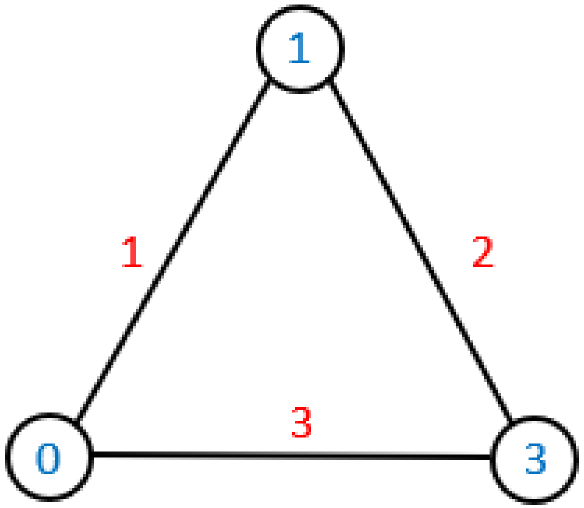
Граціозна розмітка графів

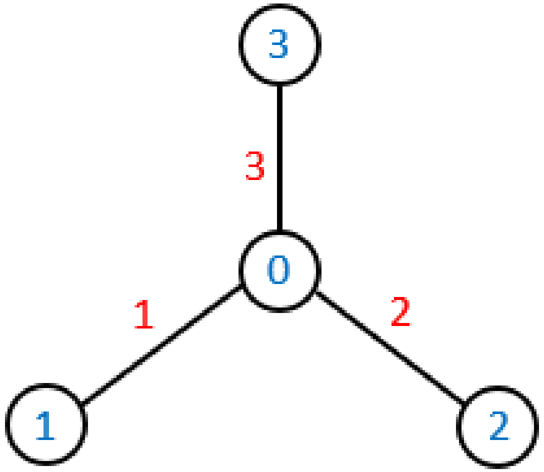
АВЕРКІН ОЛЕКСАНДР

ТРАВЕНЬ 2023

Мета

1. Дослідити і структурувати означення, твердження і теореми з граціозної розмітки графів.
2. Представити огляд граціозних алгоритмів маркування для різних класів графів.
3. Реалізувати програмний модуль мовою Python для побудови граціозної розмітки графів.





Розмітка графа

- *Вершинною розміткою* графа G = (V, E) є функція з множини вершин V у множину міток N ($f : V \to N$).
- *Реберною розміткою* графа G = (V, E), в свою ж чергу, є функція з множини ребер E у множину міток N ($g : E \to N$).
- ***Граціозною розміткою*** графа G з n ребрами називають таку ін'єктивну функцію вершинної розмітки $f : V \to [0, n]$, та ін'єктивну функцію реберної розмітки $f\gamma : E \to [1, n]$, що $f\gamma(uv) = |f(u) - f(v)|$.

Умова парності
та реберно-послідовна граціозність графа

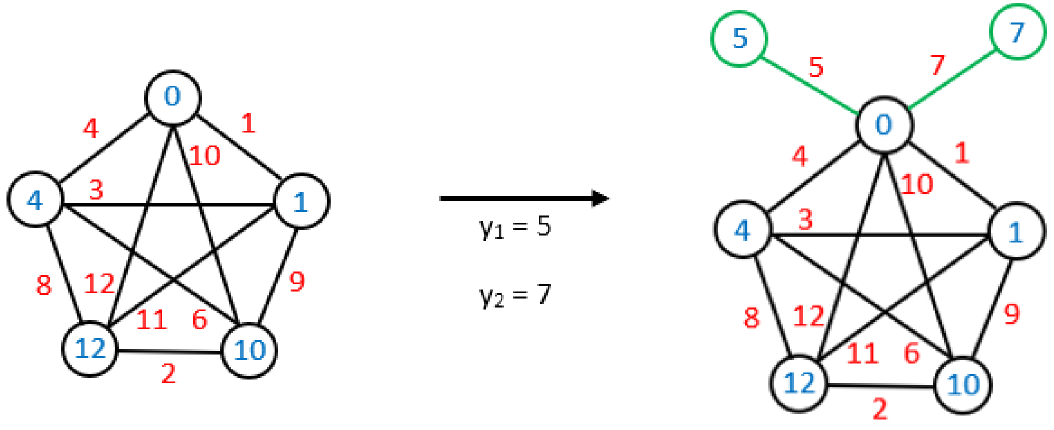
Теорема. [1] Нехай G — ейлеровий граф. Якщо m ≡ 1, 2 (mod 4), то G не є граціозним.

