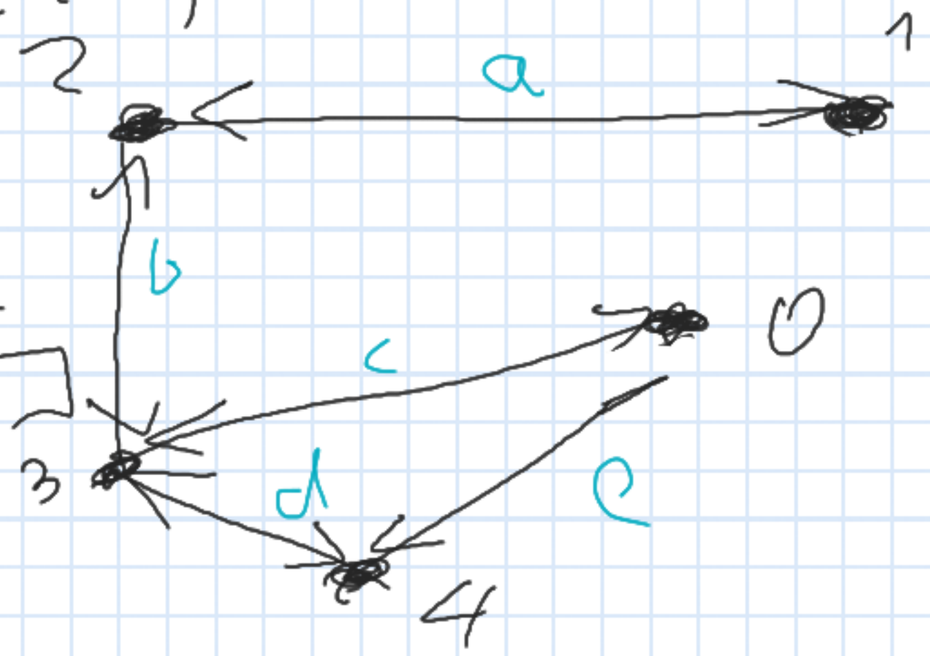
$\{(1,2),(2,1),(3,4),(4,3),\ldots\} = E$

