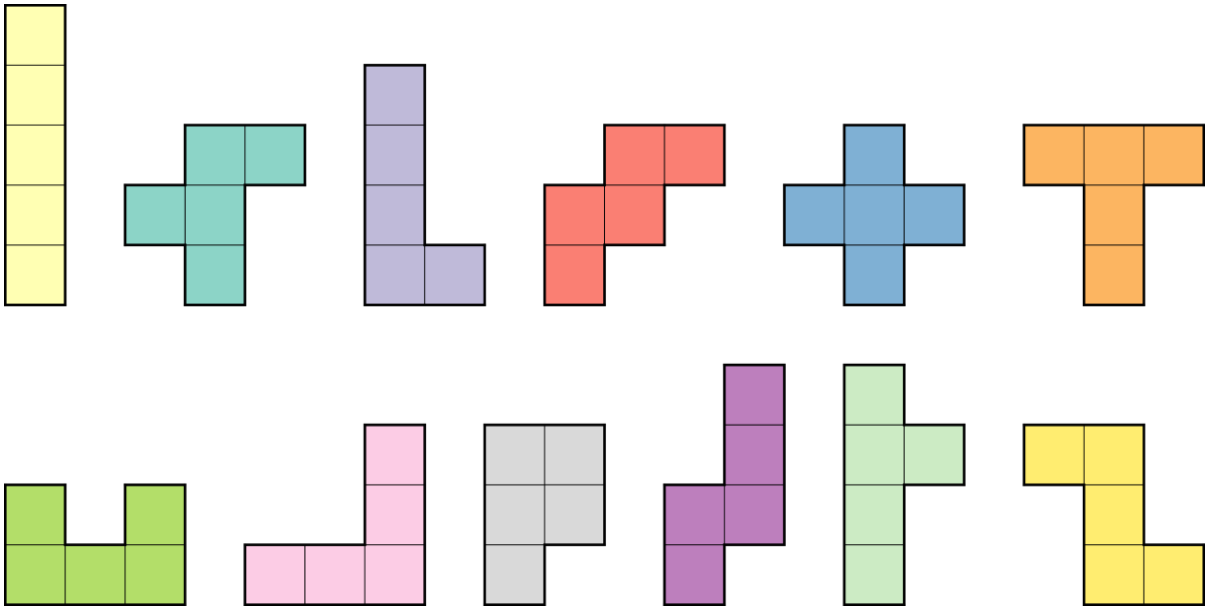
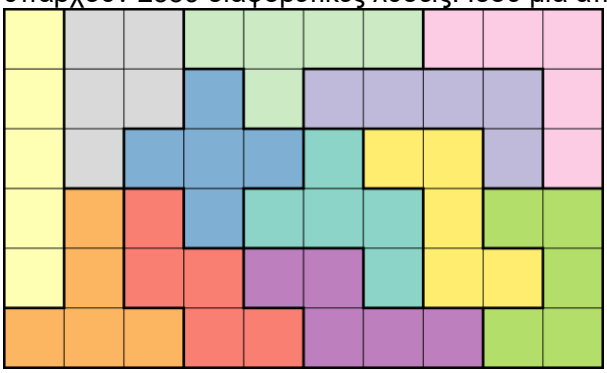
Πολυόμινο

Στο βιβλίο επιστημονικής φαντασίας του Arthur C. Clarke «Η Αυτοκρατορία της Γης» (Imperial Earth) στον πρωταγωνιστή τίθενται δύο προβλημάτα με *πεντόμινο* (pentomino). Ένα πεντόμινο είναι ένα πολύγωνο που απαρτίζεται από πέντε ίσα τετράγωνα όταν τοποθετηθούν έτσι ώστε η πλευρά κάθε τετραγώνου να εφάπτεται τουλάχιστον με την πλευρά ενός άλλου τετραγώνου. Υπάρχουν 12 διαφορετικά πεντόμινο αν δεν λάβουμε υπόψη περιστροφές και συμμετρίες:



Τα δύο προβλήματα είναι:

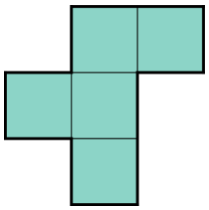
- Μπορούμε να γεμίσουμε ένα ορθογώνιο παραλληλόγραμμο διαστάσεων 6×106×10 χρησιμοποιώντας τα 12 αυτά πεντόμινο; Η απάντηση είναι ναι, και υπάρχουν 2339 διαφορετικές λύσεις. Ιδού μία από αυτές:



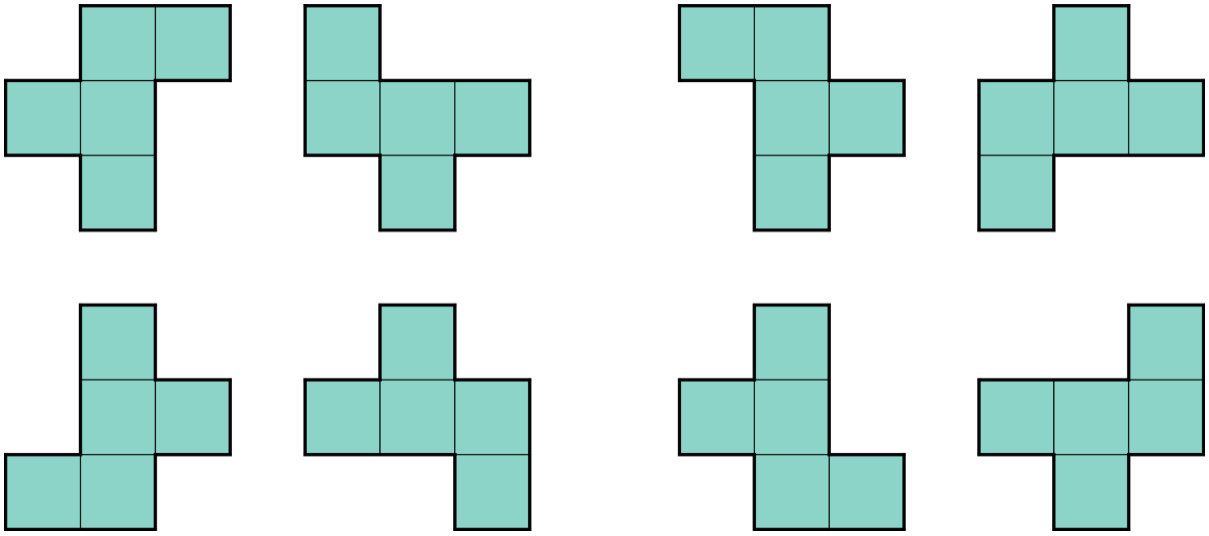
- Μπορούμε να γεμίσουμε ένα ορθογώνιο παραλληλόγραμμο διαστάσεων 3×203×20 χρησιμοποιώντας τα 12 αυτά πεντόμινο; Η απάντηση είναι ναι, αλλά υπάρχουν μόνο δύο λύσεις. Ιδού η μία από αυτές:

Όπως μπορείτε να δείτε, τα πεντόμινο στα παραπάνω σχήματα μπορούμε να τα περιστρέψουμε ή να τα αναποδογυρίσουμε, οριζόντια ή κάθετα. Όμως, πόσα δυνατά σχήματα προκύπτουν τότε;

Αν επιτρέψουμε στα πεντόμινο να περιστρέφονται ή να τα αναποδογυρίζουμε οριζόντια ή κάθετα, τότε δεν έχουμε μόνο 12. Για παράδειγμα, έστω το πεντόμινο:

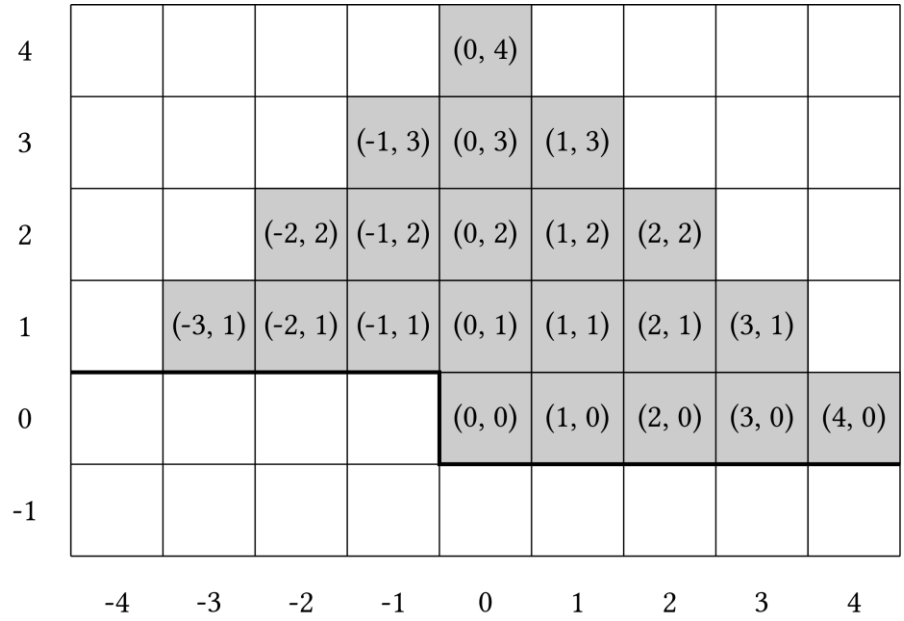


Αν το περιστρέψουμε ή το αναποδογυρίσουμε, μπορούμε να πάρουμε τα παρακάτω 8 σχήματα:



Συνολικά υπάρχουν 63 διαφορετικά πεντόμινο αν επιτρέψουμε περιστροφές και συμμετρίες. Όταν δεν επιτρέπουμε περιστροφές και συμμετρίες έχουμε *ελεύθερα* (free) πεντόμινο, τα οποία είναι τα 12 που είδαμε στην αρχή. Διαφορετικά, έχουμε τα 63 *σταθερά* (fixed) πεντόμινο.

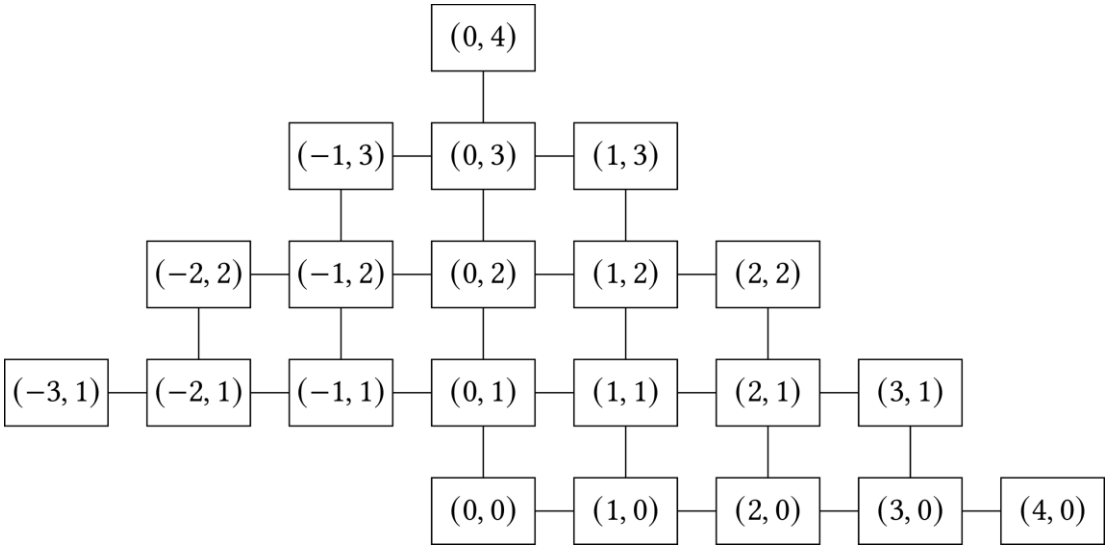
Μπορούμε τώρα να γενικεύσουμε και να δουλεύουμε με διαφορετικό αριθμό τετραγώνων αντί για πέντε. Ένα *πολυόμινο* (polyomino) είναι ένα γεωμετρικό σχήμα που αποτελείται από έναν αριθμό τετραγώνων κάθε δύο από τα οποία έχουν μια κοινή πλευρά. Όπως και τα πεντόμινο, τα πολυόμινο μπορεί να είναι ελεύθερα ή σταθερά. Εμάς μας ενδιαφέρει να βρούμε έναν τρόπο που θα μας μετράει τον αριθμό των σταθερών πολυόμινο, για κάθε δυνατό πλήθος τετραγώνων nn .
Για να το κάνουμε αυτό μπορούμε να εργαστούμε ως εξής. Θα χρησιμοποιήσουμε πεντόμινο ως παράδειγμα. Ορίζουμε ένα τετραγωνικό δικτύωμα (square lattice) στο επίπεδο, πάνω στο οποίο θα μπορούν να τοποθετηθούν τα διαφορετικά πεντόμινο. Θεωρούμε ότι όλα τα πεντόμινο περιέχουν το τετράγωνο στο σημείο $(0,0)$(0,0) . Τότε τα πεντόμινο μπορούν να εκταθούν στα γραμμοσκιασμένα τετράγωνα του παρακάτω σχήματος:



Η έντονη γραμμή δείχνει τα τετράγωνα για των οποίων τις συντεταγμένες έχουμε:

$$\{(x,y) \mid (y>0) \text{ or } (y=0) \text{ and } x\geq 0\}\{(x,y) \mid (y>0) \text{ or } (y=0) \text{ and } x\geq 0\}$$

Τα δυνατά πεντόμινο τότε προκύπτουν από πέντε τετράγωνα κάθε δύο από τα οποία έχουν κοινή πλευρά και περιέχουν το τετράγωνο στη θέση $(0,0)(0,0)$.
Ας ορίσουμε τώρα έναν γράφο όπου κάθε κόμβος είναι ένα από τα γραμμοσκιασμένα τετράγωνα και οι γείτονές του είναι τα γειτονικά γραμμοσκιασμένα τετράγωνα:



Τότε, τα πεντόμινο είναι οι συνδεμένοι υπογράφοι του γράφου που έχουν πέντε κόμβους.

Συμπεραίνουμε λοιπόν ότι για να βρούμε πόσα πεντόμινο υπάρχουν, θα πρέπει να βρούμε πόσοι συνδεμένοι υπογράφου του γράφου έχουν πέντε κόμβους. Γενικότερα, αν έχουμε πολυόμινο μεγέθους nn , η λογική παραμένει η ίδια. Κατασκευάζουμε τον αντίστοιχο γράφο και καταμετράμε τον αριθμό των συνδεμένων υπογράφων με nn κόμβους. Για να βρούμε το σύνολο αυτών των υπογράφων μπορούμε να χρησιμοποιήσουμε τον παρακάτω αλγόριθμο:

Algorithm 1: Count Fixed Polyominoes

```
CountFixedPolyominoes(G, untried, n, p, c) → r
    Input: G = (V, E), a graph
           untried, a set of nodes
           n, the size of polyominos
           p, the current polyomino
           c, a counter
    Output: r, the number of polyominoes of size n
1   while not IsSetEmpty(untried) do
2       u ← RemoveFromSet(untried)
3       AppendToList(p, u)
4       if SizeList(p) = n then
5           AddOne(c)
6       else
7           new_neighbors ← CreateSet()
8           foreach v in AdjacencyList(G, u) do
9               if v ∉ untried and v ∉ p and v ∉ Neighbors(p \ u) then
10                  AddToSet(new_neighbors, v)
11          new_untried ← untried + new_neighbors
12          CountFixedPolyominoes(G, new_untried, n, p, c)
13      RemoveFromList(p, u)
14  return ValueOf(c)
```

