Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ”

Кафедра математики факультету інформатики

Курсова робота на тему:
Королі в турнірах

Керівник курсової роботи:
к. ф.-м. н. *Козеренко С.О.*
(прізвище та ініціали)

_____
(підпис)
“____”_____ 2023 р.

Виконав студент
3-го року навчання спеціальності
113 “Прикладна математика”
Севергін Олександр Вадимович
(ПІБ)

Київ – 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ”

Кафедра математики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри математики,
доц., к.ф-м.н.
_____ Р. К. Чорней
(підпис)
“____”_____ 2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу
студенту 3-го курсу факультету інформатики
Севергіну Олександру Вадимовичу

Тема:Королі в турнірах.
Вихідні дані: Досліджено королів у турнірах.
Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Анотація
Вступ
1 Основіні означення
2 Королі в турнірах
2.1 Королі в турнірах T
2.2 Сильні королі в турнірах T
2.3 Королі в тензорному добутку графів $G_1 \times G_2$
3 Алгоритми
3.1 Необхідні функції
3.2 Алгоритми на турнірах T
Висновки
Література

Дата видачі “____”_____ 2022 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Тема: Королі в турнірах.
Календарний план виконання роботи:

Номер	Назва етапу курсової	Термін виконання етапу	Примітка
1.	Отримання теми курсової роботи.	вересень 2022 р.	
2.	Ознайомлення з темою курсової.	вересень 2022 р.	
3.	Розробка плану та структури роботи.	жовтень 2022 р.	
4.	Робота з науковою літературою, опис основних означень з теорії графів.	листопад 2022 р.	
5.	Дослідження основних властивостей королей в турнірах.	листопад - грудень 2022 р.	
6.	Дослідження королей в турнірах, їх типів та реалізація алгоритмів мовою програмування Python. Отримання основних результатів.	січень - квітень 2023 р.	
7.	Робота над текстовим оформленням результатів.	квітень 2023 р.	
8.	Попередній аналіз курсової. Виправлення помилок.	початок травня 2023 р.	
9.	Захист курсової роботи. роботи.	кінець травня 2023 р.	

Зміст

Анотація **5**

Вступ **6**

1 Основні означення **7**

2 Королі в турнірах **9**
2.1 Королі в турнірах T 9
2.2 Сильні королі в турнірах T 12
2.3 Королі в тензорному добутку орієнтованих графів $G_1 \times G_2$ 12

3 Алгоритми **14**
3.1 Необхідні функції 14
3.2 Алгоритми на турнірах T 17

Висновки **27**

Висновки **28**

Література **29**

Анотація

Нехай T – повний орієнтований граф, а вершина $v \in V(T)$ є королем, тоді й тільки тоді, коли $\forall u \in V(T)$ існує елементарний шлях з вершини v у вершину u довжиною 1 або 2. Мета роботи полягає в дослідженні королів у турнірах і їх властивостей. У курсовій роботі доведені теореми, що стосуються різних властивостей королів, і реалізовані алгоритми пошуку королів й побудови турнірів мовою програмування Python. Також розглянуті наступні поняття: сильний король, королі в двочасткових турнірах і королі в тензорному добутку орієнтованих графів.

Ключові слова: орієнтований граф, турнір, король, сильний король.

Вступ

Турніри - це не нове поняття в теорії графів. Основні поняття ввів американський математик-біолог і соціолог Х. Г. Ландау (Hyman Garshin Landau) у своїй роботі "On dominance relations and the structure of animal societies. III. The condition for a score structure яка була видана в 1953 році. У цій роботі він також використовував поняття королів, хоча сам його фактично не вводив. Також Ландау ввів теорему про послідовність результатів турніру, яку пізніше було доведено. Саме поняття турнір було введено пізніше на основі роботи Ландау.

Турніри вивчалися багатьма світовими математиками, наприклад у Дж. В. Муна є ціла робота присвячена турнірам [2], С. Б. Маурер ввів поняття короля й кріпака, а К. Б. Рейд досліджував існування турнірів з довільною кількістю вершин й королів, з певними винятками.

Турніри мають місце не тільки в математиці, а й в реальному житті. Наприклад, за допомогою турнірів вивчають теорію голосування і теорію соціального вибору, що є дуже актуальним, особливо, в наші часи. Тому, досліджувати тему королів у турнірах цікаво.

У цій курсовій роботі ми вивчаємо основні властивості турнірів і королів у них, вивчаємо різноманітні типи королів, доводимо теореми про існування турнірів на n вершинах і з заданою кількістю королів, існування королів у турнірах, існування сильних королів у турнірах й умови існування короля в тензорному добутку орієнтованих графів. Також у роботі реалізуємо алгоритми пошуку королів у турнірах, пошуку сильних королів як у звичайних турнірах, так і у двочасткових, а також побудови турнірів на n вершинах з k королями.

У літературі використаної під час написання роботи можна знайти як і загальну інформацію про теорію графів, так і доволі специфічну. Наприклад, в роботі Дугласа Бр. Веста "Introduction in Graph Theory" [1] наводять основні поняття з теорії графів, а також означення турнірів, королей в турнірі. У роботі Дж. В. Муна [2] можна зустріти означення регулярних графів і вправи, що стосуються них. А в книзі М. Х. МакЕндрю [4] досліджується добуток орієнтованих графів, що потім був використаний М. Норджем у його роботі [5].

1 Основні означення

Означення 1.1. *Граф* G — це впорядкована пара (V, E), де V — множина вершин або вузлів, E — множина пар вершин з V, які називаються ребрами.

Означення 1.2. *Орієнтований граф (орграф)* G — це граф у якого всі ребра мають напрямок.

Означення 1.3. Пара вершин графа i, j графа G називається *суміжними*, якщо $ij \in E(G)$.

Означення 1.4. *Множина сусідів вершини* $v \in V$ - це множина всіх вершин графа G, які є суміжними вершині v. Позначимо $N^+(v)$ - множину сусідів для вершини v, що $\forall\, x \in N^+(v)\ \exists\, arc(v, x)$ і $N^-(v)$ - множину сусідів для вершини v, що $\forall\, x \in N^-(v)\ \exists\, arc(x, v)$.

Означення 1.5. *Повний граф* G - це граф, у якому кожна пара вершин i, j $(i \neq j)$ є суміжною.

Означення 1.6. *Двочастковий граф* $G(V, E)$ - це граф множину вершин V якого розбита на дві підмножини таким чином, щоб кожне ребро графа G мало одну вершину з першої підмножини і одну з другої.

Означення 1.7. *Повний двочастковий граф* $G(V, E)$ - це двочастковий граф, у якого кожна вершина з будь-якої підмножини з'єднанами ребрами з усіма вершинами іншої підмножини.

