**ЛАБОРАТОРНАЯ РАБОТА.**

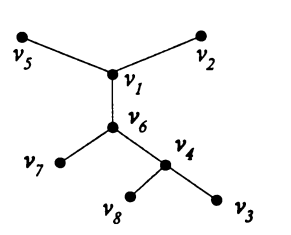
***Кодирование деревьев.***

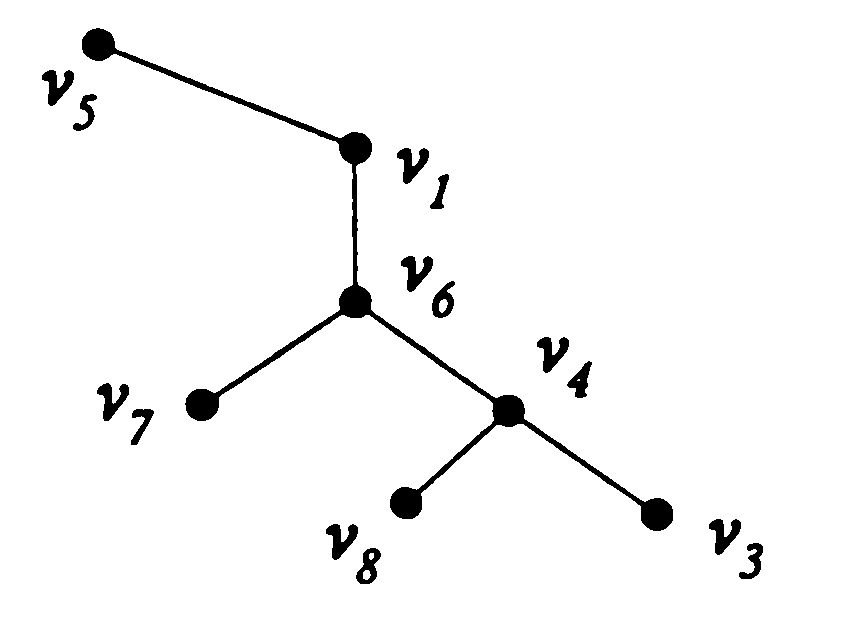
Рассмотрим коды *Прюфера*, которые *однозначно* представляют дерево с *n* вершинами в виде последовательности  длины *n – 2*, где элемент последовательности  может быть любым целым числом между *1* и *n*. Верно также и обратное утверждение.

*Алгоритм построения кода для**.*

1. Для выбора  взять лист  с *наименьшим* значением *j*. Всегда существует единственное *k* такое, что  является ребром дерева. Удалить это ребро и положить .
2. Если выбрано , то для выбора  из оставшегося дерева взять лист с *наименьшим* значением *j*. Всегда существует единственное *k* такое, что является ребром дерева. Удалить это ребро и положить .
3. Продолжать, пока не будет выбрано .

Поясним построение на примере. Пусть задано дерево *T* с *8* вершинами, *рисунок 1а*. Вершина  - лист с наименьшим индексом, удаляем лист вместе с ребром  и полагаем  равным индексу родителя вершины , . (*рисунок 1б.*)