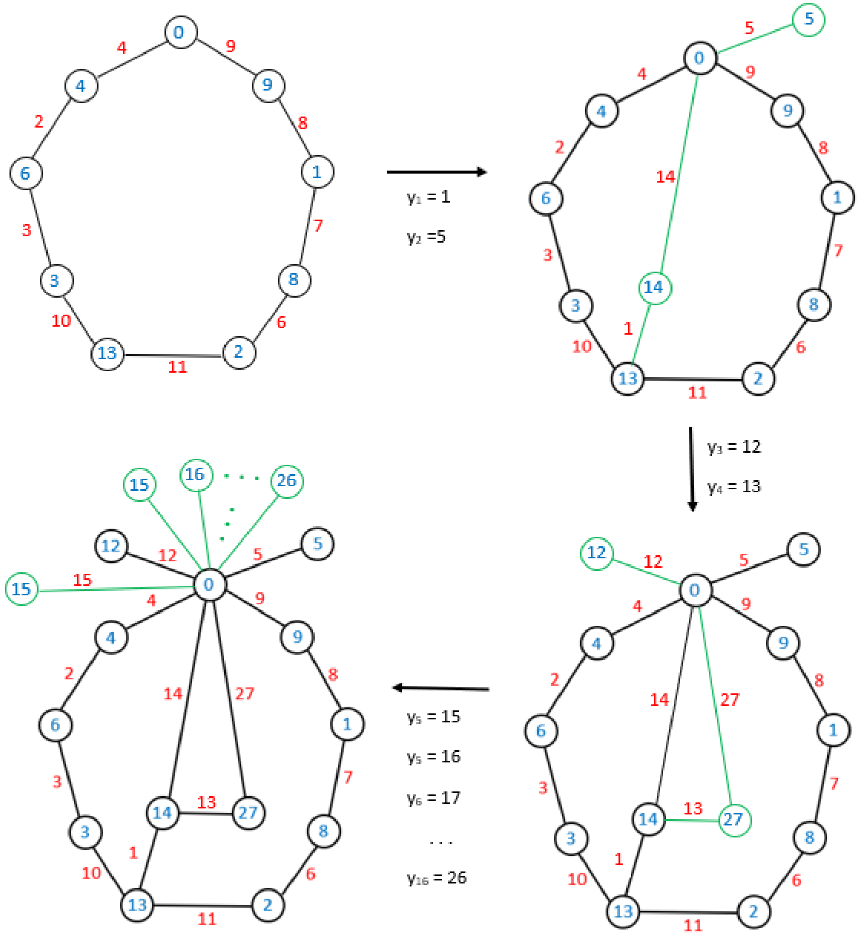
Теорема. [2] Кожен граф є індукованим підграфом граціозного графа.

Алгоритм побудови граціозного графа з неграціозного, який міститься в ньому як індукований підграф:

нехай задані розмітки f : V → [0, k] (ін'єктивна для деякого k ≥ m) та f_γ : E → N (також ін'єктивна), і існують u,v ∈ V з f (u) = 0 і f (v) = k. Нехай $\{y_1, ..., y_k\}$ – множина відсутніх міток ребер графа, $\{x_i, ..., x_r\}$ – множина міток вершин графа.

Додамо вершину w_i : $\begin{cases} f(w_i) = y_i;\ f_\gamma(uw_i) = y_i, & \text{якщо } y_i \notin \{x_i, ..., x_r\} \\ f(w_i) = k + y_i;\ f_\gamma(uw_i) = k + y_i;\ f_\gamma(vw_i) = y_i, & \text{якщо } y_i \in \{x_i, ..., x_r\} \end{cases}$





Реберно-послідовна граціозність графа

ПРИКЛАД. Конструювання граціозного графа з K_5 та C_9

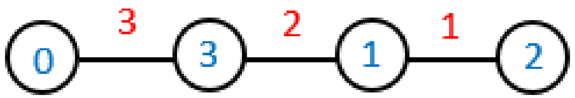
Граціозна розмітка графів-ланцюгів P_n

Твердження [1] .