Означення 1.8. *Степінь вершини орієнтованого графа* - це число ребер, які виходять із вершини v і позначають $deg^+(v)$ й число ребер, які входять у вершину v і позначають $deg^-(v)$.

Означення 1.9. *Рахунок вершини турніра* - це кількість вершин в турнірі, до яких проведено ребро з даної, тобто існує $(v, e) \in E$ для вершини v.

Означення 1.10. *Матриця суміжності* - це матриця розмірності nxn, яка задає граф, де v_{ij} дорівнює 0, якщо $arc(i, j) \notin E$ або дорівнює 1, якщо $arc(i, j) \in E$.

Означення 1.11. *Ланцюг* - це послідовність $v_0e_1v_1e_2...v_ie_i$, що складається з вершин $v_i \in G(V)$ і ребер $e_i \in G(E)$, що з'єднують ці вершини. Вершини й ребра в ланцюгу можуть повторюватись.

Означення 1.12. *Елементарний шлях в орієнтованому графі* G - це ланцюг у якого жодна вершина не повторюється і всі ребра є дугами в $G(E)$, і напрямлені в напрямку руху від початкової до кінцевої вершини.

Означення 1.13. *Циклом у графі* G називається ланцюг у якому перша та остання вершини однакові.

Означення 1.14. *Орієнтованим циклом у графі* G називається цикл у якому кожне ребро через, яке він проходить, є дугою орієнтованого графа G і з'єднує попередню вершину з наступною в циклі.

Означення 1.15. *Турнір* T - це орієнтований повний граф.

Означення 1.16. *Двочастковий турнір* T - це повний двочастковий граф, усі ребра якого мають напрям.

Означення 1.17. *k-король у турнірі* T - це така вершина $v \in T$, що має елементарний шлях довжини k або менше у будь-яку іншу вершину з турніра T.

Означення 1.18. *Король у турнірі* T - це 2-король.

Означення 1.19. *Король у двочастковому турнірі* T - це 4-король.

Означення 1.20. *Сильний король (англ. strong king) у турнірі* T - це така вершина $v \in T$, що для кожної іншої вершини $u \in T$ $b(v,u) > b(u,v)$, де $b(v,u)$ - кількість елементарних шляхів довжиною 2 з вершини v у вершину u.

Означення 1.21. *Тензорний добуток* $G \times H$ графів G і H - це граф, у якого множина вершин є декартовим добутком $V(G) \times V(H)$ і різні пари (u,u') і (v,v') є суміжними в $G \times H$ тоді й тільки тоді, коли u суміжна з v і u' суміжна з v'.

Означення 1.22. *d(D)* - це найбільший спільний дільник довжин всіх орієнтованих циклів в графі D.

Означення 1.23. $g_D(v)$ - це найбільший спільний дільник довжин всіх орієнтованих циклів у графі D, що містять вершину v. $g_D(v) = \infty$, якщо такі цикли відсутні.

Означення 1.24. $gcd(a,b)$ - це добуток $a * b$ поділений на найменше спільне кратне a і b.

Означення 1.25. *Сильнозв'язний орграф* G - це такий граф G, у якому для кожної пари вершин $v, u \in G$ існує напрямлений шлях з вершини v у вершину u і навпаки.

Означення 1.26. *Компонента зв'язності* C - це зв'язний підграф графа G,який не є складовою іншого більшого зв'язного графа.

Означення 1.27. *Сильна компонента* C - це найбільша компонента зв'язності орграфа G, що є сильнозв'язним підграфом графа G.

Означення 1.28. *Регулярний турнір* T - це такий турнір T, у якого степені вершин є рівними $\forall v \in T$.

Усі ці означення можна знайти в книгах [1], [2], [4] й в роботі [5].

2 Королі в турнірах

2.1 Королі в турнірах T

Теорема 2.1. [1, p. 66] *Для кожного $n \in \mathbb{N}$ існує турнір на n вершинах, у якому кожна вершина є королем, тоді й тільки тоді, коли $n \neq \{2; 4\}$.*

Доведення. Доведемо ітеративно.

Для $n = 1$ матимемо такий турнір:

Королем є вершина 1.

Для $n = 2$ матимемо такий турнір:

Королем є вершина 1.

Для $n = 3$ матимемо турнір, що зображено на рис. 2.1.1:

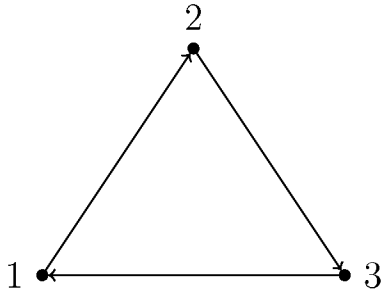

Рис. 2.1.1. Турнір із 3 вершинами й 3 королями: 1, 2, 3.

Для $n = 4$ матимемо турнір, що зображено на рис. 2.1.2:

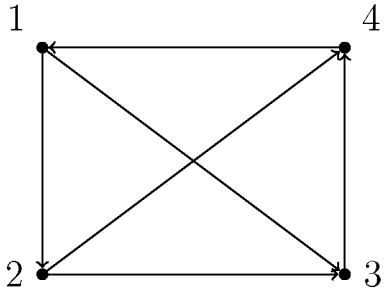

Рис. 2.1.2. Турнір із 4 вершинами й 3 королями: 1, 2, 4.

Для $n = 5$ матимемо турнір, що зображено на рис. 2.1.3:

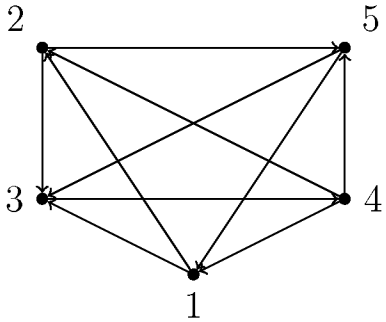

Рис. 2.1.3. Турнір із 5 вершинами й 5 королями: 1, 2, 3, 4, 5.

Для $n = 6$ матимемо турнір, що зображено на рис. 2.1.4:

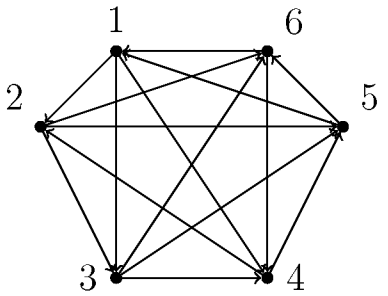

Рис. 2.1.4. Турнір із 6 вершинами й 6 королями: 1, 2, 3, 4, 5, 6.