Ο αλγόριθμος είναι αναδρομικός. Ως παραμέτρους παίρνει τον γράφο που κατασκευάσαμε (*G*G), ένα σύνολο από τετράγωνα που πρέπει να δοκιμάσουμε (*untried*untried), το μέγεθος των πολυόμινο (*n*n), το τρέχον πολυόμινο που κατασκευάζουμε (*p*p) και έναν αριθμητή (*c*c). Τον αλγόριθμο τον καλούμε δίνοντας ως *untried*untried το σύνολο {(0,0)}{(0,0)} , ως τρέχον πολυόμινο μια κενή λίστα, και έναν μετρητή αρχικοποιημένο στο μηδέν.

Ο αλγόριθμος εκτελείται όσο το σύνολο *untried*untried δεν είναι άδειο. Τότε, αφαιρούμε ένα στοιχείο από το σύνολο, δηλαδή έναν κόμβο τον οποίο αποκαλούμε *u*u (γραμμή 2), και τον προσθέτουμε στο τρέχον πολυόμινο (γραμμή 3). Αν το τρέχον πολυόμινο έχει μέγεθος ίσο με *n*n , αυξάνουμε τον μετρητή κατά ένα (γραμμές 4-5). Αλλιώς, στις γραμμές 7-12 βρίσκουμε τους γείτονες του κόμβου *u*u για τους οποίους ισχύουν τα εξής:

1. Δεν ανήκουν στο σύνολο *untried*untried .
2. Δεν ανήκουν στο τρέχον πολυόμινο *p*p που κατασκευάζουμε.
3. Δεν είναι γείτονες των υπόλοιπων κόμβων, εκτός του *u*u , που ανήκουν στο *p*p : με το *p**u*p\u συμβολίζουμε τους κόμβους του *p*p εκτός του *u*u . Εδώ μπορεί να αναρωτηθείτε «Μα καλά, το *untried*untried δεν περιέχει τους γείτονες του υπό κατασκευή πολυόμινο; Γιατί χρειάζεται αυτός ο έλεγχος αφού ήδη ελέγχουμε ότι *v*∉*untried*v∉untried ;». Η απάντηση είναι όχι, δεν περιέχει αναγκαστικά όλους τους γείτονες του υπό κατασκευή πολυόμινο γιατί κάθε φορά που περνάμε από τη γραμμή 2 του αλγορίθμου αφαιρούμε από το *untried*untried ένα στοιχείο, οπότε ένας γείτονας μπορεί να είναι ένα από τα στοιχεία που έχουν αφαιρεθεί.

Αφού τους βρούμε, δημιουργούμε ένα νέο σύνολο με τετράγωνα που πρέπει να δοκιμάσουμε (γραμμή 11) και καλούμε αναδρομικά τον αλγόριθμο. Πριν την επόμενη επανάληψη του βρόχου των γραμμών 1-13 αφαιρούμε τον κόμβο *u*u από το *p*p .

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 αποδωθεί. 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-2020-1`.
- Μέσα στον παραπάνω κατάλογο το πρόγραμμα θα πρέπει να αποθηκευτεί με το όνομα `count_fixed_polyominoes.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 `argparse` ή της βιβλιοθήκης `sys`(συγκεκριμένα, της λίστας `sys.argv`) προκειμένου να διαβάσει το πρόγραμμα το μέγεθος των πολυόμινο και το διακόπτη που θα ορίζει αν θα εκτυπωθεί ο γράφος που κατασκευάζουμε (βλ. παρακάτω).
- Επιτρέπεται η χρήση της βιβλιοθήκης `pprint` για την εκτύπωση του γράφου που κατασκευάζουμε.
- Το πρόγραμμα θα πρέπει να είναι γραμμένο σε Python 3.
- Το πρόγραμμα θα το καλεί ως χρήστης ως εξής (όπου `python` η κατάλληλη εντολή στο εκάστοτε σύστημα):

```
python count_fixed_polyominoes.py [-p] <n>
```

