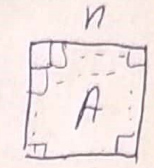
(1) Szukamy takiego i że:

$\forall_j \; A[i][j] == 1$

$\forall_j \; A[j][i] == 0$

reprezentacja grafu:



$A \neq B$

<u>Obserwacja:</u> Jeśli sprawdzamy czy istnieje krawędź $A \to B$

- jeśli istnieje to B nie może być źródłem bo ma krawędź wchodzącą
- jeśli nie istnieje to A nie może być źródłem, bo źródło ma krawędzie wychodzące do innych wierzchołków

Więc sprawdzając $A \to B$ możemy odrzucić jednego kandydata

<u>Algorytm:</u>

$1, 2, \dots, n$ - wierzchołki

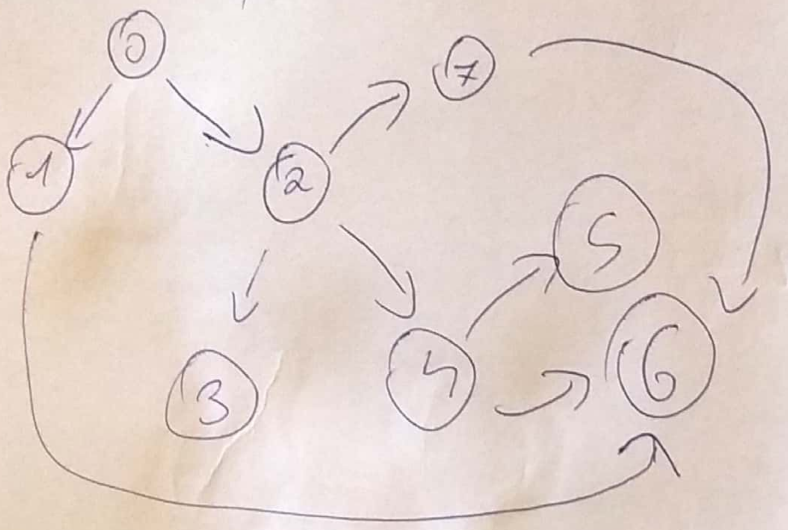
$A[n][n]$ - tablica 2D - sąsiedztwa

```
kandydat = 1
for(i=2; i<n; ++i)                 // eliminacja wierzchołków
    if(A[kandydat][i]==0)          posiadających krawędzie
        kandydat = i               wchodzące  O(n)

                                   // zostaje tylko jedna możliwość
for (i=0; i<kandydat; ++i)         // sprawdzamy czy są krawędzie
    if(A[kandydat][i]==0)          wychodzące z kandydata
        zwróć 'brak źródła'        (i> kandydat już sprawdzaliśmy)
                                   O(n)
for (i=0; i<n; ++i)                // sprawdzamy 'pionowo'
                                   czy kandydat nie ma
                                   krawędzi wchodzących
                                   O(n)
    if (A[i][kandydat] != 0)       Złożoność O(3n)=O(n)
        zwróć 'brak źródła'        // albo kandydat==i też
zwróć kandydat
```

MDL-11

② np: 0, 2, 7, 4, 5, 3, 1, 6



Idea:

- zapamiętujemy odwiedzone wierzchołki
- mamy stos S z którego wyrzucając po 1 elemencie dostaniemy wynik
- uruchamiamy DFS na każdym sąsiedzie wierzchołka
- po zakończeniu DFS dodajemy obecny wierzchołek na stos S.
- W ten sposób wierzchołki odwiedzone jako ostatnie będą najniżej na stosie, czyli odwrócony stos to sortowanie topologiczne wierzchołków

Algorytm

G - graf jako lista list (G[v] - sąsiedzi v)

S - stos

odwiedzony[n] = false

```
DFS(v):
    odwiedzony[v] = true        O(m)
    for(u in G[v]):
        if (odwiedzony[u] == false):
            DFS(u)
    S.push(v)
TopologicalSort()                O(n)
    for(i = 0; i < n; ++i):      // żeby zwiedzić cały graf
        if (odwiedzony[i] == false)
            DFS(i)
    while (S != ∅)  // wypisujemy wynik
        print S.top
        S.pop()
```

DFS na listach sąsiedztwa ma złożoność O(n+m)

3 Niech $\deg_{out}(v)$ - oznacza liczbę krawędzi wychodzących z wierzchołka v

Szukany wierzchołek w turnieju, z którego można dojść do każdego innego ścieżką po drodze o długości co najwyżej 2 to: $v^* = \max\{\deg_{out}(v) : v \in V\}$, $D(V,E)$

D-d (nie wprost): Załóżmy, że istnieje u t.że z v^* do u nie można dojść w co najwyżej 2 krokach.