Для побудови турніра T з 6-ма вершинами, візьмемо турнір з 5-ма вершинами і королями, додамо до нього 6-ту вершину й з'єднаємо її за наступним правилом:

- якщо n парне, то, при додаванні вершини $n+1$, степені протилежних вершин мають бути врівноваженими. Тобто, якщо для вершин $v, u \in T$ $deg^-(v) < deg^+(u)$, то проводимо $arc(n+1, v)$ і $arc(u, n+1)$ або, якщо $deg^-(u) < deg^+(v)$, то проводимо $arc(n+1, u) i arc(v, n+1)$. Таким чином ми з'єднаємо вершину $n+1$ з іншими та зробимо її королем.

- якщо n непарне, то, при додаванні вершини $n+1$, для кожної вершини $v \in V$, окрім $n+1$ треба врівноважити $deg^-(v)$ і $deg^+(v)$. Якщо $deg^-(v) < deg^+(v)$, проводимо $arc(n+1, v)$ або, якщо $deg^-(v) > deg^+(v)$, проводимо $arc(v, n+1)$. Таким чином ми додамо до турніру з n вершин вершину $n+1$, з'єднаємо її з іншими та зробимо її королем.

Для побудови турніра T з 7-ма вершинами, візьмемо турнір із 6-ма вершинами й королями й додамо до нього 7, з'єднуючи її з іншими за правилом описаним вище. Для побудови турніра T з 8-ма вершинами, візьмемо турнір із 7-ма вершинами й королями, додамо до нього 8-му вершину, яку з'єднаємо за правилом вище.

Для $n = 7$ матимемо турнір, що зображено на рис. 2.1.5:

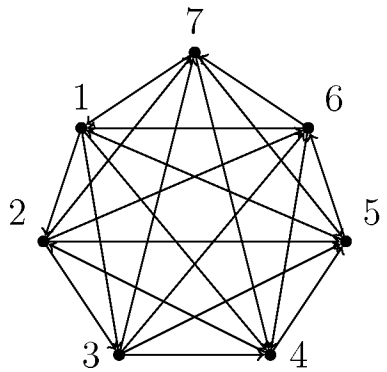

Рис. 2.1.5. Турнір із 7 вершинами й 7 королями: 1, 2, 3, 4, 5, 6, 7.

Для $n = 8$ матимемо турнір, що зображено на рис. 2.1.6:

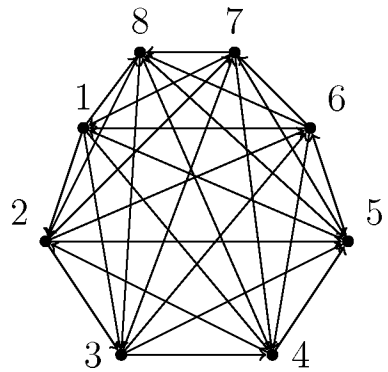

Рис. 2.1.6. Турнір із 8 вершинами й 8 королями: 1, 2, 3, 4, 5, 6, 7, 8.

□

Теорема 2.2. [1, p. 65] *Кожний турнір T має короля. Нехай, T - турнір, у якого для кожної вершини $v \in V$ виконується $deg^-(v) = 0$.*

a) Якщо x є королем в T, тоді T має іншого короля в $N^-(x)$.

b) Використовуючи, частину a), турнір T має принаймні три королі.

c) Для кожної $n \geq 3$, побудуйте турнір T з $\delta^-(T) > 0$ і має тільки 3 королі. (Коментар: На n вершинах існує турнір, що має рівно k королів, коли $n \geq k \geq 1$, за винятком випадків, коли $k = 2$ і коли $n = k = 4$)

Доведення.

a) Нехай у $N^-(x)$ є король v. Тоді, v - вершина з найбільшим deg^+. Припустимо, що в $N^-(x)$ існує така вершина w. Тоді, якщо v не доходить за менше, ніж 2 кроки в w, то $N^+(v) \cup \{v\} \subset N^+(w)$. Отримали суперечність $N^+(v) < N^+(w)$, у той час як вершина v є вершиною з найбілим степенем. Отже, в $N^-(x)$ існує інший король.

b) Використовуючи a) маємо, що в $N^-(x)$ існує інший король, назвемо його y. Тоді, використовуючи a), маємо іншого короля в $N^-(y)$, назвемо його z. Очевидно, що $z \neq x$, так як z має шлях довжиною 1 або 2 у вершину y, у той час як y має шлях довжиною 1 або 2 у вершину x.

c) Побудуємо турніри для $n \geq 3$.

Для $n = 3$ матимемо турнір, що зображено на рис. 2.1.7:

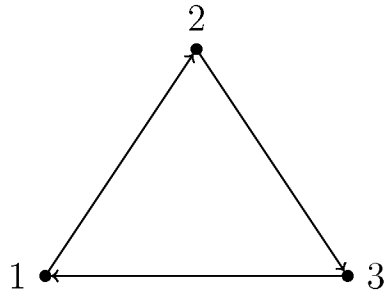

Рис. 2.1.7. Турнір із 3 вершинами й 3 королями: 1, 2, 3.

Для $n = 4$ матимемо турнір, що зображено на рис. 2.1.8:

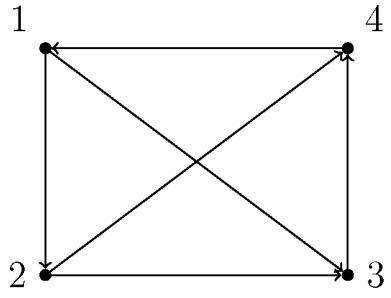

Рис. 2.1.8. Турнір із 4 вершинами й 3 королями: 1, 2, 4.

Для $n = 5$ матимемо турнір, що зображено на рис. 2.1.9:

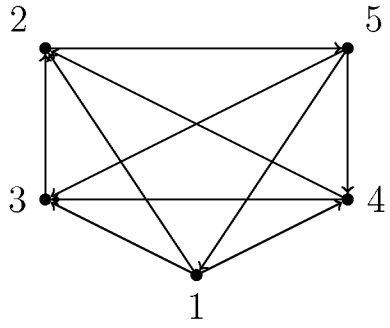

Рис. 2.1.9. Турнір із 5 вершинами й 3 королями: 1, 2, 5.

Для $n = 6$ матимемо турнір, що зображено на рис. 2.1.10:

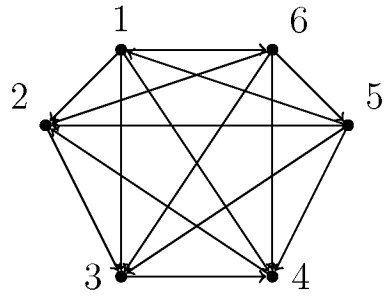

Рис. 2.1.10. Турнір із 6 вершинами й 3 королями: 1, 5, 6.