$[0,1,2,3,4,5] = V$

$[[3,4],[2],[2,0,4],\ldots]$

$[\,]$



	0	1	2	3	4	5
0						-1
1						
2						
3						
4	1					
5						

1
1
1
1
1

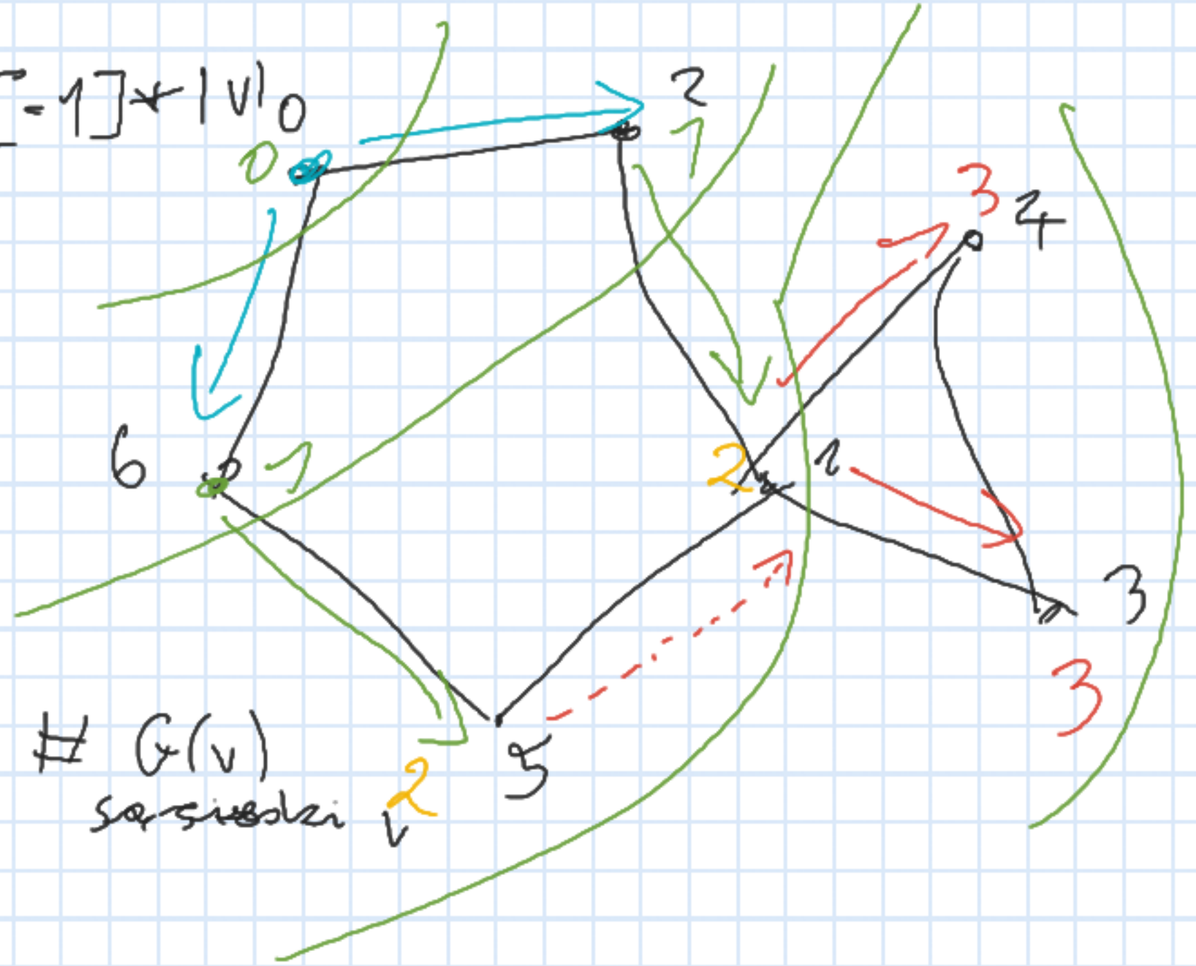
1	•
2	• •
3	
4	
5	

a b < d e

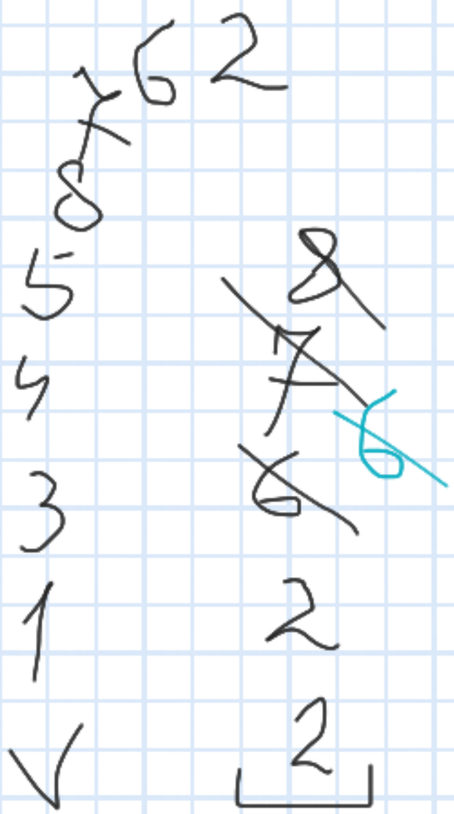
• 5

```
def BFS(s, G):
    visited = [False] * |V|
    q = Q()
    q.put((s, 0))
    while not q.empty():
        v, w = q.get()
        for u in G(v):
            if not visited[u]:
                q.put((u, w+1))
                visited[u] = w
```

```
visited = [-1] * |V|
```

