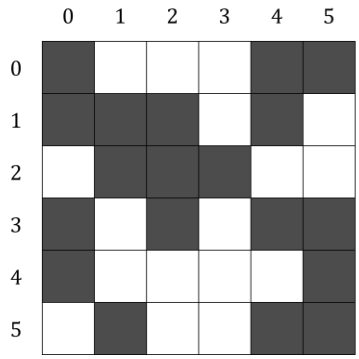
Πανεπιστήμιο Ιωαννίνων – Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Δομές Δεδομένων [ΜΥΥ303] – Χειμερινό Εξάμηνο 2022

1η Εργαστηριακή Άσκηση
Δομές Εύρεσης-Ένωσης
Παράδοση έως Τετάρτη 26/10, 14:00 από το eCourse

ΠΡΟΣΟΧΗ: Γράψτε σε κάθε αρχείο που παραδίδετε τα ονόματα και τους Α.Μ. των μελών της ομάδας σας. Συμπεριλάβετε όλα τα αρχεία σας (κώδικας Java και lab1results.txt) σε ένα zip αρχείο το οποίο να έχει το όνομα ενός μέλους της ομάδας σας με λατινικούς χαρακτήρες.

Ένα επιτραπέζιο παιχνίδι παίζεται σε μία σκακιέρα διαστάσεων $n \times n$, όπου κάθε τετράγωνο έχει άσπρο ή μαύρο χρώμα. Δύο τετράγωνα (i, j) και (k, l) είναι ***γειτονικά*** αν έχουν μια κοινή οριζόντια ή κάθετη ακμή. Δηλαδή, το τετράγωνο (i, j) έχει το πολύ 4 γειτονικά τετράγωνα $(k, l) \in \{(i-1, j), (i, j-1), (i+1, j), (i, j+1)\}$ (για όσα ισχύει $0 \leq k \leq n-1$ και $0 \leq l \leq n-1$). Ένα ***εφικτό μονοπάτι*** στη σκακιέρα αποτελείται από μια ακολουθία τετραγώνων $(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)$, όπου κάθε τετράγωνο (i_a, j_a), $1 \leq a \leq k$, είναι μαύρο και το επόμενο τετράγωνο (i_{a+1}, j_{a+1}) είναι γειτονικό του (i_a, j_a). Για παράδειγμα, στο παρακάτω σχήμα, η ακολουθία $(0,0), (1,0), (1,1), (1,2), (2,2), (2,1)$ αποτελεί ένα εφικτό μονοπάτι.



$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Αρχικά, όλα τα τετράγωνα είναι άσπρα και στην πορεία του παιχνιδιού κάποια γίνονται μαύρα. Ένας παίκτης μπορεί να κινείται μόνο πάνω στα μαύρα τετράγωνα, χρησιμοποιώντας εφικτά μονοπάτια. Μια ***συστάδα*** C είναι ένα μεγιστοτικό σύνολο μαύρων τετραγώνων, τέτοιο ώστε οποιαδήποτε δύο τετράγωνα του C να συνδέονται με κάποιο εφικτό μονοπάτι. Στο παραπάνω σχήμα, έχουμε 5 συστάδες, τις $\{(0,0), (1,0), (1,1), (1,2), (2,1), (2,2), (2,3), (3,2)\}$, $\{(0,4), (0,5), (1,4)\}$, $\{(3,0), (4,0)\}$, $\{(5,1)\}$ και $\{(3,4), (3,5), (4,5), (5,4), (5,5)\}$.

Δώστε μια αποδοτική υλοποίηση του παρακάτω παιχνιδιού χρησιμοποιώντας κατάλληλες δομές δεδομένων για τη διατήρηση των τετραγώνων που συνδέονται με εφικτά μονοπάτια. Συγκεκριμένα, θα πρέπει να υλοποιήσετε τις ακόλουθες μεθόδους:

void colorBlack(int i, int j)	Χρωματίζει μαύρο το τετράγωνο (i, j). Αν το (i, j) έχει γειτονικά μαύρα τετράγωνα, τότε υπολογίζει τις νέες συστάδες της σκακιέρας.
boolean test(int i, int j, int k, int l)	Επιστρέφει true αν τα τετράγωνα (i, j) και (k, l) συνδέονται με εφικτό μονοπάτι (δηλαδή ανήκουν στην ίδια συστάδα). Διαφορετικά, επιστρέφει false.

int clusters()	Επιστρέφει το πλήθος των συστάδων της σκακιέρας.

Θα πρέπει να συμπληρώσετε τις παραπάνω μεθόδους στο αρχείο BoardGame.java. Αναπαριστούμε τη σκακιέρα ως ένα διδιάστατο $n \times n$ πίνακα ακεραίων B, όπου $B[i,j] = 0$ αν το τετράγωνο (i,j) είναι άσπρο και $B[i,j] = 1$ αν το τετράγωνο (i,j) είναι μαύρο, όπως φαίνεται στο παραπάνω σχήμα.

Για την αποδοτική υλοποίηση των μεθόδων, μπορείτε να χρησιμοποιήσετε μια δομή εύρεσης-ένωσης, με στοιχεία τα μαύρα τετράγωνα της σκακιέρας. Μπορείτε να δώσετε στο τετράγωνο (i,j) την ακέραιη ταυτότητα $i \cdot n + j$. Επίσης, παρατηρείστε ότι το τετράγωνο με ακέραιη ταυτότητα v είναι το $(v \text{ div } n, v \bmod n)$, όπου $v \text{ div } n$ το πηλίκο και $v \bmod n$ το υπόλοιπο της ακέραιης διαίρεσης του v με το n.