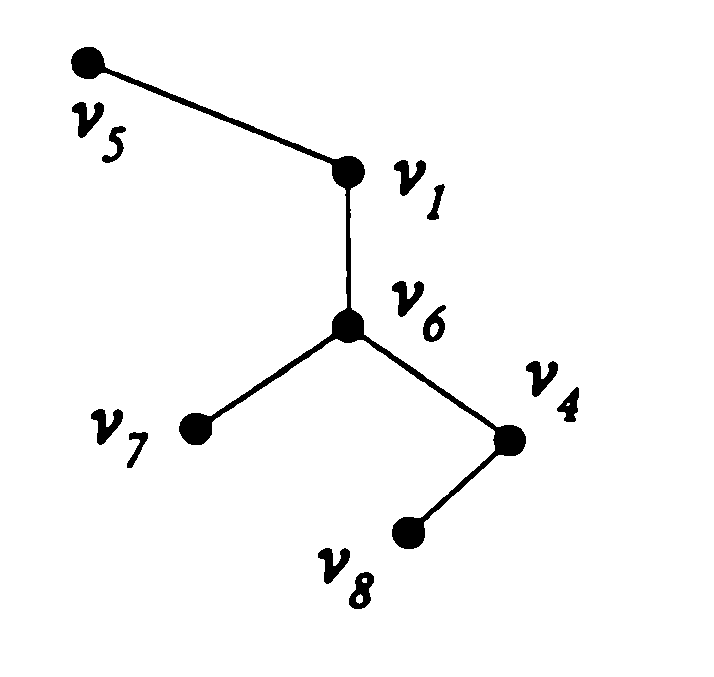
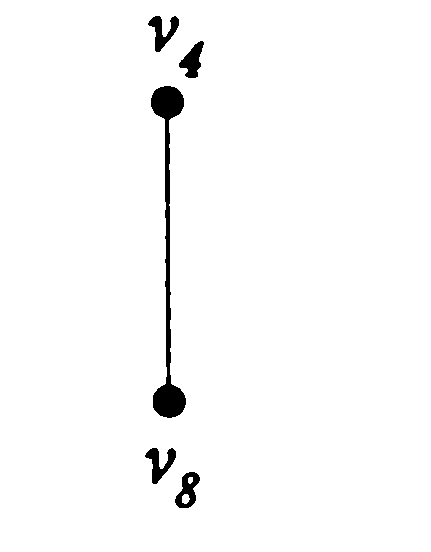




*Рис. 1а. Рис 1б.*

Поскольку, в оставшемся дереве вершина  - лист с наименьшим индексом, удаляем ребро  и полагаем  (*рис. 2а*). Далее повторяем наш алгоритм. В результате получим только ребро, изображенное на *рис. 2б*. Искомая числовая последовательность длины *6* имеет вид:

*1, 4, 1, 6, 6, 4*.

*Рис. 2а. Рис. 2б.*

Теперь, имея числовую последовательность, по алгоритму декодировки мы *однозначно* можем восстановить дерево.

*Отметим, что число вхождений в последовательность индекса каждой вершины на единицу меньше ее степени*.

*Алгоритм декодировки последовательности.*

1. Для каждого если *i* появляется в последовательности раз, положить

.

1. Прочитать и сформировать ребро, , где *j* — наименьшая метка, такая что .

3. Уменьшить и на *1*.

4. Предположим, что прочитано, читать и формировать ребро ,

где *j* — наименьшая такая метка, что .

5. Уменьшить и на *1*.

6. После того, как прочитано, создать ребро между двумя оставшимися

вершинами степени *1*.

. Рассмотрим последовательность *1,4,1,6,6,4* и воспользуемся ею для восстановления исходного дерева.