Граф-ланцюг P_n є граціозним для всіх $n \geq 1$.

Граціозна розмітка P_n :

$$\begin{cases} \frac{k}{2}, & \text{якщо } k - \text{парне} \\ n - \frac{(k+1)}{2}, & \text{якщо } k - \text{непарне} \end{cases}$$

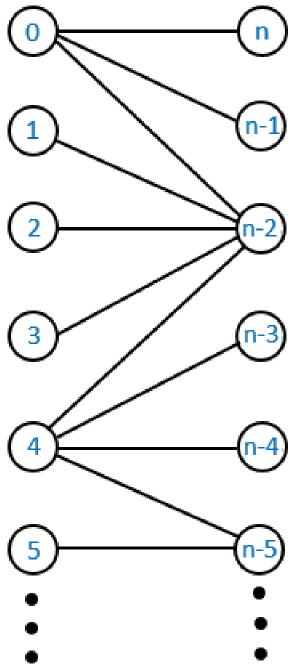



```python
import networkx as nx

#перевірка чи є граф ланцюгом
def is_pathgraph(G):
    degrees = dict(G.degree())
    ends = [v for v, d in degrees.items() if d == 1]
    if len(ends) != 2:
        return False
    for v, d in degrees.items():
        if d != 2 and v not in ends:
            return False
    return True

#Функція, що граціозно маркує граф-ланцюг
def pathgraph_labeling(G):
  if not is_pathgraph(G):
        print("Граф не є ланцюгом")
        return
  n = len(G)
  labels = {}
  for node in sorted(G.nodes()):
    if(node % 2 == 0):
      labels[node] = node//2
    else:
      labels[node] = n - (node+1)//2
  return labels

# Приклад використання
G = nx.Graph([(0, 1), (1, 2), (2, 3)])
labels = pathgraph_labeling(G)

print("Граціозна розмітка: ", labels)
```

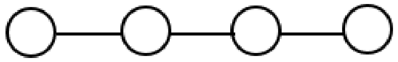
```
Граціозна розмітка:  {0: 0, 1: 3, 2: 1, 3: 2}
```

Граціозна розмітка гусеничних дерев

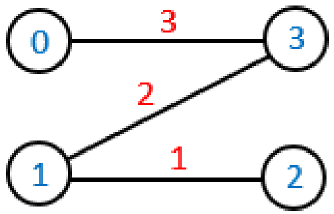
Твердження. [1] Всі гусеничні дерева є граціозними.



<u>Приклад.</u> Представимо граф P_4 (рисунок а) як планарний та застосуємо схему граціозної розмітки як для гусеничного дерева (рисунок б).



а)



б)

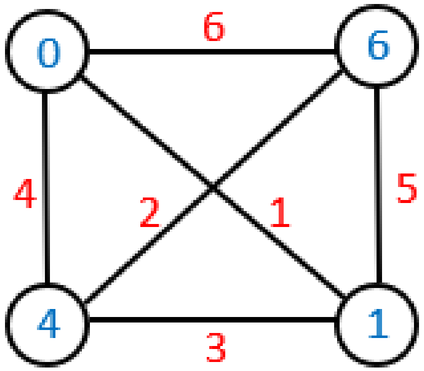
Граціозна розмітка повних графів K_n

Твердження. [1]

Повний граф K_n є граціозним тоді і тільки тоді, коли $n \leq 4$.