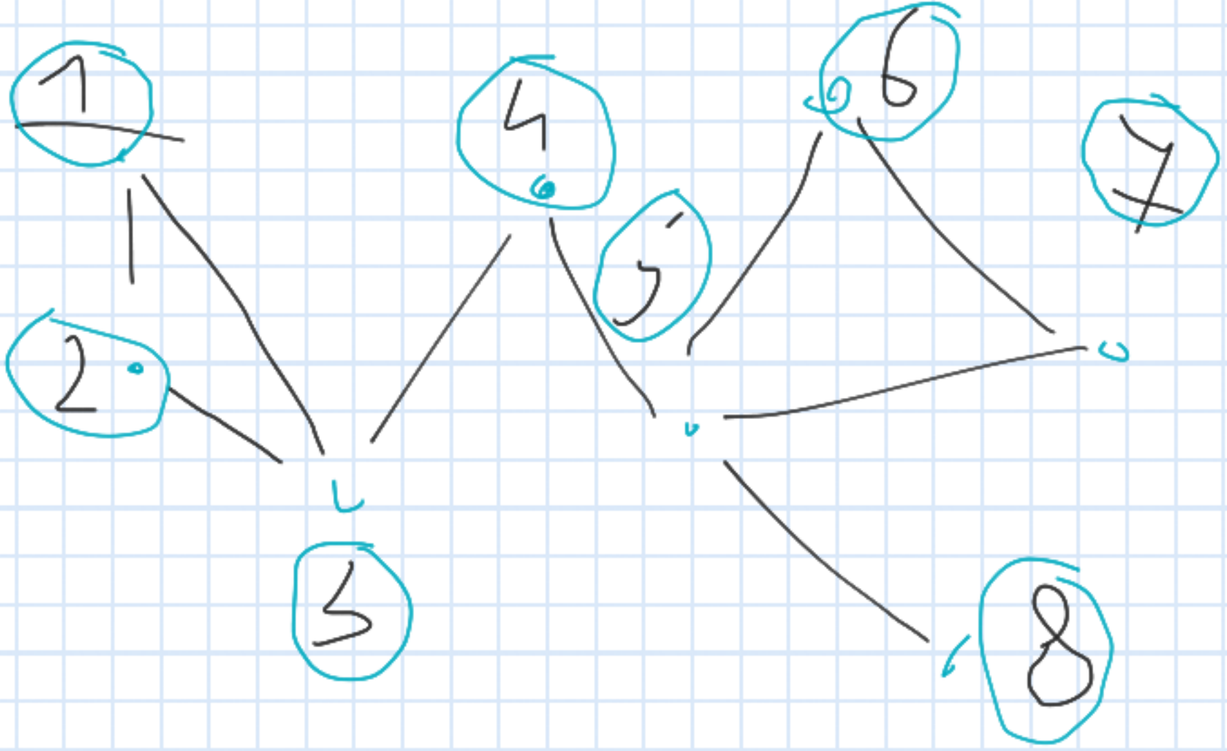


G(v) sąsiedzi



GLOB

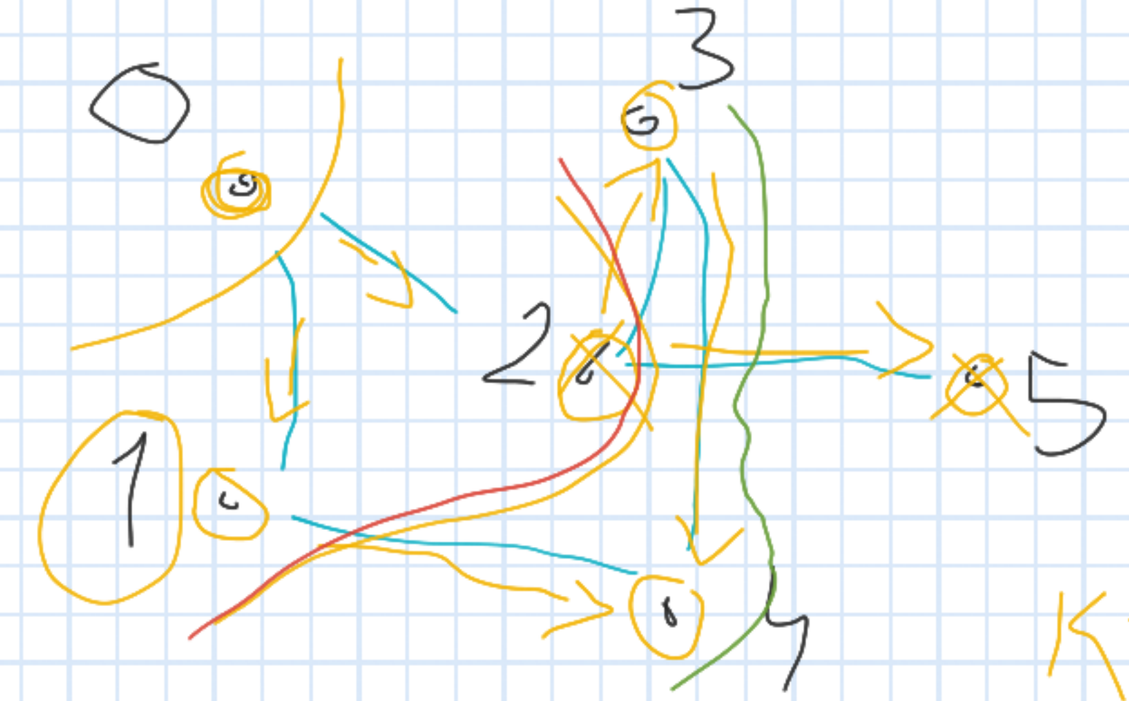
vis

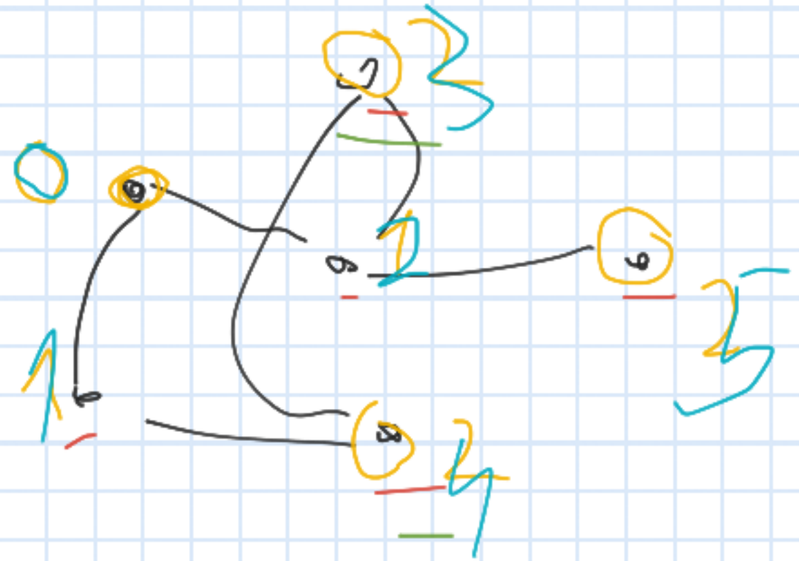


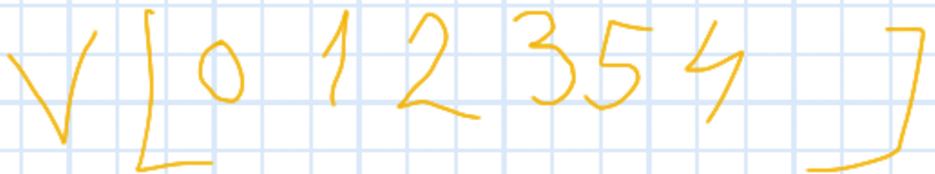

```
DFS(v, G):
  vis.app(v)
  foreach neigh:
    DFS(neigh, G)
```