Δομή Γρήγορης Ένωσης με Σταθμισμένη Ένωση και Συμπίεση Διαδρομής

Συμπληρώσετε στο αρχείο UnionFind.java την υλοποίηση μιας αποδοτικής δομής εύρεσης-ένωσης για το πολύ N στοιχεία, τα οποία είναι οι ακέραιοι από το σύνολο $\{0,1,\dots,N-1\}$. Η δομή σας θα πρέπει να χρησιμοποιεί σταθμισμένη γρήγορη ένωση και συμπίεση διαδρομής, και να υποστηρίζει τις ακόλουθες μεθόδους:

UnionFind(int N)	Μέθοδος κατασκευής (constructor) της δομής εύρεσης-ένωσης. Αρχικοποιεί τη δομή για το πολύ N στοιχεία, τα οποία είναι οι ακέραιοι από το $\{0,1,2,\dots,N-1\}$. Αρχικά, πριν δημιουργηθεί οποιοδήποτε σύνολο, έχουμε $parent[v] = -1$ και $size[v] = 0$ για κάθε $0 \le v < N$.
void makeSet(int v)	Δημιουργεί ένα μονοσύνολο $\{v\}$ για το στοιχείο v, με αντιπρόσωπο τον εαυτό του. Δηλαδή, θέτει $parent[v] = v$ και $size[v] = 1$.
int find(int v)	Επιστρέφει τον αντιπρόσωπο (ρίζα) του συνόλου που περιέχει το αντικείμενο v.
void unite(int v, int u)	Αν τα αντικείμενα v και u βρίσκονται σε διαφορετικά σύνολα τότε ενώνει τα δύο αυτά σύνολα.
int setCount()	Επιστρέφει το πλήθος των συνόλων στη δομή.

Οι μέθοδοι UnionFind() και makeSet() σας δίνονται έτοιμες, και επομένως θα πρέπει να υλοποιήσετε κατάλληλα τις υπόλοιπες μεθόδους.

Για τη σταθμισμένη ένωση πρέπει να λαμβάνετε υπόψη το μέγεθος κάθε συνόλου, Για το σκοπό αυτό, χρησιμοποιούμε τον πίνακα size[] ο οποίος αποθηκεύει το πλήθος των αντικειμένων ενός συνόλου ως εξής. Αν το στοιχείο v είναι ο αντιπρόσωπος του συνόλου του (δηλαδή ισχύει parent[v] == v) τότε το σύνολο περιέχει size[v] στοιχεία. Αν ενώσουμε τα σύνολα με αντιπροσώπους p και q με size[p] ≥ size[q] τότε θα έχουμε size[p] = size[p] + size[q].

Το αποτέλεσμα της find(v) με συμπίεση διαδρομής είναι η αλλαγή του γονέα parent[u] για όλα τα αντικείμενα u που βρίσκονται στο μονοπάτι από το v έως τη ρίζα του δένδρου (τον αντιπρόσωπο του συνόλου) που περιέχει το v. Ο νέος γονέας αυτών των κόμβων είναι ο αντιπρόσωπος του συνόλου.

Η μέθοδος main() στο αρχείο UnionFind.java αρχικοποιεί μια δομή εύρεσης-ένωσης για $N = 16$. Στη δημιουργεί μονοσύνολα για τα στοιχεία με άρτιο αριθμό, και τα ενώνει ανά δύο μέχρι να σχηματιστεί ένα σύνολο {0,2,4,6,8,10,12,14}. Βεβαιωθείτε ότι η υλοποίησή σας δουλεύει σωστά.

Εκτέλεση Προγραμμάτων

Η μέθοδος main() στο αρχείο BoardGame.java κατασκευάζει μια σκακιέρα $n \times n$ διαστάσεων, όπου αρχικά μόνο τα 4 γωνιακά τετράγωνα $(0,0)$, $(0, n-1)$, $(n-1,0)$ και $(n-1, n-1)$ είναι μαύρα. Στη συνέχεια εκτελεί $n^2 + n^4/2$ επαναλήψεις, όπου σε κάθε επανάληψη επιλέγει ένα τυχαίο τετράγωνο που θα χρωματιστεί μαύρο (αν δεν είναι ήδη μαύρο). Μόλις τερματίσει αυτή η διαδικασία, ελέγχει αν το τετράγωνο $(0,0)$ συνδέεται με εφικτό μονοπάτι με κάποιο από τα υπόλοιπα τρία γωνιακά τετράγωνο. Τέλος, τυπώνει το πλήθος των μαύρων τετραγώνων και των συστάδων.

Εκτελέστε το πρόγραμμα BoardGame.java (αφού έχετε υλοποιήσει και το UnionFind.java) για $n = 10$, 20, 100, 1000 και 2000, και αποθηκεύστε τα αποτελέσματα της εκτέλεσης του στο αρχείο lab1results.txt. Για να εκτελέσετε το BoardGame.java για $n = 10$, δώστε στη γραμμή εντολών

java BoardGame 10

Ομοίως και για τις άλλες τιμές του n. Αποθηκεύστε τα αποτελέσματα της εκτέλεσης τους στο αρχείο lab1results.txt.

Παραδοτέα

Ανεβάστε στο eCourse ένα zip αρχείο με τα τελικά σας προγράμματα BoardGame.java και UnionFind.java, καθώς και με το αρχείο των αποτελεσμάτων lab1results.txt. Το zip αρχείο πρέπει να έχει το όνομα ενός μέλους της ομάδας σας με λατινικούς χαρακτήρες.