Στην εντολή αυτή, η παράμετρος `<n>` είναι το μέγεθος των πολυόμινο. Αν ο χρήστης δώσει την παράμετρο `-p` πριν από αυτά θα εκτυπώνεται στην οθόνη ο γράφος που κατασκευάζουμε. Προσοχή, στην περίπτωση αυτή οι κόμβοι στις λίστες γειτνίασης θα πρέπει να εμφανίζονται με τη σειρά που προκύπτει αντίθετα από τη φορά των δεικτών του ρολογιού· δείτε παρακάτω στα παραδείγματα.

Η έξοδος του προγράμματος θα είναι ο γράφος που κατασκευάζουμε για την επίλυση του προβλήματος και ο αριθμός των ελεύθερων πολυόμινο.

Παραδείγματα Εκτέλεσης

Παράδειγμα 1

Αν ο χρήστης καλέσει το πρόγραμμα με τον ακόλουθο τρόπο:

```
python count_fixed_polyominoes.py -p 3
```

η έξοδος του προγράμματος θα πρέπει να είναι *ακριβώς* η παρακάτω:

```
{(-1, 1): [(0, 1)],
 (0, 0): [(1, 0), (0, 1)],
 (0, 1): [(1, 1), (0, 2), (-1, 1), (0, 0)],
 (0, 2): [(0, 1)],
 (1, 0): [(2, 0), (1, 1), (0, 0)],
 (1, 1): [(0, 1), (1, 0)],
 (2, 0): [(1, 0)]}
6
```

Παράδειγμα 2

Αν ο χρήστης καλέσει το πρόγραμμα με τον ακόλουθο τρόπο:

```
python count_fixed_polyominoes.py -p 4
```

η έξοδος του προγράμματος θα πρέπει να είναι ακριβώς η παρακάτω:

```
{(-2, 1): [(-1, 1)],
 (-1, 1): [(0, 1), (-1, 2), (-2, 1)],
 (-1, 2): [(0, 2), (-1, 1)],
 (0, 0): [(1, 0), (0, 1)],
 (0, 1): [(1, 1), (0, 2), (-1, 1), (0, 0)],
 (0, 2): [(1, 2), (0, 3), (-1, 2), (0, 1)],
 (0, 3): [(0, 2)],
 (1, 0): [(2, 0), (1, 1), (0, 0)],
 (1, 1): [(2, 1), (1, 2), (0, 1), (1, 0)],
 (1, 2): [(0, 2), (1, 1)],
 (2, 0): [(3, 0), (2, 1), (1, 0)],
 (2, 1): [(1, 1), (2, 0)],
 (3, 0): [(2, 0)]}
19
```

Παράδειγμα 3

Αν ο χρήστης καλέσει το πρόγραμμα με τον ακόλουθο τρόπο:

```
python count_fixed_polyominoes.py -p 5
```

η έξοδος του προγράμματος θα πρέπει να είναι ακριβώς η παρακάτω:

```
{(-3, 1): [(-2, 1)],
 (-2, 1): [(-1, 1), (-2, 2), (-3, 1)],
 (-2, 2): [(-1, 2), (-2, 1)],
 (-1, 1): [(0, 1), (-1, 2), (-2, 1)],
 (-1, 2): [(0, 2), (-1, 3), (-2, 2), (-1, 1)],
 (-1, 3): [(0, 3), (-1, 2)],
 (0, 0): [(1, 0), (0, 1)],
 (0, 1): [(1, 1), (0, 2), (-1, 1), (0, 0)],
 (0, 2): [(1, 2), (0, 3), (-1, 2), (0, 1)],
 (0, 3): [(1, 3), (0, 4), (-1, 3), (0, 2)],
 (0, 4): [(0, 3)],
 (1, 0): [(2, 0), (1, 1), (0, 0)],
 (1, 1): [(2, 1), (1, 2), (0, 1), (1, 0)],
 (1, 2): [(2, 2), (1, 3), (0, 2), (1, 1)],
 (1, 3): [(0, 3), (1, 2)],
 (2, 0): [(3, 0), (2, 1), (1, 0)],
 (2, 1): [(3, 1), (2, 2), (1, 1), (2, 0)],
 (2, 2): [(1, 2), (2, 1)],
 (3, 0): [(4, 0), (3, 1), (2, 0)],
 (3, 1): [(2, 1), (3, 0)],
 (4, 0): [(3, 0)]}
63
```

