**Лабораторна робота №3**

**Тема: Представлення дерева кодом Прюфера.**

**Теоретичнi вiдомостi.**

Усi загальнi способи представлення графiв є дiйсними й у випадку дерев. Проте дерево є особливим графом, i для нього можливi також iншi способи представлення.

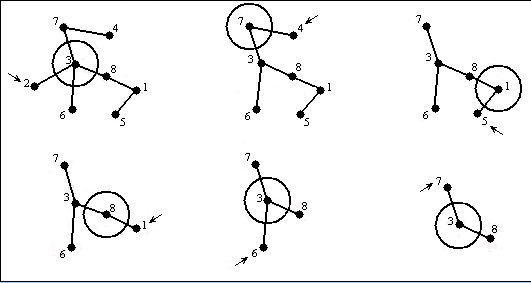
У 1918 роцi Прюфер запропонував метод представлення дерева з *n > 2* вершинами, який тепер називають **кодом Прюфера**. Алгоритм укладання коду Прюфера є таким:

|  |
| --- |
| PRUF{G,Code)   Перенумерувати вершини дерева **G** довiльним чином.   **while** (у деревi не залишилось одне ребро)     Знайти висячу вершину з найменшим номером.     Видалити знайдену вершину разом з iнцидентним ребром.     Занести номер вершини, сумiжної видаленiй, до коду **Code**.   **end** {while} end {PRUF} |

Пiсля укладання коду в деревi завжди залишаються двi вершини, тому довжина коду Прюфера для будь-якого дерева буде на два менше числа вершин дерева: **К = n - 2**. Отже, маючи код, можна визначити кiлькiсть вершин у графi: їх завжди буде на двi бiльше кiлькостi елементiв коду.

Номери вершин у кодi можуть повторюватися, а у випадку зiркового графа вони взагалi будуть однаковими.

На наступному рисунку показаний процес укладання коду Прюфера для дерева.



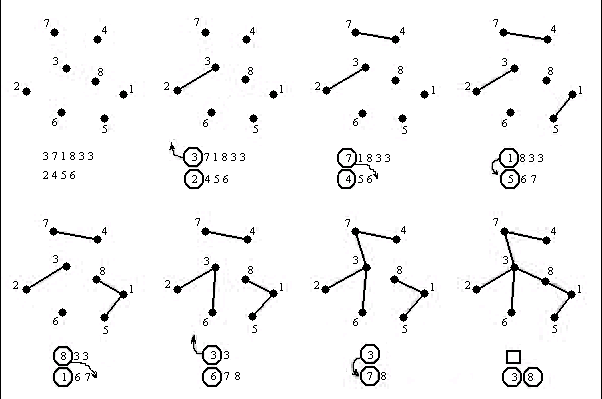
Результатом роботи алгоритму є код **К = 3 7 1 8 3 3**.

Вiдновлення дерева з коду виконується за допомогою антикоду. Антикод - це впорядкований кортеж, складений з номерiв вершин, що не увiйшли до коду. Для розглянутого прикладу антикод **A = 2 4 5 6**.

Алгоритм вiдновлення дерева за кодом Прюфера показаний нижче:

|  |
| --- |
| TREE(Code, Tree)   Встановити кiлькiсть вершин дерева: **n - Length(Code) + 2**.   Отримати антикод **AntiCode**.   Створити **n** вершин i дати їм номери.   **while** (Code > 0)     Витягнути першi елементи **u** i **v** з коду й антикоду.     Вiдновити ребро **(u, v)** в деревi.     Видалити елемент **v** за антикоду.     if (елемент **u** ще зустрiчається далi в кодi)       then         Витягнутий елемент коду **u** вiдкинути       else         if (елемент **u** менший за всi елементи антикоду)           then             Поставити елемент на початок антикоду           else             Витягнутий елемент коду **u** додати в кiнець антикоду.           end {if}       end {if)   **end** {while}   З'єднати ребром останнi двi вершини, що залишились в антикодi. end {TREE} |

Робота алгоритму вiдновлення дерева показана на рисунку.



**Завдання:**

1. За кодом Прюфера згенерувати антикод та вiдновити граф.
2. Побудувати його матрицi сумiжностi вершин та iнцидентностi.
3. Малюнок графа.