Граціозна розмітка K_n :

1. Позначаємо будь-яку вершину міткою 0, і будь-яку іншу вершину міткою m
2. Третій вершині даємо мітку 1
3. Останній вершині без мітки надаємо мітку 4

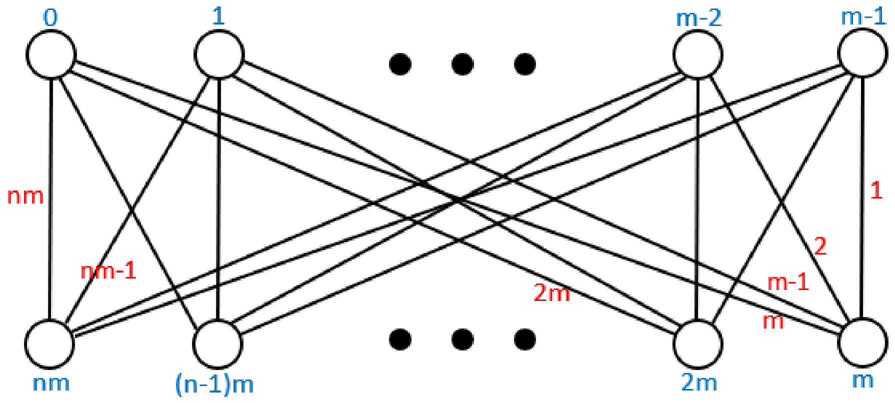



```python
import networkx as nx

# перевірка чи є граф повним (кількість ребер у графі має
# дорівнювати n(n-1)/2, де n - кількість вершин у графі)
def isComplete(graph):
  n = len(G.nodes())
  if len(G.edges()) == n*(n-1)/2:
    return True
  else:
    return False

#Функція, що граціозно маркує повний граф
def completeGraph_labeling(graph):
  n = len(G.nodes())
  m = len(G.edges())
  if isComplete(G) and n <= 4:
    n = len(graph)
    labeling = {}
    if n == 1:
        labeling[0] = 0
    elif n == 2:
        labeling[0] = 0
        labeling[1] = m
    elif n == 3:
        labeling[0] = 0
        labeling[1] = m
        labeling[2] = 1
    elif n == 4:
        labeling[0] = 0
        labeling[1] = m
        labeling[2] = 1
        labeling[3] = 4
    else:
        return None
    return labeling
  else:
    print("Граф не є повним або має більше 4 вершин.")

# Приклад використання
G = nx.Graph([(0, 1), (0, 2), (0, 3), (1,2), (1,3), (2,3)])
labeling = completeGraph_labeling(G)
print("Граціозна розмітка: ", labeling)
```

```
Граціозна розмітка:  {0: 0, 1: 6, 2: 1, 3: 4}
```

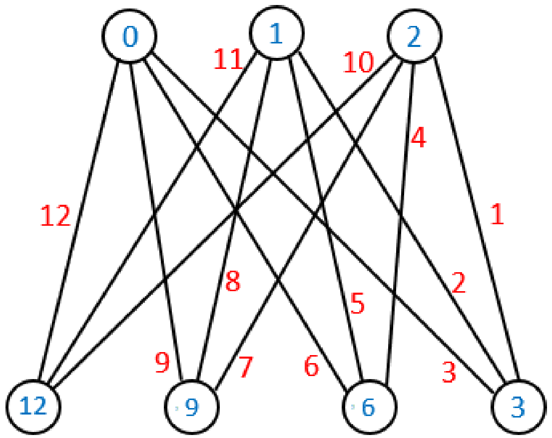
Граціозна розмітка повних двочасткових графів $K_{m,n}$

Твердження. [1] Повний двочастковий граф $K_{m,n}$ є граціозним для всіх m, n ≥ 1.




```
import networkx as nx

#Функція, що граціозно маркує повний двочастковий граф
def completeBipartiteGraph_labeling(m, n):
    graph = nx.complete_bipartite_graph(m, n)
    labeling = {}
    for i in range(m):
        labeling[i] = i
    for j in range(n):
        labeling[m+j] = (j+1)*m
    return labeling

# Приклад використання
m, n = 3, 4
labeling = completeBipartiteGraph_labeling(m, n)
print("Граціозна розмітка: ", labeling)
```

```
Граціозна розмітка:  {0: 0, 1: 1, 2: 2, 3: 3, 4: 6, 5: 9, 6: 12}
```



Реалізація на Python побудови граціозної розмітки для $K_{3,4}$

Граціозна розмітка циклів C_n

Твердження. [1]
Цикл C_n є граціозним тоді і тільки тоді, коли $n \equiv 0, 3 \pmod 4$.