Поскольку *1*, *4* и *6* появляются в последовательности дважды, то

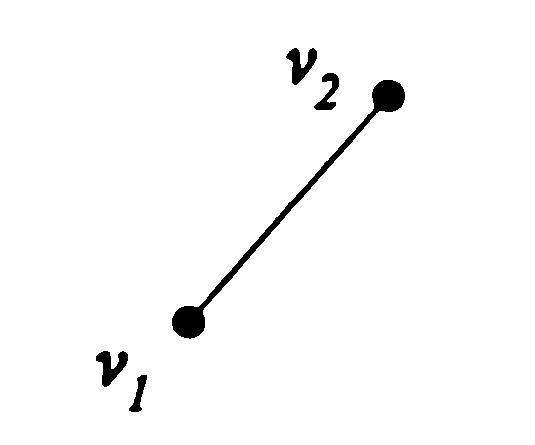
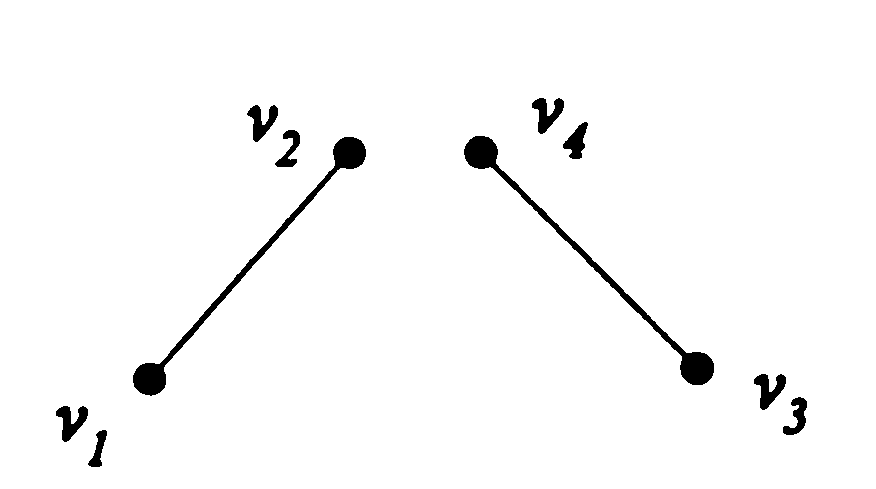
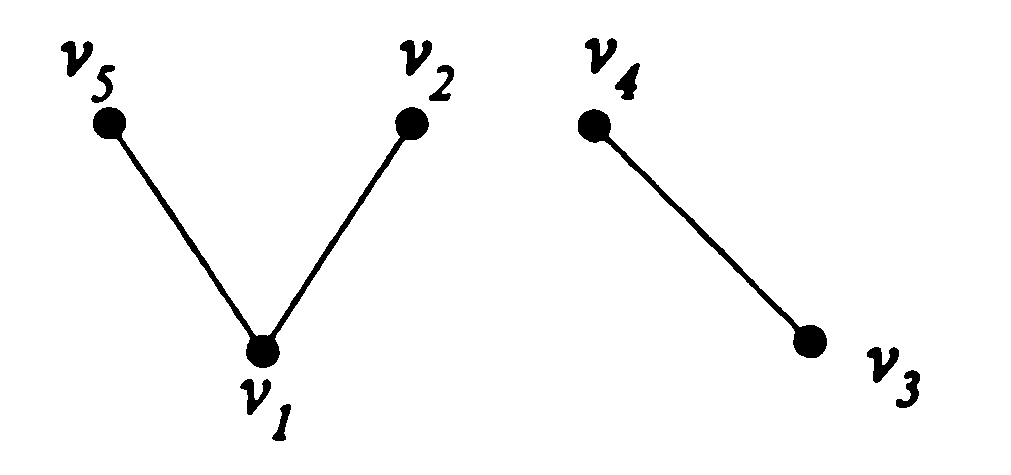
.