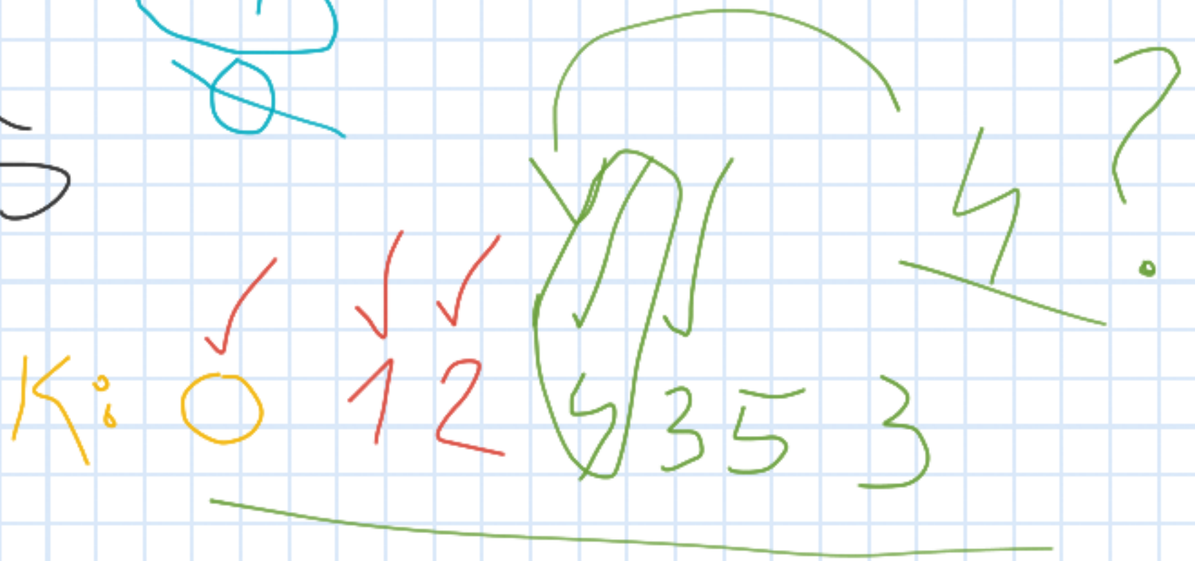
Zadanie 1: Detekcja cyklu

Napisz algorytm sprawdzający, czy graf nieskierowany posiada cykl.