**Примiтка:** *Нумерацiя вершин починається з 0.*

*Граф має 21 вершину.*

**Iндивiдуальнi завдання:**

1. 2, 2, 6, 6, 2, 3, 10, 10, 10, 3, 4, 5, 15, 15, 6, 5, 17, 17, 17.
2. 12, 12, 2, 13, 13, 14, 14, 19, 15, 15, 18, 16, 16, 17, 19, 18, 20, 20, 17.
3. 2, 2, 1, 5, 5, 1, 0, 9, 9, 8, 12, 12, 8, 0, 15, 16, 16, 15, 19.
4. 3, 3, 3, 4, 5, 5, 5, 4, 10, 10, 10, 4, 13, 13, 15, 13, 17, 17, 17.
5. 1, 1, 2, 5, 12, 8, 12, 8, 2, 3, 4, 16, 16, 9, 15, 9, 4, 5, 6.
6. 3, 6, 7, 6, 7, 6, 8, 12, 12, 14, 14, 12, 8, 3, 1, 1, 3, 2, 2.
7. 4, 10, 6, 6, 0, 1, 7, 7, 1, 2, 8, 8, 2, 3, 9, 9, 3, 4, 10.
8. 1, 5, 5, 4, 8, 8, 4, 3, 11, 11, 3, 2, 14, 14, 2, 1, 2, 18, 18.
9. 1, 2, 7, 6, 10, 10, 2, 3, 4, 14, 14, 6, 5, 4, 12, 13, 17, 13, 18.
10. 2, 2, 2, 4, 8, 8, 8, 4, 12, 12, 12, 4, 19, 19, 19, 20, 20, 20, 4.
11. 3, 3, 4, 4, 7, 3, 7, 12, 13, 11, 13, 16, 11, 12, 16, 17, 16, 12, 7.
12. 4, 3, 3, 4, 8, 7, 7, 8, 9, 8, 12, 12, 13, 13, 12, 18, 17, 17, 18.
13. 5, 5, 2, 6, 5, 6, 8, 8, 6, 9, 11, 12, 13, 14, 17, 14, 14, 13, 17.
14. 4, 5, 6, 4, 5, 9, 10, 8, 11, 10, 14, 14, 8, 9, 15, 15, 9, 10, 16.
15. 2, 2, 6, 7, 5, 7, 6, 5, 6, 10, 11, 14, 13, 14, 11, 10, 13, 17, 17.
16. 1, 10, 7, 7, 7, 7, 8, 10, 8, 12, 8, 8, 10, 12, 12, 12, 10, 13, 13.
17. 1, 5, 1, 2, 3, 4, 4, 5, 5, 5, 4, 3, 19, 20, 20, 20, 20, 19, 19.
18. 3, 3, 3, 6, 6, 3, 4, 4, 4, 5, 6, 16, 16, 16, 16, 17, 17, 17, 17.
19. 1, 6, 1, 1, 2, 20, 20, 20, 20, 2, 3, 4, 5, 6, 16, 16, 16, 16, 6.
20. 1, 4, 4, 6, 6, 4, 15, 15, 1, 1, 1, 2, 4, 2, 14, 14, 14, 14, 17.
21. 2, 2, 3, 6, 5, 4, 3, 8, 12, 11, 10, 9, 16, 15, 14, 8, 9, 18, 18.
22. 1, 3, 1, 2, 6, 6, 2, 3, 8, 11, 11, 7, 12, 12, 7, 8, 13, 19, 19.
23. 1, 3, 1, 2, 6, 6, 3, 10, 11, 13, 10, 11, 11, 17, 17, 11, 2, 3, 13.
24. 1, 2, 3, 6, 6, 2, 3, 3, 2, 15, 15, 15, 15, 15, 2, 16, 16, 16, 16.
25. 1, 5, 4, 8, 3, 3, 4, 3, 2, 12, 1, 8, 2, 15, 15, 15, 18, 15, 18.
26. 1, 4, 1, 7, 7, 3, 4, 1, 1, 2, 4, 3, 2, 13, 13, 2, 14, 14, 14.
27. 1, 6, 1, 2, 3, 4, 4, 6, 5, 2, 3, 3, 15, 5, 4, 3, 15, 15, 15.
28. 1, 3, 1, 3, 3, 1, 2, 10, 3, 2, 10, 10, 10, 15, 13, 13, 13, 10, 15.
29. 1, 3, 1, 2, 3, 6, 6, 6, 2, 12, 3, 3, 2, 12, 12, 12, 18, 12, 18.
30. 1, 3, 6, 1, 2, 7, 9, 3, 7, 12, 7, 2, 15, 15, 3, 3, 2, 15, 15.
31. 3, 3, 3, 5, 5, 5, 8, 8, 8, 11, 11, 11, 14, 14, 14, 17, 17, 17, 19.
32. 1, 2, 1, 1, 2, 7, 7, 7, 2, 12, 1, 2, 15, 12, 15, 15, 2, 1, 12.

