ЛАБОРАТОРНАЯ РАБОТА № 5
ПОИСК КРАТЧАЙШХ ПУТЕМ МЕТОДОМ ФЛОЙДА

Цель работы: Изучить алгоритм поиска кратчайших путей в графе используя алгоритм Флойда.

Теоретические сведения

Алгоритм Флойда (иногда называемый алгоритм Флойда–Уоршелла) — алгоритм нахождения длин кратчайших путей между всеми парами вершин во взвешенном ориентированном графе. Работает корректно, если в графе нет циклов отрицательной величины.

Для заданного ориентированного взвешенного графа алгоритм находит кратчайшие расстояния между всеми парами вершин за время $O(n^3)$. Алгоритм применим к графам с произвольными, в том числе с отрицательными, весами.

Ключевая идея алгоритма — разбиение процесса поиска кратчайших путей на фазы.

Перед *k*-ой фазой (k=1..n) считается, что в матрице расстояний **C[,]** сохранены длины таких кратчайших путей, которые содержат в качестве внутренних вершин только вершины из множества {1,2,…,k-1} (вершины графа нумеруются начиная с единицы).

Иными словами, перед *k*-ой фазой величина **C[i,j]** равна длине кратчайшего пути из вершины *i* в вершину *j*, если этому пути разрешается заходить только в вершины с номерами, меньшими k (начало и конец пути не считаются).

Легко убедиться, что чтобы это свойство выполнилось для первой фазы, достаточно в матрицу расстояний **C[,]** записать матрицу смежности графа. При этом, если между какими-то вершинами ребра нет, то записать следует величину "бесконечность". Из вершины в саму себя всегда следует записывать величину 0.

Пусть теперь мы находимся на *k*-ой фазе, и хотим пересчитать матрицу **C[,]** таким образом, чтобы она соответствовала требованиям уже для (k+1)-ой фазы. Зафиксируем какие-то вершины *i* и *j*.

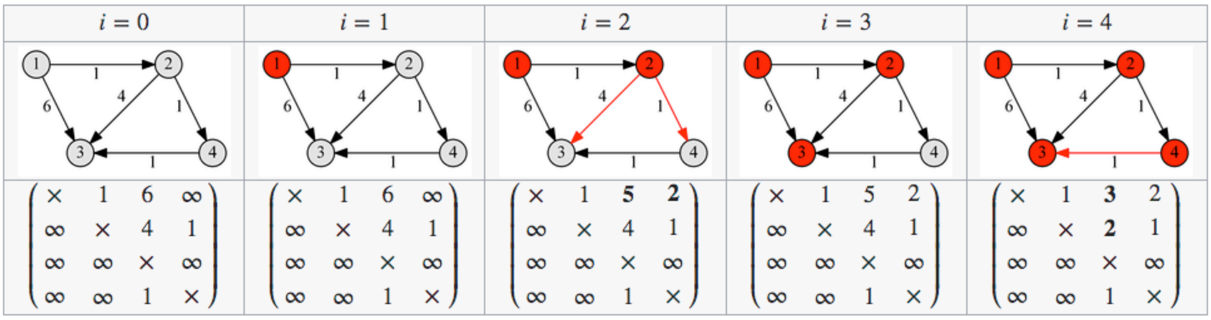
Возможно два случая:

1. Кратчайший путь из вершины *i* в вершину *j*, которому разрешено дополнительно проходить через вершины {1, 2, …, *k*}, совпадает с кратчайшим путём, которому разрешено проходить через вершины множества {1, 2, …, *k*-1}. В этом случае величина **C[i,j]** не изменится при переходе с *k*-ой на (*k*+1)-ую фазу.

2. Текущий кратчайший путь стал короче пути предыдущего шага. Это означает, что "новый" кратчайший путь проходит через вершину *k*. Текущий кратчайший путь делится вершиной *k* на две половинки (одна идущая *i* –> *k*, а другая — *k* –> *j*), то каждая из этих половинок уже не заходит в вершину *k*. Но тогда получается, что длина каждой из этих половинок была посчитана ещё на (k-1)-ой фазе, и нам достаточно взять просто сумму **C[i,k]**+**C[k,j]**, она и даст длину "нового" кратчайшего пути.

Таким образом, вся работа, которую требуется произвести на k-ой фазе — это перебрать все пары вершин и пересчитать длину кратчайшего пути между ними. В результате после выполнения n-ой фазы в матрице расстояний **C[i,j]** будет записана длина кратчайшего пути между i и j, либо бесконечность, если пути между этими вершинами не существует.

Пример работы алгоритма:



Псевдокод алгоритма:

Матрица **C** содержит длины кратчайших путей между всеми вершинами графа. Перед работой алгоритма матрица **C** заполняется длинами рёбер графа (или “машинной бесконечностью”, если ребра нет).

```
for k = 1 to n
  for i = 1 to n
    for j = 1 to n
      C[i,j] = min(C[i,j], C[i,k] + C[k,j])
```

Порядок выполнения работы

1. Составить программу, осуществляющую ввод матрицы смежности.
2. Используя алгоритм Флойда найти кратчайшие пути в заданном графе.