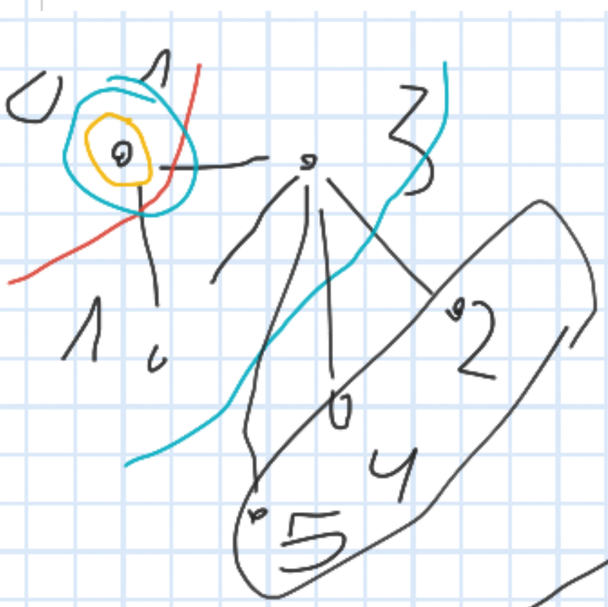


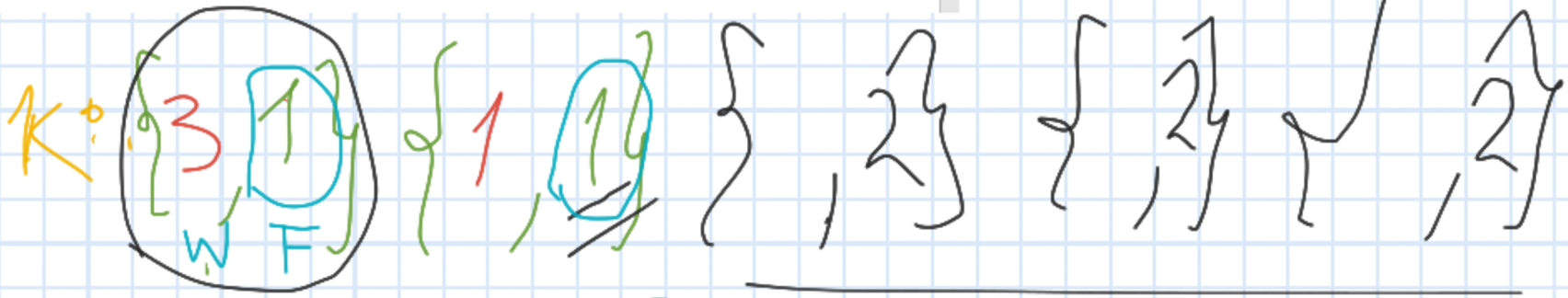
Zadanie 2: Wiadomość

Otrzymujemy na wejściu listę par ludzi, które się wzajemnie znają. Osoby są reprezentowane przez liczby od 0 do n - 1.

Dnia pierwszego osoba 0 przekazuje pewną wiadomość wszystkim swoim znajomym. Dnia drugiego każdy ze znajomych przekazuje tę wiadomość wszystkim swoim znajomym, którzy jej jeszcze nie znali, i tak dalej.

Napisz algorytm, który zwróci dzień, w którym najwięcej osób poznało wiadomość oraz ilość osób, które tego dnia ją otrzymały.





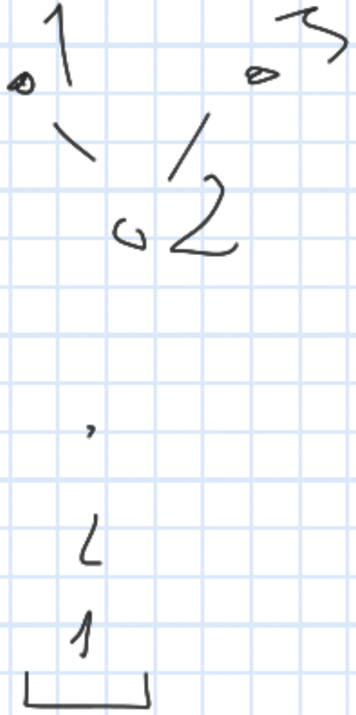
Zadanie 3: Jeziora

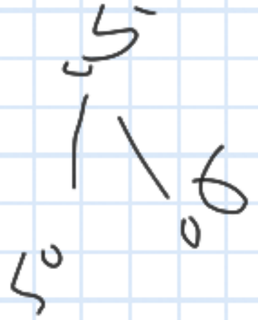
Dana jest dwuwymiarowa tablica N x N, w której każda komórka ma wartość “W” - reprezentującą wodę lub “L” - ląd. Grupę komórek wody połączonych ze sobą brzegami nazywamy jeziorem.

a) Policz, ile jezior jest w tablicy

b) Policz, ile komórek zawiera największe jezioro

L	W	L	L	L	L	L	L
L	W	L	W	W	L	L	L
L	L	L	W	W	L	W	L
L	W	W	W	W	L	W	L
L	L	W	W	L	L	L	L
L	W	L	L	L	L	W	W
W	W	L	W	W	L	W	L
L	L	L	W	L	L	L	L





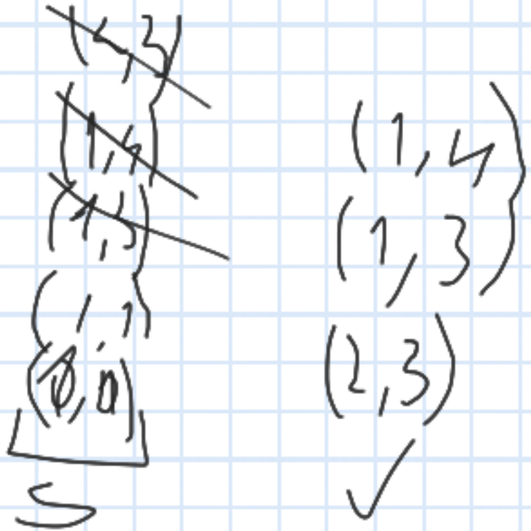
Bit Algo START

Zadanie 3: Jeziora - cd.

c) Zakładając, że pola o indeksach [0][0] i [n-1][n-1] są lądem, sprawdź czy da się przejść drogą lądową z pola [0][0] do pola [n-1][n-1]. Można chodzić tylko na boki, nie na ukos.

d) Znajdź najkrótszą ścieżkę między tymi punktami. Wypisz po kolei indeksy pól w tej ścieżce

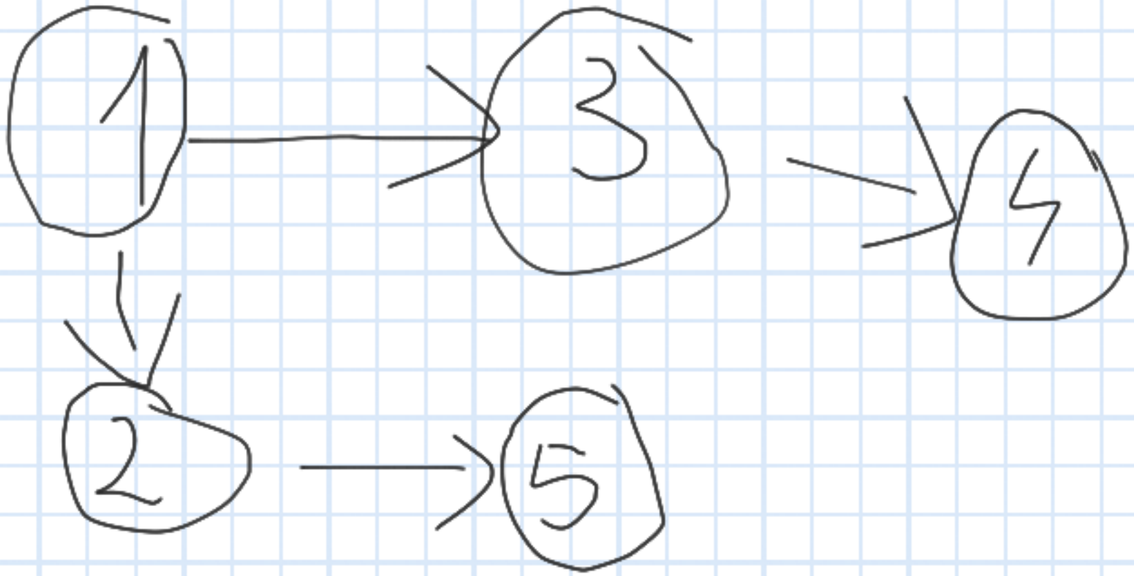
L	W	L	L	L	L	L	L
L	W	L	W	W	L	L	L
L	L	L	W	W	L	W	L
L	W	W	W	W	L	W	L
L	L	W	W	L	L	L	L
L	W	L	L	L	L	W	W
W	W	L	W	W	L	W	L
L	L	L	W	L	L	L	L



Zadanie 4: Sklejanie przedziałów

Dany jest ciąg przedziałów postaci $[a_i, b_i]$. Dwa przedziały można skleić, jeśli mają dokładnie jeden punkt wspólny. Podaj algorytm, który sprawdza, czy da się uzyskać przedział [a, b] poprzez sklejanie odcinków.

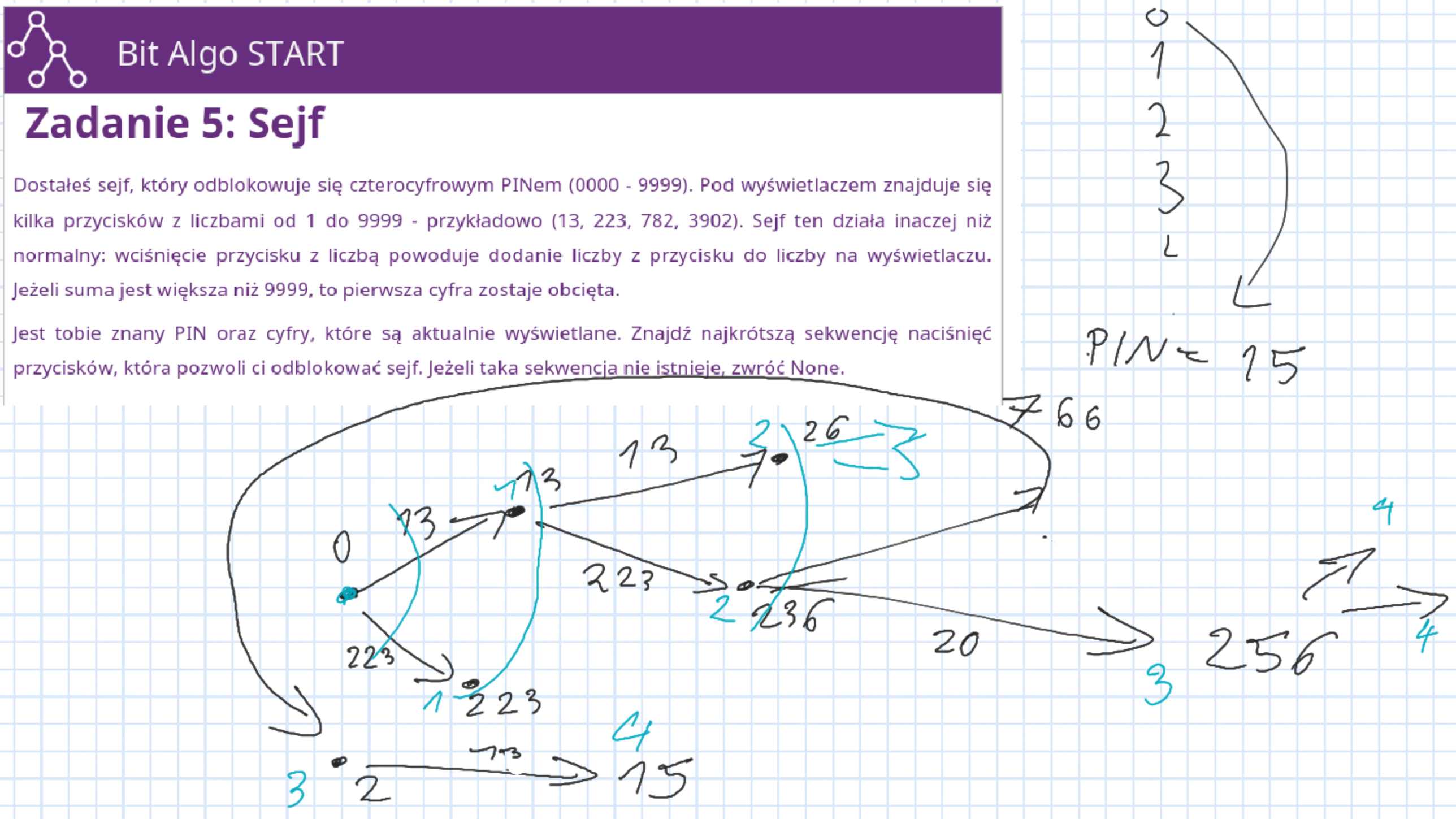
[(1, 3), (2, 5), (3, 4), (1, 2)]