<https://ru.coursera.org/learn/teoriya-grafov/lecture/oSPgU/postroieniie-kodov-priufiera>

## Алгоритм построения кодов Прюфера

Кодирование Прюфера переводит **помеченные деревья порядка** [n](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9A%D0%BE%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%D0%BF%D0%BE%D0%BC%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8B%D1%85_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D0%B5%D0%B2#.D0.9F.D0.BE.D0.BC.D0.B5.D1.87.D0.B5.D0.BD.D0.BD.D0.BE.D0.B5_.D0.B4.D0.B5.D1.80.D0.B5.D0.B2.D0.BE) в последовательность чисел от 1 до n по алгоритму:  
Пока количество вершин больше двух:

1. Выбирается лист v с минимальным номером.
2. В код Прюфера добавляется номер вершины, смежной с v.
3. Вершина v и инцидентное ей ребро удаляются из дерева.

Полученная последовательность называется **кодом Прюфера** *(англ. Codes Priifer)* для заданного дерева.

|  |
| --- |
| **Лемма**: |
| Номер вершины v встречается в коде Прюфера тогда и только тогда, когда v не является листом, причём встречается этот номер к коде дерева в точности \deg v - 1 раз. |
| **Доказательство:** |
| 1. Вершина с номером n не может быть удалена, следовательно на последнем шаге у неё была смежная вершина, и число nвстретилось в коде. 2. Если вершина не является листом, то у неё на некотором шаге была смежная вершина - лист, следовательно номер этой вершины встречается в коде. 3. Если вершина является листом с номером меньше n, то она была удалена до того, как был удален её сосед, следовательно её номер не встречается в коде.   Таким образом, номера всех вершин, не являющихся листьями или имеющих номер n, встречаются в коде Прюфера, а остальные - нет. |
| **Лемма**: |
| По любой последовательности длины n - 2 из чисел от 1 до n можно построить помеченное дерево, для которого эта последовательность является кодом Прюфера. |
| **Доказательство:** |
| Доказательство проведем по индукции по числу n *База индукции:*  n = 1 - верно.  *Индукционный переход:*  Пусть для числа n верно, построим доказательство для n+1:  Пусть у нас есть последовательность: A = [a_1, a_2, ..., a_{n - 2}].  Выберем минимальное число v не лежащее в A. По предыдущей лемме v - вершина, которую мы удалили первой. Соединим v и a_1 ребром. Выкинем из последовательности A число a_1. Перенумеруем вершины, для всех a_i > v заменим a_i на a_i - 1. А теперь мы можем применить предположение индукции. |
| **Теорема**: |
| Кодирование Прюфера задаёт биекцию между множествами помеченных деревьев порядка n и последовательностями длиной n - 2 из чисел от 1 до n |
| **Доказательство:** |
| 1. Каждому помеченному дереву приведенный алгоритм сопоставляет последовательность. 2. Каждой последовательности, как следует из предыдущей леммы, соотвествует помеченное дерево. |

Следствием из этой теоремы является **формула Кэли.**

|  |  |
| --- | --- |
| Пример построения кода Прюфера [Prufer.png](https://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%B0%D0%B9%D0%BB:Prufer.png) | Пример декодирования кода Прюфера [Backprufer.png](https://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%B0%D0%B9%D0%BB:Backprufer.png)  [Backprufer.png](https://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%B0%D0%B9%D0%BB:Backprufer.png) |