Παράδειγμα 4

Αν ο χρήστης καλέσει το πρόγραμμα με τον ακόλουθο τρόπο:

```
python count_fixed_polyominoes.py 10
```

η έξοδος του προγράμματος θα πρέπει να είναι ακριβώς η παρακάτω:

```
36446
```

Παράδειγμα 5

Αν ο χρήστης καλέσει το πρόγραμμα με τον ακόλουθο τρόπο:

```
python count_fixed_polyominoes.py 15
```

η έξοδος του προγράμματος θα πρέπει να είναι ακριβώς η παρακάτω:

```
27394666
```

Επιπλέον Παραδείγματα

Για τα διαφορετικά μεγέθη πολυόμινο μέχρι και το 15, το πλήθος τους είναι όπως στον παρακάτω πίνακα:

Μέγεθος Πολυόμινο	Πλήθος
1	1
2	2
3	6
4	19
5	63
6	216
7	760
8	2725
9	9910
10	36446
11	135268
12	505861
13	1903890
14	7204874
15	27394666

Καλή Επιτυχία!

Για Περισσότερες Πληροφορίες

Ο Arthur C. Clarke έμαθε τα πολυόμινο από τον σκηνοθέτη Stanley Kubrick στα γυρίσματα της ταινίας 2001: Η Οδύσσεια του Διαστήματος (2001: A Space Odyssey)) και τα συνύφανε στο βιβλίο του:

Arthur C. Clarke, 1975. Imperial Earth, Gollancz, London.

Ο αλγόριθμος που πρέπει να υλοποιήσετε στην εργασία είναι ουσιαστικά ο αλγόριθμος του D. Hugh Redelmeier:

D. Hugh Redelmeier, 1981. Counting Polyominoes: Yet Another Attack, Discrete Mathematics 36, 191-203.

Ο αλγόριθμος αυτός δεν είναι γρήγορος, καθώς η πολυπλοκότητά του είναι εκθετική. Ο πιο γρήγορος αλγόριθμος που γνωρίζουμε, ο οποίος όμως εξακολουθεί να έχει εκθετική πολυπλοκότητα, είναι ο αλγόριθμος του Iwan Jensen:

Iwan Jensen, 2001. Enumerations of Lattice Animals and Trees, Journal of Statistical Physics, 102, 865-881.

Το πρόβλημα μπορεί να επεκταθεί σε τρεις και παραπάνω διαστάσεις. Για λεπτομέρειες, δείτε:

Gadi Aleksandrowicz and Gill Barequet, 2006. Counting *d*d -Dimensional Polycubes and Nonrectangular Planar Polyominoes. In D.Z. Chen and D.T. Lee (Eds.): COCOON 2006, LNCS 4112, pp. 418–427, 2006.
Το όνομα πολυόμινο εισήγαγε ο Solomon W. Golomb το 1954. Περισσότερες πληροφορίες για αυτά μπορείτε να δείτε στο βιβλίο του:

Solomon W. Golomb, (1994). Polyominoes (2nd ed.). Princeton, New Jersey: Princeton University Press.

Στο ευρύ κοινό τα πολυόμινο παρουσίασε ο Martin Gardner το 1957 και το 1960 στη στήλη του στο Scientific American:

Martin Gardner, 1957. More about complex dominoes. Scientific American, 197(6), 126–140.

Martin Gardner, 1960. More about the shapes that can be made with complex dominoes. Scientific American. 203 (5), 186–201.