# Лабораторная работа 1

**1.1. [# 5]** Сгенерировать с помощью алгоритма Нарайаны порождения всех перестановок с выводом времени работы до сотых секунд для . Записать время порождения всех перестановок для , , без вывода полученных последовательностей. Оценить время работы программы для входа и .

**1.2. [# 15]** Реализовать алгоритм Джонсона – Троттера с выводом времени работы до сотых секунд. Записать время порождения всех перестановок для , , без вывода полученных последовательностей. Оценить время работы программы для входа N=50 и N=100.

**1.3. [# 5]** Реализовать алгоритм порождения случайной перестановки.

**1.4. [# 15]** Реализовать алгоритм порождения сочетаний в лексикографическом порядке с выводом времени работы до сотых секунд. Записать время порождения всех сочетаний для и , , . Оценить время работы программы для и и .

**1.5. [# 10]** Реализовать алгоритм порождения размещений с выводом времени работы до сотых секунд. Записать время порождения всех размещений для и , . Оценить время работы программы для и .

**1.6.** **[# 10]** Реализовать алгоритм порождения всех подмножеств множества из , элементов. Записать время порождения всех подмножеств множества.

**1.7.[# 25]** Реализовать алгоритм порождения композиции положительного числа в последовательность положительных целых чисел , . Здесь учитывается порядок чисел и

**1.8.** **[# 35]** **Задача об укладке рюкзака.** Есть различных предметов. Каждый предмет с номером , где , имеет заданный положительный вес и стоимость . Нужно уложить рюкзак так, чтобы общая стоимость предметов в нем была не менее , а вес не превышал заданного . Форма предметов значения не имеет

**Вход.** В первой строке три числа и , в следующих строчках – по два числа (вес и стоимость )

**Выход.** В первой строке – вес получившегося рюкзака, во второй строчке строка из 0 и 1 поясняющих какие предметы берутся, а какие – не берутся

# Лабораторная работа 2

**2.1. [# 50]** В заданном неориентированном связном графе найти все точки сочленения. Граф задан матрицей смежности.

**Вход.** Граф задан списком рёбер. Первая строка исходного файла содержит два целых числа () – количество вершин и – количество рёбер.

В последующих строчках содержатся по 2 числа , разделенных пробелами: наличие ребра между вершинами и .

Вершины пронумерованы от 1 до

**Выход.** В первую строчку вывести через пробел номера точек сочленения.

**2.2. [# 50]** Проверить, что ориентированный граф является сильно связанным. Граф задан матрицей смежности.

**Вход.** Первая строка исходного файла содержит целое число () – количество вершин. В последующих строках содержатся по чисел, разделенных пробелами: вес ребер между вершинами. Вершины пронумерованы от 1 до

**Выход.** В единственной строчке вывести 1 или 0 в зависимости от того сильно связанный граф или нет.

**2.3. [# 50]** Для ориентированного графа найти гамильтонов цикл методом ветвей и границ. Граф задан матрицей смежности.

**Вход.** Первая строка исходного файла содержит целое число () – количество вершин. В последующих строках содержатся по чисел, разделенных пробелами: вес ребер между вершинами. Вершины пронумерованы от 1 до .

**Выход.** В единственной строчке вывести список вершин, образующих направление гамильтонова цикла.

**2.4. [# 50]** Проверить, что граф плоский и произвести укладку графа, используя гамма-алгоритм.

**Вход.** Первая строка исходного файла содержит целое число () – количество вершин. В последующих строках содержатся по чисел, разделенных пробелами: вес ребер между вершинами. Вершины пронумерованы от 1 до .

**Выход.** Если граф не планарный, то в единственной строчке вывести одно число 0. Если граф планарный, то в первой строчке вывести количество граней .

В следующих строчках вывести циклы, образующие каждую грань.