Zadanie 5: Sejf

Dostałeś sejf, który odblokowuje się czterocyfrowym PINem (0000 - 9999). Pod wyświetlaczem znajduje się kilka przycisków z liczbami od 1 do 9999 - przykładowo (13, 223, 782, 3902). Sejf ten działa inaczej niż normalny: wciśnięcie przycisku z liczbą powoduje dodanie liczby z przycisku do liczby na wyświetlaczu. Jeżeli suma jest większa niż 9999, to pierwsza cyfra zostaje obcięta.

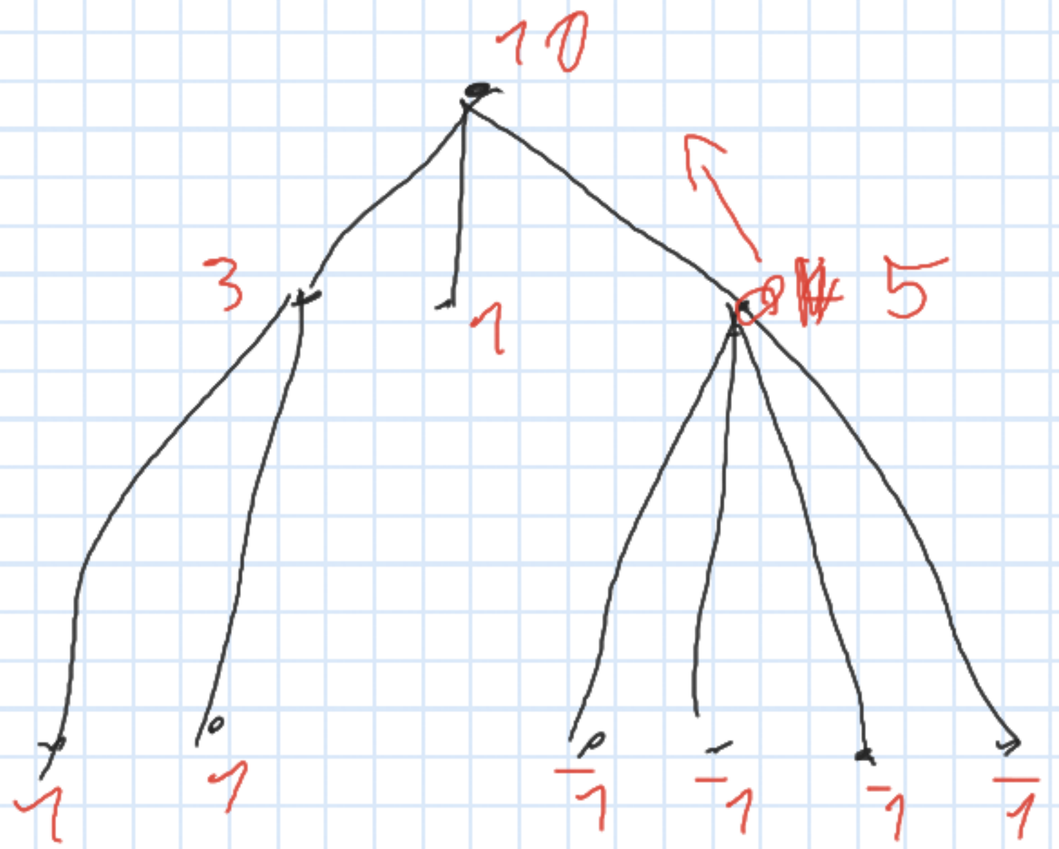
Jest tobie znany PIN oraz cyfry, które są aktualnie wyświetlane. Znajdź najkrótszą sekwencję naciśnięć przycisków, która pozwoli ci odblokować sejf. Jeżeli taka sekwencja nie istnieje, zwróć None.



