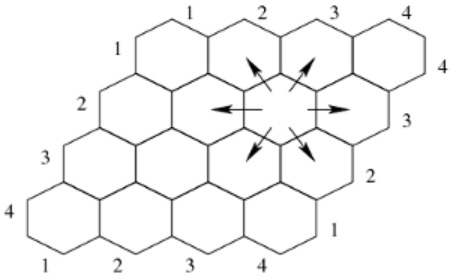
Find-Union i Grafy

1. (*)[XII OI, Skarbonki] Mamy n skarbonek otwieranych kluczykami. Do każdej skarbonki jest inny kluczyk. Kluczyki do skarbonek są powrzucane do skarbonek -- dowolnie. Żeby dostać się do zawartości skarbonki, skarbonkę można otworzyć kluczykiem (jeśli się go ma) lub ją rozbić.
   Skarbonki są ponumerowane od 0 do $n-1$. Na podstawie tablicy a (rozmiaru n), takiej, że $a[i] =$ numer skarbonki, w której znajduje się kluczyk do skarbonki numer i, oblicz minimalną liczbę skarbonek, które należy rozbić, tak aby dostać się do zawartości wszystkich skarbonek.
2. (*)[X OI, Małpki] Na drzewie wisi n małpek ponumerowanych od 1 do n. Małpka z nr 1 trzyma się gałęzi ogonkiem. Pozostałe małpki albo są trzymane przez inne małpki, albo trzymają się innych małpek, albo jedno i drugie równocześnie. Każda małpka ma dwie przednie łapki, każdą może trzymać co najwyżej jedną inną małpkę (za ogon). Rozpoczynając od chwili 0, co sekundę jedna z małpek puszcza jedną łapkę. W ten sposób niektóre małpki spadają na ziemię, gdzie dalej mogą puszczać łapki (czas spadania małpek jest pomijalnie mały).
   Zaprojektuj algorytm, który na podstawie opisu tego która małpka trzyma którą, oraz na podstawie opisu łapek puszczanych w kolejnych chwilach, dla każdej małpki wyznaczy moment, kiedy spadnie ona na ziemię.
3. (*) Na szachownicy jest ustawionych n wież, które należy pokolorować. Jeśli dwie wieże się atakują (są w tej samej kolumnie lub wierszu), to muszą być tego samego koloru. Napisz procedurę `int kolory(const vector<pair<int, int>> &towers)`, która na podstawie list współrzędnych wież wyznaczy maksymalną liczbę kolorów, których można użyć kolorując wieże. Podaj złożoność czasową i pamięciową swojego rozwiązania.
4. Dana jest plansza podzielona na sześciokątne pola, o wymiarach $n \times n$ (patrz rysunek). i-te pole w j-tym rzędzie sąsiaduje z polami:
   - $i-1$ i i w rzędzie $j-1$,
   - $i-1$ i $i+1$ w rzędzie j,
   - i i $i+1$ w rzędzie $j+1$.

   Dwaj gracze, biały i czarny, grają w grę *hex*. Zajmują oni na przemian wolne pola na planszy. Zaczyna gracz biały. Gracz biały stara się połączyć swoimi polami lewy i prawy brzeg planszy, a gracz czarny górny i dolny brzeg planszy. Wygrywa gracz, który jako pierwszy połączy odpowiednie brzegi planszy, w ten sposób, że z jednego brzegu na drugi będzie można przejść przez sąsiadujące pola zajęte przez danego gracza. Napisz procedurę `int hex(int n, const vector<pair<int, int>> &moves)`, która na podstawie wielkości planszy n, oraz list współrzędnych (rząd, pozycja w rzędzie) kolejno zajmowanych przez graczy pól określi, czy wygrał gracz biały (wynik dodatni), czy czarny (wynik ujemny) i w którym ruchu (wartość bezwzględna z wyniku). Jeżeli gra nie została rozstrzygnięta, to oczekiwanym wynikiem jest zero.
5. Dany jest graf nieskierowany, którego wierzchołki są ponumerowane od 0 do $n-1$. Napisz procedurę `int path(const Graph &g)`, która wyznacza długość (liczoną liczbą krawędzi) najdłuższej ścieżki w tym grafie, na której numery wierzchołków tworzą ciąg rosnący.
6. [II OI, zadanie *Klub Prawoskrętnych Kierowców*, PCh] Mamy dany układ ulic, leżących na prostokątnej siatce $m \times n$. Ulice są jedno- lub dwukierunkowe. Przyjmujemy, że cztery strony świata są reprezentowane za pomocą liczb całkowitych od 0 do 3: `enum directions {north, east, south, west};`
   Układ ulic jest dany w formie trójwymiarowej tablicy wartości logicznych `vector<vector<vector<bool>>>ulice`: $\texttt{ulice}[i][j][k]$ określa, czy z punktu o współrzednych (i, j) można przejechać w kierunku *k* jedną jednostkę odległości. Można założyć, że tablica ta uniemożliwia wyjechanie poza obszar $\{0, \ldots, m-1\} \times \{0, \ldots, n-1\}$.
   Członkowie Klubu Prawoskrętnych Kierowców na skrzyżowaniach jeżdżą prosto lub skręcają w prawo. Sprawdź, czy z punktu $(0, 0)$ prawoskrętny kierowca może dojechać do punktu $(m-1, n-1)$.

?

7. Dana jest prostokątna mapa wysokości górzystego terenu, w postaci prostokątnej tablicy dodatnich liczb całkowitych, o wymiarach $N \times M$. Chcemy przejść z pola o współrzędnych $(0, 0)$ na pole o współrzędnych $(N-1, M-1)$, ale nie chcemy wspinać się zbyt wysoko. (Możemy się przesuwać w kierunkach N, W, S, E.) Napisz procedurę `int wysokosc(const vector<vector<int>> &m)`, która dla danej mapy terenu określi minimalną największą wysokość, na którą musimy wejść w trakcie podróży.
8. W parku jest prostokątna sadzawka pokryta lodem. Jaś bawi się na lodzie. Niestety przyszła odwilż i lód zaczął pękać. Masz daną prostokątną tablicę zawierającą dodatnie liczby całkowite. Jest to mapa sadzawki, a liczby reprezentują moment, w którym lód w danym miejscu pęka. Napisz procedurę `int sadzawka(const vector<vector<int>> &mapa, int jas_x, int jas_y)`, która mając daną taką tablicę oraz współrzędne miejsca, w którym bawi się Jaś, wyznaczy moment, w którym Jaś straci możliwość wyjścia na brzeg (bo albo zostanie odcięty od brzegu spękanym lodem, albo wpadnie do wody).

   Możesz założyć, że:

   - Jaś może poruszać się po polach sąsiadujących ze sobą bokami,
   - Jaś porusza się nieporównanie szybciej, niż postępuje pękanie lodu,
   - liczby w danej tablicy są z zakresu od 1 do liczba elementów w tablicy.
9. (*) [XI OI, *Wyspy*] Dana jest prostokątna mapa $n \times m$ przedstawiająca archipelag na oceanie, reprezentowana jako `vector<vector<bool>>`. Elementy tablicy równe `true` reprezentują ląd, a `false` wodę. Na zewnątrz mapy znajduje się ocean. Wyspy na oceanie są wyspami rzędu 1. Na wyspach rzędu 1 mogą znajdować się jeziora rzędu 1, na nich mogą znajdować się wyspy rzędu 2 itd. Ogólnie:

   - ocean ma rząd 0,
   - wyspa na jeziorze (lub oceanie) rzędu k, ma rząd $k+1$,
   - jezioro na wyspie rzędu k ma rząd k.

   Pola wody przylegające do siebie bokami lub rogami należą do tego samego jeziora (lub oceanu). Pola lądu przylegające do siebie bokami należą do tej samej wyspy. Napisz procedurę `int wyspa(const vector<vector<bool>> &mapa)`, która dla danej mapy zwróci maksymalny rząd wyspy na mapie. Jeżeli na oceanie nie ma żadnej wyspy, to procedura powinna zwrócić 0. Przykład: 69.793N, 108.241W.

10. (*) [J.Paw.] Dany jest nieskierowany graf g oraz trzy wierchołki w tym grafie: u, v i w. Napisz procedurę `int polaczenie(const Graph &g, int u, int v, int w)`, która mając dane g, u, v i w wyznaczy minimalną liczbę krawędzi w grafie, jakie są konieczne, żeby wierzchołki u, v i w pozostały połączone. Jeżeli nie jest to możliwe, poprawnym wynikiem jest -1.
11. [XIV OI, zadanie *Biura*] Firma ma *n* pracowników. Każdy pracownik ma telefon komórkowy, a w nim telefony pewnej grupy innych pracowników. Przy tym, jeżeli pracownik *A* ma telefon pracownika *B*, to pracownik *B* ma numer pracownika *A*. Firma chce podzielić pracowników na grupy, w taki sposób żeby:
    - każdych dwóch pracowników albo było w tej samej grupie, albo miało nawzajem swoje numery telefonów,
    - liczba grup była maksymalna.

    Wyznacz podział pracowników na grupy.
12. Dany jest acykliczny graf skierowany (DAG). Jego wierzchołki są ponumerowane od 0 do $n-1$, a krawędzie są dane w postaci tablicy sąsiedztwa (kwadratowej tablicy `e` wartości logicznych; w grafie mamy krawędź $u \to v$ wtw., gdy `e[u][v]`). Powiemy, że wierzchołek u *dominuje* wierzchołek v, jeżeli dla każdego wierzchołka *w*, z którego można dojść do v, z u można dojść do w.
    Napisz procedurę `int zdominowani(const vector<vector<bool>> &g)`, która na podstawie tablicy opisującej graf obliczy liczbę wierzchołków, dla których istnieją wierzchołki je dominujące.

13. (*) [XIV OI, zadanie *Powódź*] Dana jest mapa topograficzna miasta położonego w kotlinie, w postaci prostokątnej tablicy typu `vector<vector<pair<int, bool>>>`. Liczby określają wysokość kwadratów jednostkowych, a wartości logiczne określają, czy dany kwadrat należy do terenu miasta. Przyjmujemy, że teren poza mapą jest położony wyżej niż jakikolwiek kwadrat na mapie.
    Miasto zostało całkowicie zalane. Żeby je osuszyć należy w kotlinie rozmieścić pewną liczbę pomp. Każda z pomp wypompowuję wodę aż do osuszenia kwadratu jednostkowego, na którym się znajduje. Osuszenie dowolnego kwadratu pociąga za sobą obniżenie poziomu wody lub osuszenie kwadratów jednostkowych, z których woda jest w stanie spłynąć do kwadratu, na którym znajduje się pompa. Woda może przepływać między kwadratami, które stykają się bokami.

    Wyznacz minimalną liczbę pomp koniecznych do osuszenia miasta. Teren nie należący do miasta nie musi być osuszony. Miasto nie musi tworzyć spójnego obszaru.

Last modified: Sunday, 18 December 2022, 1:00 PM