Граціозна розмітка C_n :

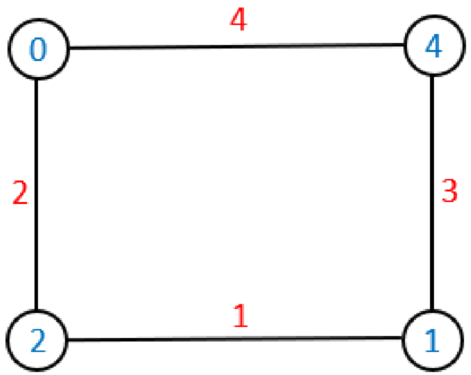
- Якщо $n \equiv 0 \pmod 4$, то вершини маркуються за наступною формулою:

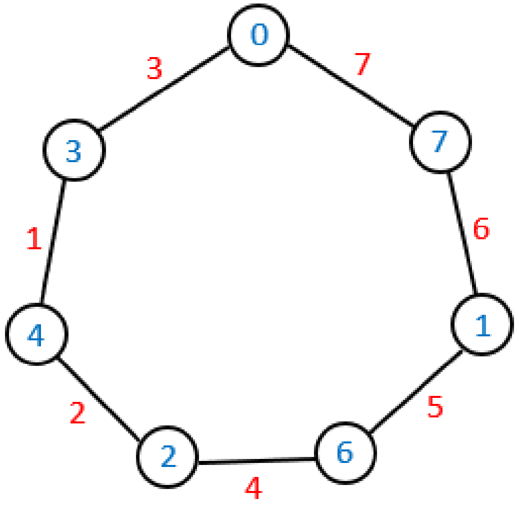
$$\begin{cases} \frac{i}{2}, & \text{якщо } i = 0, 2, 4, \ldots, n-2 \\ n - \frac{(i-1)}{2}, & \text{якщо } i = 1, 3, 5, \ldots, \frac{n}{2} - 1 \\ n - \frac{(i-1)}{2} - 1, & \text{якщо } i = \frac{n}{2} + 1, \frac{n}{2} + 3, \ldots, n-1 \end{cases}$$

- Якщо $n \equiv 3 \pmod 4$, то вершини маркуються за наступною формулою:

$$\begin{cases} \frac{i}{2}, & \text{якщо } i = 0, 2, 4, \ldots, n-1 \\ n - \frac{(i-1)}{2}, & \text{якщо } i = 1, 3, 5, \ldots, \frac{n+1}{2} - 1 \\ n - \frac{(i-1)}{2} - 1, & \text{якщо } i = \frac{n+1}{2} + 1, \frac{n+1}{2} + 3, \ldots, n-2 \end{cases}$$

Реалізація на Python побудови граціозної розмітки для C_4 та C_7






```python
import networkx as nx

#перевірка чи є граф циклом
def is_cycle(G):
    if not nx.is_connected(G):
        return False
    for node in G.nodes:
        if G.degree(node) != 2:
            return False
    return True

#Функція, що граціозно маркує цикл
def cycle_labeling(G):
  if not is_cycle(G):
    print("Граф не є циклом")
    return
  n = G.number_of_nodes()
  labels = {}
  if(n % 4 == 0):
    for i in range(n):
      if i in range(n//2 + 1, n, 2):
        labels[i] = n - (i-1)//2 - 1
      elif (i % 2 == 0 and i < n-1):
        labels[i] = i // 2
      elif i in range(1, (n//2), 2):
        labels[i] = n - (i-1)//2
    return labels
  elif (n % 4 == 3):
    for i in range(n):
      if i in range((n+1)//2 + 1, n-1, 2):
        labels[i] = n - (i-1)//2 - 1
      elif(i % 2 == 0):
        labels[i] = i // 2
      elif i in range(1, ((n+1)//2), 2):
        labels[i] = n - (i-1)//2
    return labels
  else:
      return("Для даної к-ті вершин циклу неможливо задати граціозну розмітку")

# Приклад використання
C4 = nx.Graph([(0, 1), (1, 2), (2, 3), (3, 0)])
labeling_C4 = cycle_labeling(C4)
print("Граціозна розмітка C4: ", labeling_C4)

C7 = nx.Graph([(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 0)])
labeling_C7 = cycle_labeling(C7)
print("Граціозна розмітка C7: ", labeling_C7)
```

```
Граціозна розмітка C4:  {0: 0, 1: 4, 2: 1, 3: 2}
Граціозна розмітка C7:  {0: 0, 1: 7, 2: 1, 3: 6, 4: 2, 5: 4, 6: 3}
```

Граціозна розмітка графів-колес Wp

Твердження. [1]

Колесо W_p є граціозним для всіх $p \geq 3$.