**2.5. [# 50]** Раскрасить заданный граф, используя алгоритм Ершова.

**Вход.** Первая строка исходного файла содержит целое число () – количество вершин. В последующих строках содержатся по чисел, разделенных пробелами: вес ребер между вершинами. Вершины пронумерованы от 1 до

**Выход.** В строчках вывести по два числа: номер вершин (от 1 до ) и номер цвета вершины (нумерация цветов начинается с 1)

# Лабораторная работа 3

**3.1.** **[# 100]** Провести триангуляцию Делоне для заданного множества точек.

**Вход.** Первая строка исходного файла содержит целое число () – количество вершин. В последующих строках содержатся по чисел, разделенных пробелами: вес ребер между вершинами. Вершины пронумерованы от 1 до

**Выход.** В первой строчке вывести количество внутренних треугольников . В следующих строчках вывести циклы, образующие каждую грань

**3.2.** **[# 100]** На плоскости заданы множество точек на плоскости (терминалы). Используя алгоритм Кокейна, найти кратчайшую сеть, соединяющую все терминалы, используя дополнительные вершины – точки Штейнера

**Вход.**Первая строка исходного файла содержит одно целое число () – количество вершин (терминалов). В последующих строках содержатся по два числа, разделенных пробелами: координаты терминалов. Терминалы пронумерованы от 1 до

**Выход.** В первой строчке одно натуральное число – количество точек Штейнера. Точки Штейнера пронумеровать . В следующих строчках по два числа: координаты точек Штейнера. В следующих строчках по два числа: номера вершин, соединенных ребром в дереве Штейнера.

**3.3.** **[# 50]** Для заданного графа, содержащего вершин, найдите количество остовных деревьев.

**Вход.** Первая строка исходного файла содержит одно целое число () – количество вершин. Вершины пронумерованы от 1 до . В последующих строках содержатся по чисел, разделенных пробелами: вес ребер между вершинами.

**Выход.** В первой строке должно содержаться одно натуральное число – количество остовных деревьев.

# Лабораторная работа 4

**4.1. [# 100]** Задан орграф , где – множество вершин, – множество ребер, |V|=n. Каждой дуге поставлено в соответствие положительное вещественное число – пропускная способность. Вершину с номером 0 назовём источником, а вершину с номером стоком. Требуется найти максимальный поток.

**Вход.** Первая строка исходного файла содержит целое число () – количество вершин. В последующих строках содержатся по чисел, разделенных пробелами: пропускная способность между вершинами. Вершины пронумерованы от 0 до .

**Выход.** Первая строка исходного файла содержит целое число – количество вершин. В последующих строках содержатся по чисел, разделенных пробелами: величина потока между вершинами.

**4.2.** **[# 50]** Задан двудольный граф , где – множество вершин, ; – множество ребер. В заданном двудольном графе найти полное паросочетание, воспользовавшись алгоритмом Куна.

**Вход.** Двудольный граф задается матрицей *, , ,* в которой , если есть ребро и , если такого ребра нет. Первая строка входного файла содержит два числа  – количество вершин в каждой доле. В последующих строках содержатся по чисел матрицы . Вершины в каждой доле пронумерованы от 1.

**Выход.** В первой строчке Yes либо No, в зависимости от того есть максимальное паросочетание или нет. Во втором случае, следующих строчках по два числа – вершины, соединенные ребром (первое число – первая доля, второе – вторая).

**4.3.** **[# 50]** Задан двудольный граф , где – множество вершин, ; – множество ребер. В заданном двудольном графе найти полное паросочетание, воспользовавшись алгоритмом Рабина-Вазирани.

**Вход.** Двудольный граф задается матрицей *, ,* , в которой , если есть ребро и , если такого ребра нет. Первая строка входного файла содержит два числа  – количество вершин в каждой доле. В последующих строках содержатся по чисел матрицы . Вершины в каждой доле пронумерованы от 1

**Выход.** В первой строчке Yes либо No, в зависимости от того есть максимальное паросочетания или нет. Во втором случае, следующих строчках по два числа – вершины соединенные ребром (первое число – первая доля, второе – вторая)

**4.4.** **[# 50]** Задан двудольный граф *,* где – множество рабочих, – множество работ, ; – множество ребер, определяет стоимость выполнения рабочим работу . Каждому рабочему подобрать определенный вид работы, чтобы суммарная стоимость выполнения всех работ была минимальной, воспользовавшись венгерским алгоритмом

*Вход.* Двудольный граф задается матрицей , в которой . Первая строка входного файла содержит целое число  *()* – количество вершин в одной доле. В последующих строках содержатся по чисел матрицы . Вершины в каждой доле пронумерованы от 1 до

*Выход.* В первой строчке вывести общую стоимость выполняемых работ. В следующих строчках вывести по два числа: рабочего и выполняемую им работу