А так как *2*, *3* и *5, 7, 8* не встречаются вообще, то

Запишем их с помощью восьмерки из чисел *(3,1,1,3,1,3,1,1)*, которую назовем

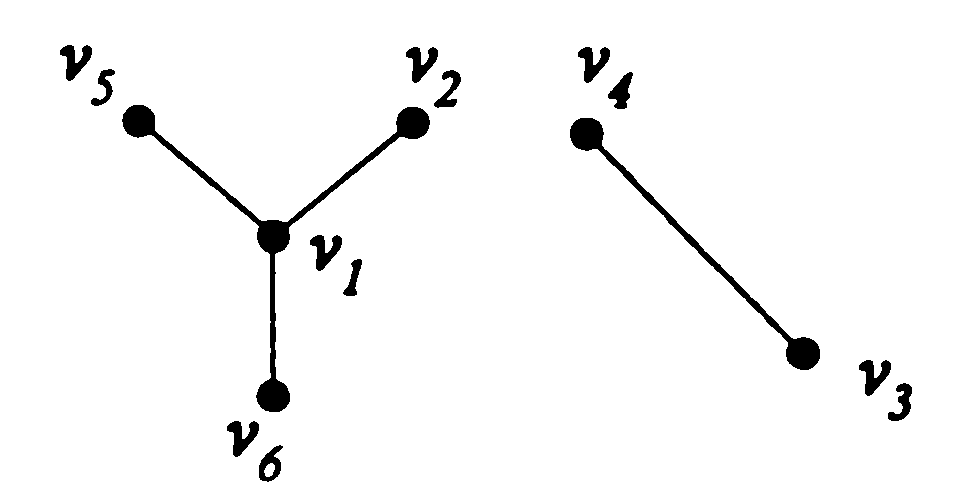
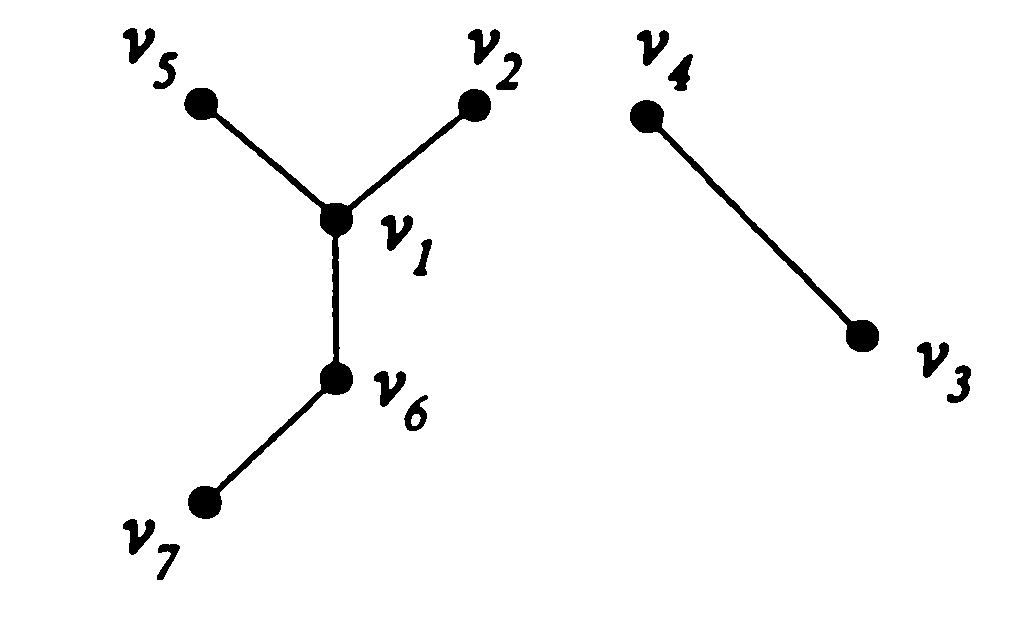
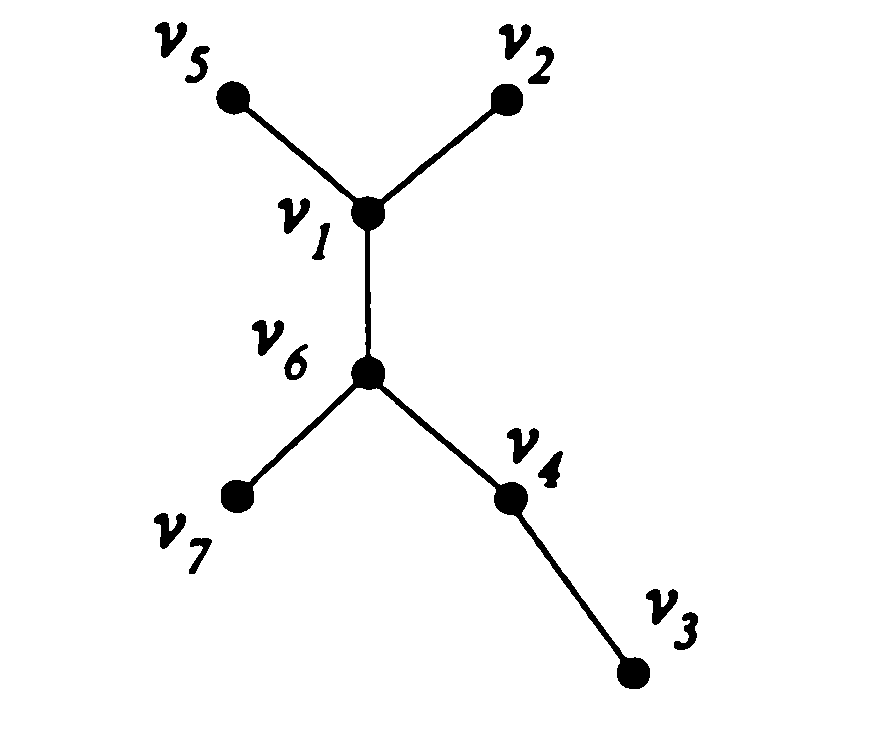
*восьмеркой* степеней.

1. Теперь считываем и, поскольку среди всех вершин степени *1* имеет аименьший индекс, создаем ребро (*рис 3a*). Положим и , поэтому восьмерка степеней теперь имеет вид *(2,0,1,3,1,3,1,1)*.

*Рис. 3а*. *Рис. 3б*. Рис. 3в

2. Далее считываем и, потому что среди всех вершин степени *1* имеет наименьший индекс, создаем ребро , после чего получаем граф, изображенный на *рис. 3б*. Полагаем и , поэтому восьмерка степеней теперь имеет вид *(2,0,0,2,1,3,1,1)*.

1. Теперь считываем и, поскольку среди всех вершин степени *1* имеет наименьший индекс, создаем ребро , после чего получаем граф, изображенный на *рис. 3в*. Полагаем и поэтому восьмерка степеней теперь имеет вид *(1,0,0,2,0,3,1,1)*.
2. Считываем и, поскольку среди всех вершин степени *1* имеет наименьший индекс, создаем ребро после чего получаем граф, изображенный на *рис. 3г*. Полагаем и , поэтому восьмерка степеней теперь имеет вид *(0,0,0,2,0,2,1,1).*  
       
    *Рис 3г. Рис.3д Рис.3е.*
3. Считываем и, потому что среди всех вершин степени *1* наименьшую степень, создаем ребро после чего получаем граф, изображенный на *рис.3д*. Полагаем и , поэтому восьмерка степеней теперь имеет вид *(0,0,0,2,0,1,0,1)*.
4. Считываем и, поскольку среди всех вершин степени *1* наименьший индекс, создаем ребро , после чего получаем граф изображенный на *рис.3е*. Полагаем и , поэтому восьмерка степеней теперь имеет вид *(0,0,0,1,0,0,0,1)*. Наконец, поскольку все последовательности прочитаны и , формируем ребро и получаем искомое дерево из *рис. 1а*.

.

***Задание.***

1. Написать программу *кодировки/декодировки* деревьев кодами *Прюфера*. Варианты деревьев указаны в *таблице 1*, деревья заданы списком ребер.
2. Графически изобразить дерево.
3. Рассмотреть указанное дерево как бинарное дерево с корнем в вершине *0*. Найти его высоту,, проверить является ли дерево сбалансированным. В случае отрицательного ответа, построить сбалансированное бинарное *дерево поиска* по заданному через таблицу списку вершин дерева. Корень в этом случае выбираем как *середину* всей последовательности ключей дерева, в левом поддереве все ключи меньше значения ключа корня, в правом – больше .

*Таблица 1.*

|  |  |
| --- | --- |
| *№ n/n* | *Дерево* |
| 1. | *(0,1),(0,2),(1,3),(1,4),(2,5)*  *(2,6),(5,7),(5,8),(8,9),(8,10)* |
| 2. | *(0,1),(0,2),(1,3),(1,4),(2,5)*  *(2,6),(5,7),(7,8),(7,9),(8,10)* |
| 3. | *(0,1),(0,2),(1,3),(1,4),(2,5)(2,6),*  *(5,7),(7,8),(7,9),(9,10),(10,11)* |
| 4. | *(0,1),(0,2),(1,3),(1,4),(2,5)*  *(2,6),(6,7),(6,8),(8,9)* |
| 5. | *(0,1),(0,2),(1,3),(1,4),(2,5)*  *(2,6),(4,7),(7,8),(7,9)* |
| 6. | *(0,1),(0,2),(1,3),(1,4),(2,5),(11,12)*  *(2,6),(4,7),(7,8),(7,9),(9,10),(9,11)* |
| 7. | *(0,1),(0,2),(1,3),(1,4),(2,5), (2,6),(3,7),*  *(3,8),(6,9),(8,10),(8,11),(10,12),(11,13)* |

***Вопросы к лабораторной работе (ответы в пмсьменном виде).***

1. Как задается дерево в программе?
2. Что такое *свободное дерево*, *ориентированное дерево*, *сбалансированное* *дерево*?
3. Как определяется *высота* дерева, *уровень* вершины?
4. Для вашего дерева указать все порядки обхода дерева: *прямой*, *обратный*, *симметричный*, *в* *глубину*, *в* *ширину*.
5. \*Что такое дерево *Хаффмана*, для чего оно служит и алгоритм его построения (пример)?