Граціозна розмітка W_p :

Нехай $V(W_p) = \{u_0, u_1, \ldots, u_{p-1}, v\}$ — множина вершин, де v — вершина, з'єднана з циклом. Розглянемо два випадки:

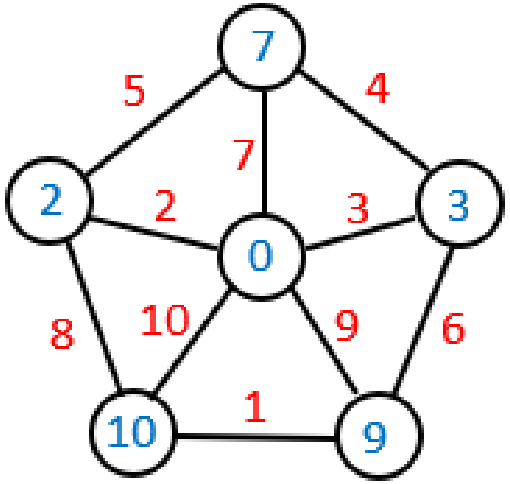
- Якщо $p \equiv 0 \pmod 2$, то вершини маркуються за наступною формулою:

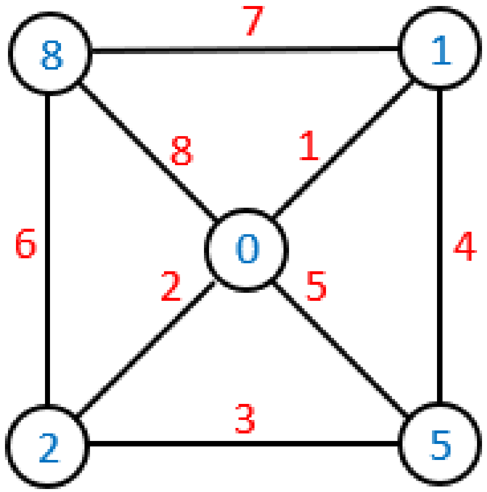
$$\begin{cases} 2p, & \text{якщо } i = 0 \\ 2, & \text{якщо } i = p-1 \\ i, & \text{якщо } i = 1, 3, 5, \ldots, p-3 \\ 2p - i - 1, & \text{якщо } i = 2, 4, 6, \ldots, p-2 \end{cases}$$

- Якщо $p \equiv 1 \pmod 2$, то вершини маркуються за наступною формулою:

$$\begin{cases} 2p, & \text{якщо } i = 0 \\ 2, & \text{якщо } i = 1 \\ p + i, & \text{якщо } i = 2, 4, 6, \ldots, p-1 \\ p + 1 - i, & \text{якщо } i = 3, 5, 7, \ldots, p-2 \end{cases}$$