Bit Algo START

Zadanie 6: Rozmiary poddrzew

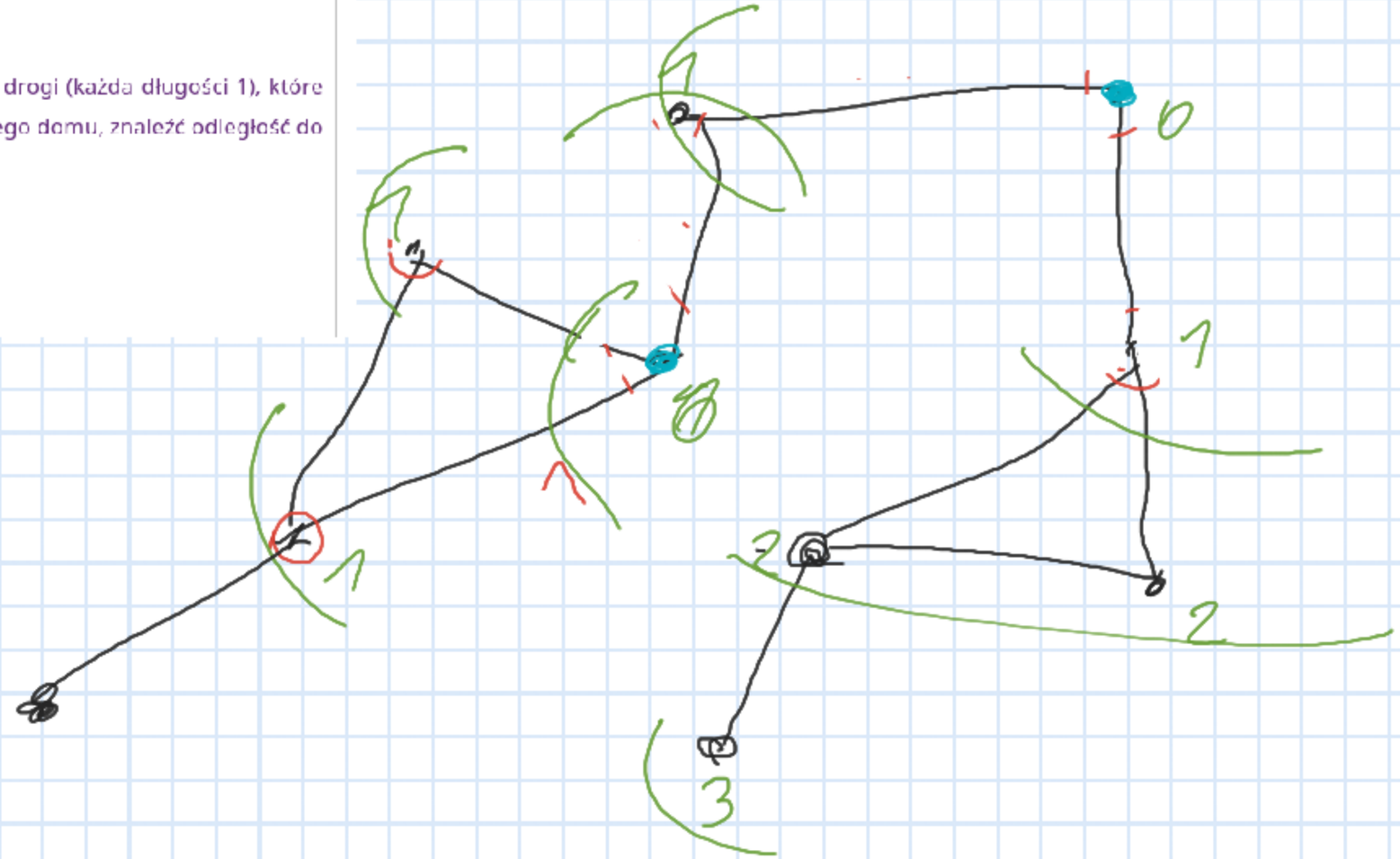
Dostajemy na wejściu listę krawędzi drzewa (niekoniecznie binarnego!) oraz wyróżniony wierzchołek - korzeń. Każdy wierzchołek tworzy swoje własne poddrzewo. Dla każdego wierzchołka, wyznacz ilość wierzchołków w jego poddrzewie.



Bit Algo START

Zadanie 7: Domy i sklepy

Mamy mapę miasteczka, w którym są domy i sklepy. Na mapie są również drogi (każda długości 1), które łączą dom z domem, albo dom ze sklepem. Naszym zadaniem jest, dla każdego domu, znaleźć odległość do najbliższego sklepu.



Bit Algo START

Zadanie 8: Średnica drzewa

Średnicą drzewa nazywamy odległość między jego najbardziej oddalonymi od siebie wierzchołkami. Napisz algorytm, który przyjmując na wejściu drzewo (niekoniecznie binarne!) w postaci listy krawędzi zwróci jego średnicę.