□

2.2 Сильні королі в турнірах T

Теорема 2.3. [3] *Кожна вершина в турнірі з найбільшим рахунком є сильним королем.*

Доведення. Нехай V - набір вершин для деякого турніра T, E - набір орієнтованих ребер для T. Припустимо, що існує вершина $u \in V$ з найбільшим рахунком, і яка не є сильним королем. Тоді існує вершина $v \in V$ така, що $v \to u$, тобто існує $(v, u) \in E$ і $b(u, v) \leq b(v, u)$.
Визначимо: $W = \{w \in V \backslash \{u, v\} : u \to w, v \to w\}$.
Тобто W - це множина всіх вершин w, до яких проведені ребра як з вершини u, так і з вершин v. Маємо, що $deg^+(u) = b(u, v) + |W|$ і $deg^+(v) = b(u, v) + |W| + 1$.
Тобто, $deg^+(u) < deg^+(v)$, що суперечить умові задачі. □

2.3 Королі в тензорному добутку орієнтованих графів $G_1 \times G_2$

Теорема 2.4. [4, p. 600] *Якщо орієнтовані графи $G_1, G_2, ..., G_r$ - сильнозв'язні орієнтовані графи, то тензорний добуток $G_1 \times G_2 \times ... \times G_r$ матиме рівно*

$$\frac{d(G_1) \cdot d(G_2) \cdot ... \cdot d(G_r)}{lcm(d(G_1), d(G_2), ..., d(G_r))}$$

сильних компонент.
Примітка: *lcm (з. англ. least common multiple) - найменше спільне кратне.*

Теорема 2.5. [5, p. 17] *Якщо D_1 і D_2 є сильнозв'язними орграфами, то $D_1 \times D_2$ має $gcd(d(D_1), d(D_2))$ сильних компонент. Зокрема, $D_1 \times D_2$ є сильнозв'язаним орграфом тоді й тільки тоді, коли $gcd(d(D_1), d(D_2)) = 1$. Тобто, за теоремою 2.4 $D_1 \times D_2$ матиме всього лише одну сильну компоненту.*

Примітка: *$d(D_1)$ було замінено на $g_{D_1}(v)$ так, як нам достатньо, щоб існувала сильна компонента з королем v.*

Теорема 2.6. [5, p. 23] *Вершина (v, w) є королем в $D_1 \times D_2$ тоді й тільки тоді, коли вершина v є королем в D_1, вершина w є королем в D_2 і $gcd(g_{D_1}(v), g_{D_2}(w)) = 1$.*

Доведення. **Необхідність**. Нехай задані орієнтовані графи D_1 і D_2, а вершини $v \in D_1$ і $w \in D_2$ є королями у відповідних графах і $gcd(g_{D_1}(v), g_{D_2}(w)) = 1$.

1) Покажемо, що $v \in D_1$ і $w \in D_2$ є королями. Припустимо, що вершина v не є королем в D_1. У такому випадку в орграфі D_1 існуватиме така вершина u, що з v не існуватиме шляху довжиною 1 або 2 у вершину u. Нехай у графі D_2 існує вершина w' і вершина w є королем в D_2. Тоді вона матиме шлях довжиною 1 або 2 у вершину w'. Тоді, в графі $D_1 \times D_2$ вершина (v, w) не матиме шляху у вершину (u, w'), тобто не буде королем, що суперечить умові. Отже, вершина v є королем в D_1. Аналогічно доводимо необхідність того, щоб вершина w також була королем в D_2. Так як ми розглядаємо тензорний добуток графів D_1 і D_2, вершина, що є королем в $D_1 \times D_2$ буде саме $k -$, де $k = |D_1(V)| * |D_2(V)|$.

2) Тепер доведемо, що тензорний добуток $D_1 \times D_2$ має рівно 1 сильну компоненту. Припустимо, що $gcd(g_{D_1}(v), g_{D_2}(w)) = \infty$. У такому випадку це означатиме, що граф $D_1 \times D_2$ не матиме сильних компонент, а отже, вершина (v, w) не буде королем, так як вона не матиме шляху довжини k або менше у всі інші вершини, а це суперечить умові. Тепер припустимо, що $1 < gcd(g_{D_1}(v), g_{D_2}(w)) < \infty$. У такому випадку, граф $D_1 \times D_2$ матиме декілька сильних компонент, а отже, вершина (v, w) не обов'язково потрапить в усі інші вершини за k кроків. Тоді, (v, w) не буде королем, а це суперечить умові. Отже, $gcd(g_{D_1}(v), g_{D_2}(w)) = 1$.

Достатність. Нехай $(v, w) \in D_1 \times D_2$ є королем. Покладемо $v \in D_1$ і $w \in D_2$ є вершинами в орієнтованих графах D_1 і D_2. Тоді, за теоремами 2.4 і 2.5 тензорний добуток $D_1 \times D_2$ матиме рівно одну сильну компоненту й вершина (v, w) матиме шлях довжини k в будь-яку іншу вершину. Тоді, якщо розглянути ці вершини окремо, то v і w будуть королями в графах D_1 і D_2, так як матимуть шляхи довжини k в усі інші вершини. а сильна зв'язність $D_1 \times D_2$ забезпечує існування шляху довжини k з вершин v і w в інші вершини. □

3 Алгоритми

3.1 Необхідні функції

Спочатку визначимо функції. які знадобляться для виконання алгоритів.

1. Імпортуємо необхідні модулі Python.

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import random

```

2. Визначимо функцію, яка трансформує матрицю суміжності в словник (колекція у Python), який складається з вершин (ключів) та суміжних вершини (значення).

```
def adj_matrix_to_dict(adj_matrix):
 n = len(adj_matrix)
 adj_dict = {}
 for i in range(n):
  adj_dict[i+1] = []
  for j in range(n):
   if adj_matrix[i][j] == 1:
    adj_dict[i+1].append(j+1)
 return adj_dict

```

3. Визначимо функцію, яка перевірятиме чи граф G, що не є двочастковим, і який подали у вигляді суміжної матриці є турніром.

```
def is_basic_tournament(adj_matrix):
 n = len(adj_matrix)
 for i in range(n):
  for j in range(n):
   if i != j and adj_matrix[i][j] + adj_matrix[j][i] != 1:
    return False
 return True
```

4. Визначимо функцію, яка перевірятиме чи граф G, який подали у вигляді суміжної матриці є двочастквим турніром.