W szczególności nie można w jednym kroku więc $(v^*,u) \notin E$ ale z def. turnieju każde pary musi być połączone, więc $(u,v^*) \in E$ (1)

Podobnie dla każdego wierzchołka połączonego bezpośrednio z v^*:

$\forall x\ (v^*,x) \in E \Rightarrow (u,x) \in E$ (bo inaczej $\exists$ (v^*,x,u) - ścieżka dł. 2) (2)

Czyli mamy że (z 1 i 2):

$\deg_{out}(u) \geq \deg_{out}(v^*) + 1$ bo $(u,v^*) \in E$

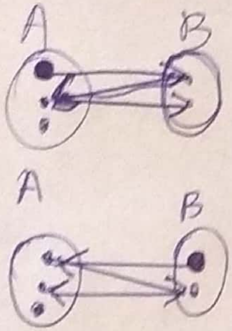
bo wszystkie krawędzie $(v^*,x) \in E$ implikują istnienie $(u,x) \in E$

Mamy sprzeczność, bo v^* miał mieć największe $\deg_{out}$

Czyli wierzchołek o największej liczbie krawędzi wychodzących jest tym szukanym. ∎

④ Jeśli $G=(V,E)$ jest grafem dwudzielnym t. że $V = A \cup B$ to aby G był Hamiltonowski (zawierający cykl Hamiltona) warunkiem koniecznym jest równoliczność A i B ($|A|=|B|$)

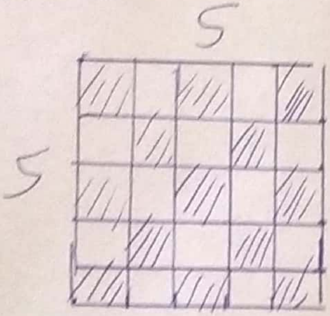
D-d (nie wprost): załóżmy bez straty ogólności że $|A|>|B|$ i $|A|=a$ oraz $|B|=b$.



Jeśli zaczęlibyśmy z A to po przejściu 2b krawędzi będziemy spowrotem w A, ale albo nie odwiedzimy wszystkich krawędzi w A, albo skończymy w innym wierzchołku niż zaczęliśmy, więc nie ma cyklu Hamiltona

Podobnie gdybyśmy zaczęli w B ■

5



Mamy 25 pól, 13 koloru A i 12 koloru B.

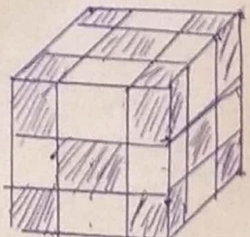
Zmieniając pole zmieniamy kolor (ruchy ⊞).

Problem można sprowadzić do szukania ~~ścieżki~~ cyklu Hamiltona w $G=(V,E)$, gdzie $V = A \cup B$, $|A| \neq |B|$, $|A|=13$, $|B|=12$

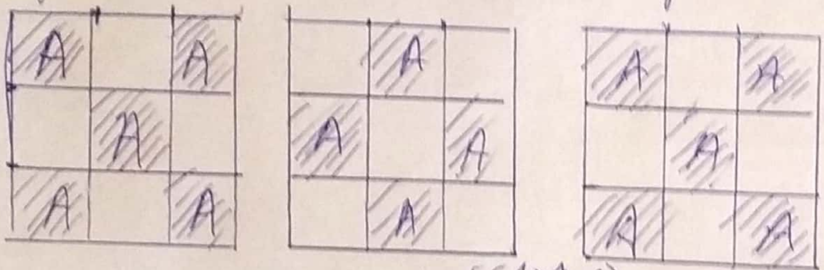
czyli nie zachodzi warunek konieczny, więc nie można wskazać takiej ścieżki skoczka.

⑤ Pomalujmy kostkę $3 \times 3 \times 3$ na 2 kolory.

Przekrój 3 ścian:



przednia środkowe tylna



Mysz przenosi się na sąsiednią kostkę $(1 \times 1 \times 1)$, a ta zawsze jest w innym kolorze, czyli A - wierzchołki kostek pomalowanych, B - wierzchołki niepomalowanych, a zadanie to szukanie ścieżki Hamiltona w dwudzielnym grafie $G=(V,E)=((A \cup B), E)$

$|A| = 14, |B| = 13$

Zaczynamy w rogu (pomalowanym $\in A$), a chcemy kończyć w środku kostki (niepomalowanym $\in B$), by to było możliwe to $|A| = |B|$ a są różne więc <u>nie jest to możliwe.</u>

np: (oznaczamy w którym zbiorze jesteśmy)

(A, B), (A, B, A, B) (A, B, A, B, A, B) .... itp.

gdyby na koniec mogła zjeść pomalowane pole to byłoby to możliwe bo (A, B, A) (A, B, A, B, A) itp. $|A| = |B|+1$ działa

⑥ Każdy turniej zawiera ścieżkę Hamiltona.

Turniej: $\forall u,v \; (u,v) \in E \lor (v,u) \in E$

D-d (indukcja): • dla $n \leq 2$ trywialnie zachodzi

• załóżmy że zachodzi pokażemy że zachodzi dla $n+1$ $\forall n \in \mathbb{N}$

$G = (V, E)$, $|V| = n+1$

Weźmy dowolny wierzchołek u z V

Pozostałe wierzchołki podzielmy na dwa podzbiory:

$V_{\leftarrow} = \{ \cancel{(x,u)\in} \, x : (x,u) \in E, x \in V \}$ - wierzchołki wchodzące do u

$V_{\rightarrow} = \{ x : (u,x) \in E, \cancel{x} \in V \}$ - wierzchołki wychodzące z u

Z def. turnieju $V = V_{\leftarrow} \cup V_{\rightarrow}$

Z zał. indukcyjnego ($|V_{\leftarrow}| \leq n$, $|V_{\rightarrow}| \leq n$, bo zabraliśmy u)

Więc w $V_{\leftarrow}$ i $V_{\rightarrow}$ istnieją ścieżki Hamiltona. Więc

$V_{\leftarrow}$, u, $V_{\rightarrow}$ można połączyć otrzymując ścieżkę Hamiltona

przechodzącą przez wszystkie $n+1$ wierzchołków

np: