* 61.10 (1954), pp. 675–682. ISSN: 00029890, 19300972. URL: http://www.jstor.org/stable/2307321 (visited on 04/04/2024).

[3] Solomon W. Golomb. *Polyominoes*. 2nd. Princeton, NJ: Princeton University Press, 1994.

Καλή Επιτυχία!