```
def is_bipirtite_tournament(adj_matrix):
 n = len(adj_matrix)
 identity_matrix = [[int(i == j) for j in range(n)] for i in range(n)]
 res = True
 for i in range(n):
  for j in range(i+1, n):
   if identity_matrix[i][j] != 0 and identity_matrix[j][i] != 0:
    res = False
    break
```

```
  if not res:
   break
 return res
```

5. Визначимо функцію, яка приймає суміжну матрицю, що є представленням турніру T, і булеан $True or False$, що позначає чи треба перевіряти турнір T, як двочастковий (значення $True$) чи як звичайний турнір (значення $False$).

```
def is_tournament(adj_matrix, is_bipartite):
 if is_bipartite == True:
  return is_bipirtite_tournament(adj_matrix)
 else:
  return is_basic_tournament(adj_matrix)

```

6. Визначимо функцію, яка обраховуватиме матрицю $D^{+(degree)}$, де D^+ і $degree$ - це параметри функції, які позначають матрицю суміжності турніра T і степінь до якого її треба піднести. Функція повертає $D^{+(degree)}$ - матрицю з кількістю елементарних шляхів довжиною, що рівна $degree$ (степеню) для $\forall v \in V$.

```
def count_D_plus_with_next_degree(adj_matrix, degree):
 n = len(adj_matrix)
 D_plus_degree = adj_matrix
 for p in range(degree-1):
  D_plus_degree = D_plus_degree @ adj_matrix
 return D_plus_degree

```

7. Визначимо функцію, яка обраховуватиме матрицю $M = I + D^+ + D^{+2} + ... + D^{+(degree-1)} + D^{+(degree)}$, де I - це одинична матриця, D^+ - це матриця з кількістю елементарних шляхів довжиною 1 для $\forall v \in V$, D^{+2} - це матриця з кількістю елементарних шляхів довжиною 2 для $\forall v \in V$, а $degree$ позначає кількість кроків, яка визначається для обчислення матриці з кількістю елементарних шляхів із заданою довжною $degree$.

```
def count_M(adj_matrix, t_as_dict, max_degree):
 D_plus = adj_matrix
 n = len(D_plus)
 I = np.identity(n)
 M = [[I[i][j] + D_plus[i][j] for j in range(n)] for i in range(n)]
 degree = 2
 while degree <= max_degree:
  D_plus_deg = count_D_plus_with_next_degree(D_plus, degree)
  for i in range(n):
   for j in range(n):
    M[i][j] += D_plus_deg[i][j]
  degree += 1
 return M

```

8. Визначимо функцію, що прийматиме граф заданий суміжною матрицею або класом DiGraph з бібілотеки networkx, що імпортуємо в Python. Також функція прийматиме набір королів і булеан, що вказує чи граф заданий матрицею суміжності (значення False) чи класом DiGraph (значення True). У залежності від значення булеана функція виклатиме функцію для малювання графа заданого суміжною матрицею або класом DiGraph.

```
def draw_graph_with_kings(graph, kings, isDiGraph):
 if isDiGraph == False:
  draw_adj_matrix(graph, kings)
 else:
  draw_DiGraph(graph, kings)

```

9. Визначимо функцію, що малюватиме турнір T, заданий матрицею суміжності, і відображатиме королів аквамариновим кольором.

```
def draw_adj_matrix(graph, kings):
 G = nx.DiGraph(graph)
 pos = nx.spring_layout(G)
 nx.draw_networkx_nodes(G, pos, nodelist=kings, node_color='violet')
 nx.draw_networkx_nodes(G, pos, nodelist=set(graph.keys())-set(kings),
   node_color='aquamarine')
 nx.draw_networkx_edges(G, pos)
 nx.draw_networkx_labels(G, pos)
 plt.title("Tournament T")
 plt.axis('off')
 plt.show()

```

10. Визначимо функцію, що малюватиме турнір T, заданий класом DiGraph, і відображатиме королів аквамариновим кольором.

```
def draw_DiGraph(T, K):
 pos = nx.circular_layout(T)
 nx.draw_networkx_nodes(T, pos, nodelist=T.nodes() - K, node_color='
   aquamarine', node_size=500)
 nx.draw_networkx_nodes(T, pos, nodelist=K, node_color='violet', node_size
   =500)
 nx.draw_networkx_edges(T, pos)
 nx.draw_networkx_labels(T, pos)
 plt.axis('off')
 plt.show()
```

3.2 Алгоритми на турнірах T

1. Алгоритм пошуку короля x в турнірі T [1, p. 65]
   Опис алгоритму: алгоритм приймає на вхід турнір T, обирає будь-яку вершину x із T і перевіряє чи є вона королем. Якщо обрана вершина x - король, то алгоритм повертає її і закінчує свою дію. Якщо ж, x не є королем, то ця вершина видаляється разом із множино сусідів $N^+(x)$. Цей процес повторюється поки алгоритм не поверне вершину x, що є королем.

   Код Python:

```python
def get_indegrees(graph):
 indegrees = {v: 0 for v in graph.keys()}
 for neighbors in graph.values():
  for neighbor in neighbors:
   indegrees[neighbor] += 1
 return indegrees

def find_king(G, T):
 indegrees = get_indegrees(T)
 for x in T:
  if indegrees[x] == 0:
   return x
 x = next(iter(T))
 N = {y for y in T if x in G[y]}
 T_prime = T - {x} - N
 find_king(G, T_prime)

graph = np.array([[0, 1, 1, 1, 1], [0, 0, 1, 1, 1], [0, 0, 0, 1, 0], [0, 0,
    0, 0, 0], [0, 0, 1, 1, 0]])
 if is_tournament(graph):
  king = []
  T = adj_matrix_to_dict(graph)
  G = nx.DiGraph(T)
  king.append(find_king(G, T))
  print("The king is:", king)
  draw_graph_with_kings(T, king)
 else:
  print("A matrix given is not a tournament!")

```

Задамо граф матрицею суміжності $\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$ і викличемо функцію.

Отримаємо результат як на рис. 3.2.1:

```
The king is: [1]
```

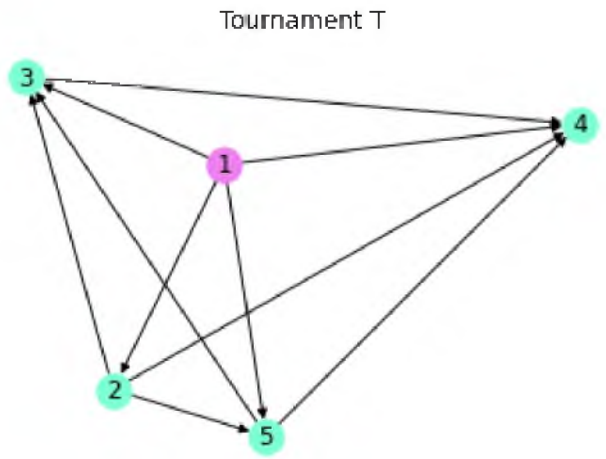

Рис. 3.2.1. Турнір із 5 вершинами й 1 королем: 1.

2. Алгоритм пошуку всіх королів у турнірі T [3]
Опис алгоритму: Опис алгоритму: Спочатку обраховується матриця інцидентності D^+, далі для кожної вершини $i \in V$ обраховується матриця M з кількістю елементарних шляхів довжиною 1 або 2 в інші вершини. Наступний кроком треба перебрати всі вершини $i \in V$ і, якщо для вершини i $\forall j \in V : M_{ij} \neq 0$, додаємо вершину i до списку королів. Після перебору й перевірки всіх вершин, повертаємо список із королями.

Псевдокод:

1) D^+
2) $K = 0$
3) for $i = 1$ to n
4) for $j = 1$ to n
5) if $(v_i, v_j) \in A$ then $(D^+)_{ij} = 1$
6) $M = I + D^+ + (D^+)^2$
7) $K = \{v_i \in V | \forall v_j \in V, (M)_{ij} \neq 0\}$
8) return K

Код Python:

```
def find_kings(graph, T):
 M = count_M(graph, T)
 kings = []
 for v in T.keys():
  counter = 0
  for i in range(len(M[0])):
   if M[v-1][i] == 0:
    counter += 1
  if counter == 0:
   kings.append(v)
 return kings

graph = np.array([[0, 1, 1, 0, 0], [0, 0, 0, 1, 0], [0, 1, 0, 0, 1], [1, 0,
    1, 0, 1], [1, 1, 0, 0, 0]])
 if is_tournament(graph):
 T = adj_matrix_to_dict(graph)
 kings = find_kings(graph, T)
 print("The kings are:", kings)
 draw_graph_with_kings(T, kings)
else:
 print("A matrix given is not a tournament!")

```

Задамо граф матрицею суміжності $\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$ і викличемо функцію.

Отримаємо результат як на рис. 3.2.2:

```
The kings are: [1, 2, 3, 4, 5]
```

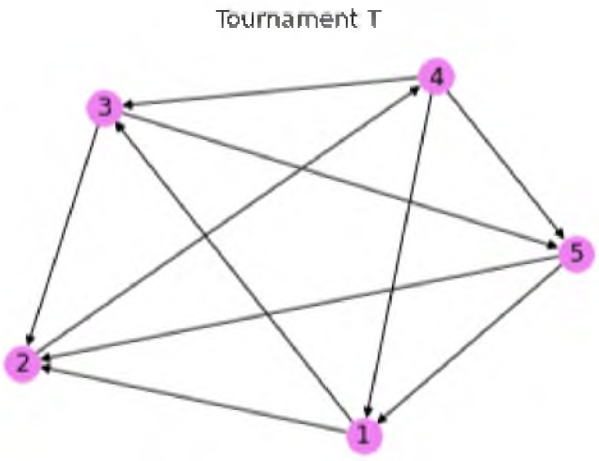

Рис. 3.2.2. Турнір із 5 вершинами й 5 королями: 1, 2, 3, 4, 5.

3. Алгоритм пошуку сильних королів в турнірі T [3]
Опис алгоритму: першим кроком треба обрахувати матриця інцидентності D^+, після цього треба обрахувати матрицю D^{+2}. Далі для кожної вершини $i \in V$ обраховується матриця M, яка має одиниці на діагоналі й суму кількості елементарних шляхів довжиною 2 або менше з вершини i у вершину v в комірці iv . Наступний кроком треба перебрати всі вершини $i \in V$ і, якщо для вершини i $\forall j \in V : M_{ij} \neq 0$ й $(M)_{ij} > (M)_{ij}$, додаємо вершину i до списку сильних королів. Після перебору й перевірки всіх вершин, повертаємо список із сильними королями.

Псевдокод:

1) D^+
2) $K = 0$
3) for $i = 1$ to n
4) for $j = 1$ to n
5) if $(v_i, v_j) \in A$ then $(D^+)_{ij} = 1$
6) $M = I + D^+ + (D^+)^2$
7) $K = \{v_i \in V | \forall j (1 \leqslant j \leqslant n \, and \, j \neq i), (M)_{ij} > (M)_{ij}\}$
8) return K

Код Python:

```
def find_strong_kings(graph, T):
 M = count_M(graph, T)
 strong_kings = []
 for v in T.keys():
  counter = 0
   for u in T.keys():
    if M[v-1][u-1] < M[u-1][v-1]:
     counter += 1
   if counter < 1:
    strong_kings.append(v)
 return strong_kings

graph = np.array([[0, 1, 1, 0, 0], [0, 0, 0, 1, 0], [0, 1, 0, 0, 1], [1, 0,
    1, 0, 1], [1, 1, 0, 0, 0]])
 if is_tournament(graph):
  T = adj_matrix_to_dict(graph)
  strong_kings = find_strong_kings(graph, T) # Find the kings
  print("The strong_kings are:", strong_kings)
  draw_graph_with_kings(T, strong_kings)
 else:
  print("A matrix given is not a tournament!")

```

Задамо граф матрицею суміжності $\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$ і викличемо функцію.

Отримаємо результат як на рис. 3.2.3:

```
The strong_kings are: [4]
```

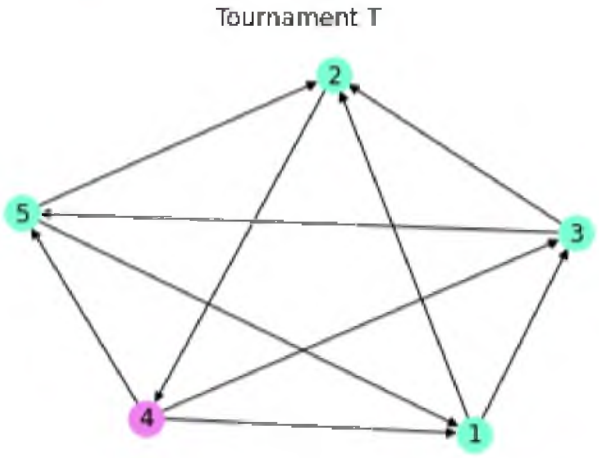

Рис. 3.2.3. Турнір із 5 вершинами й 1 сильним королем: 4.

4. Алгоритм побудови турнірів на n вершинах і з k сильними королями. [3]
Опис алгоритму: Спочатку перевіряємо, щоб $n \geq k \geq 1$ і $k \neq n-1$, коли n непарне, і $k \neq n$, коли n парне. наступним кроком ініціалізуємо K - набір сильних королів і T - орієнтований граф без ребер на n вершинах. Наступним кроком визначаємо чи є k парним. Якщо k - непарне, присвоюємо T значення регулярного турніра на k вершинах, додаємо в набір K вершини з 1-ої по k-ту. Наступним кроком, якщо $n \neq k$, додаємо до графа T всі вершини з $k+1$-ої по n і проводимо ребра з всіх вершин до останньої доданої. Якщо k - парне, присвоюємо T значення регулярного турніра на $k-1$ вершинах, додаємо до графа T вершини x, y, z і додаємо в набір K вершини з 1-ої по $v-1$-у і вершину x. Наступним кроком обираємо випадкову вершину $u \in V$, ініціалізуємо вершини v_1 - перша вершина в $deg^-(u)$ і v_2 - перша вершина в $deg^+(u)$. Після цього проводимо ребра $arc(x, v_1)$, $arc(x, u)$, $arc(v_2, x)$, $arc(v_1, y)$, $arc(u, y)$, $arc(v_2, y)$, $arc(v_1, z)$, $arc(u, z)$, $arc(z, v_2)$. Наступним кроком, якщо $n \neq k+2$, додаємо до графа T всі вершини з $k+1$-ої по n і проводимо ребра з всіх вершин до останньої доданої. У кінці повертаємо турнір T та набір сильних королів K.

Псевдокод:

1) $K = 0$
2) $T =$ null digraph on n vertexes
3) if k is odd
4) $T = T_k$
5) $K = \{v_1, ..., v_k\}$
6) if $n \neq k$
7) for $i = k + 1$ to n
8) $V = V \cup \{v_i\}$
9) $A = A \cup \{(u, v_i) : u \in V - \{v_i\}\}$
10) if k is even
11) $T = T_{k-1}$
12) $V = V \cup \{x, y, z\}$
13) $K = v_1, ..., v_{k-1}, x$
14) choose $u \in V$
15) $A = A \cup \{(v, x) : v \in O(u)\}$
16) $A = A \cup \{(x, v) : v \in \{u, y\} \cup I(u)\}$
17) $A = A \cup \{(v, y) : v \in \{u\} \cup I(u) \cup O(u)\}$
18) $A = A \cup \{(v, z) : v \in \{u\} \cup I(u)\}$
19) $A = A \cup \{(z, v) : v \in O(u)\}$
20) if $n \neq k + 2$
21) for $i = k + 1$ to n
22) $V = V \cup \{v_i\}$
23) $A = A \cup \{(u, v_i) : u \in V - \{v_i\}\}$
24) return T, K

Код Python:

```
def generate_tournament(n, k):
 if check_n_k(n,k) == True:
 K = set()
 if k % 2 == 1:
  T = generate_regular_tournament(k)
  for i in range(k):
   K.add(i + 1)
   if n != k:
    for v in range(k + 1, n + 1):
     T.add_node(v)
     for u in range(1, n):
      if u != v:
       if not T.has_edge(v, u):
        T.add_edge(u, v)
 else:
  T = generate_regular_tournament(k - 1)
  x, y, z = k, k + 1, k + 2
  for i in range(k - 1):
   K.add(i + 1)
   K.add(x)
   T.add_node(x)
   T.add_node(y)
   T.add_node(z)
   T.add_edge(x, z)
   T.add_edge(y, x)
   T.add_edge(y, z)
   u = random.choice(list(T.nodes()))
   while u == x or u == y or u == z:
    u = random.choice(list(T.nodes()))
   v1 = find_first_predecessor(T, u, x, y, z)
   v2 = find_first_successor(T, u, x, y, z)
   T.add_edge(x, v1)
   T.add_edge(x, u)
   T.add_edge(v2, x)
   T.add_edge(v1, y)
   T.add_edge(u, y)
   T.add_edge(v2, y)
   T.add_edge(v1, z)
   T.add_edge(u, z)
   T.add_edge(z, v2)
   if n != k + 2:
    for v in range(k + 1, n + 1):
     T.add_node(v)
     for u in range(1, n):
      if u != v:
       if not T.has_edge(v, u):
        T.add_edge(u, v)
 print("Strong kings are: ", K)
 draw_graph_with_kings(T, K, True)
 return T, K

def check_n_k(n, k):
 conditions_satisfied = True
 if n < k:
  conditions_satisfied = False
```

```
  print("n is less than k!")
 elif n < 1 and k < 1:
  conditions_satisfied = False
  print("n and k are less than 1!")
 elif n >= 1 and k < 1:
  conditions_satisfied = False
  print("k is less than 1!")
 else:
  if n % 2 == 0:
   if k == n:
    conditions_satisfied = False
    print("It is impossible to build a tournament with", n, "vertexes and",
   k, "kings, because k = n")
  if n % 2 == 1:
   if k == n - 1:
    conditions_satisfied = False
    print("It is impossible to build a tournament with", n, "vertexes and",
   k, "kings, because k = n - 1")
 return conditions_satisfied

def generate_regular_tournament(k):
 T = nx.DiGraph()
 score = int (k/2)
 vertices = list(range(1, k+1))
 for i in vertices:
  for v in range(1, score + 1):
   j = (i + v) % k
   if j != 0:
    T.add_edge(i, j)
   else:
    T.add_edge(i, k)
 return T

def find_first_predecessor(T, u, x, y, z):
 predecessors = list(T.predecessors(u))
 if not predecessors:
  return None
 else:
  for i in predecessors:
   if i != x and i != y and i != z:
 return i

def find_first_successor(T, u, x, y, z):
 successors = list(T.successors(u))
 if not successors:
  return None
 else:
 for i in successors:
  if i != x and i != y and i != z:
 return i

generate_tournament(6,3)

```

Викличемо функцію із значеннями $n = 7$ і $k = 3$.
Отримаємо результат як на рис. 3.2.4:

```
Strong kings are:  {1, 2, 3}
```

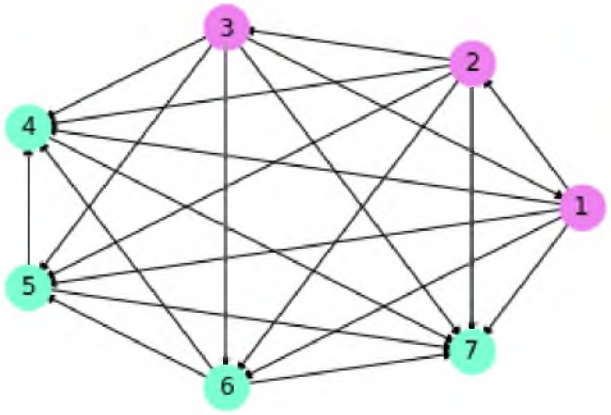

Рис. 3.2.4. Турнір із 7 вершинами й 3 королями: 1, 2, 3.

Змінимо $k = 4$ і викличемо функцію ще раз.
Отримаємо результат як на рис. 3.2.5:

```
Strong kings are:  {1, 2, 3, 4}
```

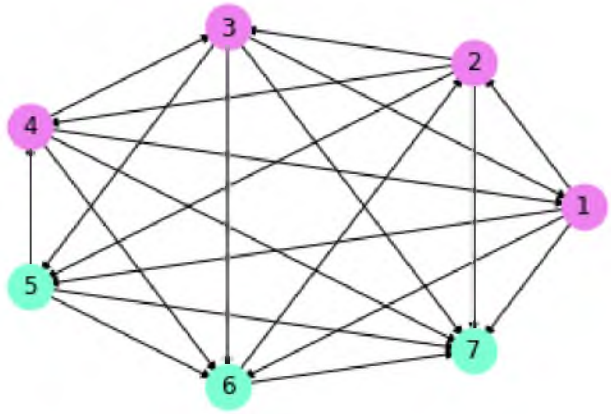

Рис. 3.2.5. Турнір із 7 вершинами й 4 королями: 1, 2, 3, 4.

5. Алгоритм пошуку королів у двочастковому турнірі T. [3]
Опис алгоритму: першим кроком треба обрахувати матриця інцидентності D^+, після цього треба обрахувати матрицю D^{+2}, D^{+3}, D^{+4}. Далі для кожної вершини $i \in V$ обраховується матриця M, яка має одиниці на діагоналі й суму кількості елементарних шляхів довжиною 4 або менше з вершини i у вершину v в комірці iv . Наступний кроком треба перебрати всі вершини $i \in V$ і, якщо для вершини i $\forall j \in V : M_{ij} \neq 0$, додаємо вершину i до списку королів. Після перебору й перевірки всіх вершин, повертаємо список із королями.

Псевдокод:

1) D^+
2) $K = 0$
3) for $i = 1$ to n
4) for $j = 1$ to n
5) if $(v_i, v_j) \in A$ then $(D^+)_{ij} = 1$
6) $M = I + D^+ + (D^+)^2 + (D^+)^3 + (D^+)^4$
7) $K = \{v_i \in V | \forall v_j \in V, (M)_{ij} \neq 0\}$
8) return K

Код Python:

```
def find_kings_in_bipartite(graph, T):
 max_degree = 4
 M = count_M(graph, T, max_degree)
 kings = []
 for v in T.keys():
  counter = 0
  for i in range(len(M[0])):
   if M[v-1][i] == 0:
    counter += 1
  if counter == 0:
   kings.append(v)
 return kings

graph = np.array([[0, 1, 0, 0, 0], [0, 0, 1, 0, 1], [0, 0, 0, 1, 0], [1, 0,
   0, 0, 1], [0, 0, 0, 0, 0]])
is_bipartite = True
if is_tournament(graph, is_bipartite):
 T = adj_matrix_to_dict(graph)
 kings = find_kings_in_bipartite(graph, T) # Find the kings
 print("The kings are:", kings)
 draw_graph_with_kings(T, kings)
else:
 print("A matrix given is not a bipartite tournament!")

```

Задамо граф матрицею суміжності $\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ і викличемо функцію.

Отримаємо результат як на рис. 3.2.6:

```
The kings are: [1, 2, 3, 4]
```

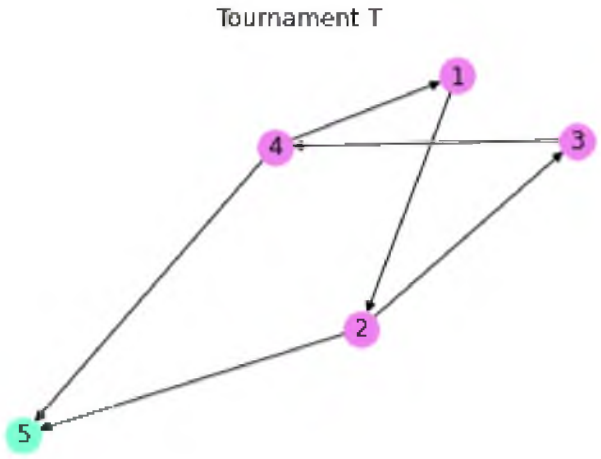

Рис. 3.2.6. Двочастковий турнір із 5 вершинами й 4 королями: 1, 2, 3, 4.

Висновки

У роботі було досліджено королів у турнірах і створені алгоритми мовою програмування Python. Було доведено різноманітні теореми та створено додаткові функції задля роботи алгоритмів.

У **Розділі 1** були наведені основні поняття з теорії графів, а також вводяться поняття турнірів, королів, сильних королів, тензорного добутку і додаткові поняття, як-от найбільший спільний дільник елементарних шляхів.

У **Розділі 2** було наведено й доведено основні теореми. У **Підрозділі 2.1** наводяться доведення теорем про існування королів у турнірах, їх кількість і можливість побудови турнірів на n вершинах із k королями. У **Підрозділі 2.2** була доведена теорема, що описує основну властивість сильного короля. У **Підрозділі 2.3** було розглянуто королів у тензорному добутку орієнтованих графів $G_1 \times G_2$.

У **Розділі 3** були наведені функції та алгоритми пошуку королів у турнірах і побудови турнірів. У **Підрозділі 3.1** описано необхідні функції, які перевіряють правильність введених даних, створюють зображення турнірів та виконують додаткові обчислення. У **Підрозділі 3.2** були наведенр й реалізовано алгоритми пошуку королів, сильних королів у звичайних і двочасткових турнірах, а також побудови турнірів на n вершинах із k королями, де n і k - задані числа.

Отже, у цій курсовій роботі досліджено основні поняття з теми "Королі в турнірах"і було реалізовано численні алгоритми, що допомагають наочно знайти королів чи побудувати турніри.

Література

[1] Douglas B. West, *Introduction in Graph Theory*, Pearson Education Inc., 2001, 589 p.

[2] J.W. Moon, *Topics on tournaments*, Holt, Rinehart and Winston Inc., 1968, 137 p.

[3] *Kings and serfs in tournaments*, [Електронний ресурс] https://gyires.inf.unideb.hu/KMITT/a53/ch07s04.html.

[4] M. H. McAndrew., *On the Product of Directed Graphs*, Proceedings of the American Mathematical Society Volume 14, 1963, 606 p.

[5] Morgan Norge, *Kings in the Direct Product of Digraphs*, Virginia Commonwealth University, 2019, 32 p.