Реалізація на Python побудови граціозної розмітки для W_5 та W_4





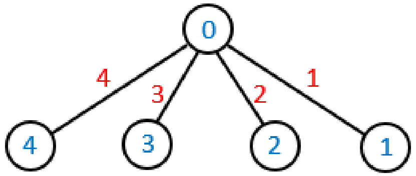

```
#Функція, що граціозно маркує колесо
def wheel_labeling(graph):
    p = len(graph) - 1
    labeling = {}
    if(p % 2 == 0):
      for i, vertex in enumerate(graph):
        if vertex == "v":
          labeling[vertex] = 0
        elif i == 0:
            labeling[vertex] = 2*p
        elif i == p-1:
            labeling[vertex] = 2
        elif i % 2 == 1:
            labeling[vertex] = i
        else:
            labeling[vertex] = 2*p - i - 1
    if(p % 2 == 1):
      for i, vertex in enumerate(graph):
        if vertex == "v":
          labeling[vertex] = 0
        elif i == 0:
          labeling[vertex] = 2*p
        elif i == 1:
          labeling[vertex] = 2
        elif i % 2 == 0:
          labeling[vertex] = p + i
        else:
          labeling[vertex] = p + 1 - i
    return labeling

# Приклад використання
W5 = ["u0", "u1", "u2", "u3", "u4", "v"]
labeling_W5 = wheel_labeling(W5)
print(labeling_W5)

W4 = ["u0", "u1", "u2", "u3", "v"]
labeling_W4 = wheel_labeling(W4)
print(labeling_W4)
```

```
{'u0': 10, 'u1': 2, 'u2': 7, 'u3': 3, 'u4': 9, 'v': 0}
{'u0': 8, 'u1': 1, 'u2': 5, 'u3': 2, 'v': 0}
```

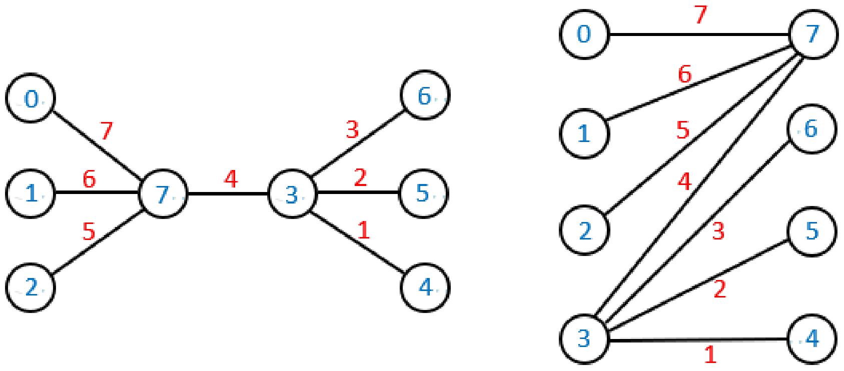
Граціозна розмітка дерев

- 1976 р. була сформульована гіпотеза, що всі дерева є граціозними [3].

Граціозна розмітка дерева діаметру 1



Граціозна розмітка дерева діаметру 2



(а) (б)

Граціозна розмітка дерева діаметру 3 (а) та його представлення у вигляді планарного двочасткового графа (б)

Граціозна розмітка дерев

- У 2010 році Фанг [4] довів, що всі дерева до 35 вершин є граціозними.

Результат Фанга замінює попередні результати в цьому напрямку: Олдред і МакКей [5] встановили в 1998 році, що всі дерева до 27 вершин є граціозними, а Хортон [6] підтвердив у 2003 році, що всі дерева з не більше ніж 29 вершинами є граціозними.

Список літератури

[1]. Rodrigo Ming Zhou. Graceful labeling of graphs. Rio de Janeiro, 2016. 40 p. URL: https://www.cos.ufrj.br/uploadfile/publicacao/2668.pdf

[2]. Arumugam S., Bagga J. Graceful labeling algorithms and complexity – a survey. Journal of the indonesian mathematical society. 2012. URL: https://doi.org/10.1109/iwaci.2010.5585144

[3]. Theory of Graphs International Symposium Rome 1966. Theory of graphs, international symposium: theorie des graphes, journees internationales d'etude; [papers]. New York : Gordon and Breach, 1967. P. 349-355.

[4]. A new approach of the division optimization of attribute spaces for the decision tree construction / D. Zhang et al. 2010 third international workshop on advanced computational intelligence (IWACI), Suzhou, China, 25–27 August 2010. 2010. URL: https://doi.org/10.1109/iwaci.2010.5585144

[5]. Aldred R. E. L., McKay B. D. Graceful and harmonious labellings of trees. Bulletin of the Institute of Combinatorics and its Applications 23 (1998). P. 69–72.

[6]. Horton M. Graceful trees: statistics and algorithms : thesis. 2003. URL: https://eprints.utas.edu.au/19/1/GracefulTreesStatisticsAndAlgorithms.

Дякую за увагу!