Задача 1

Для решения используем алгоритм BFS. Пусть невзвешенный граф задан списками смежности и данное в условие ребро соединяет вершины u и u. Запустим BFS из вершины a, получим $mindist[a, b]$, $mindist[a, u]$, $mindist[a, v]$ — минимальные расстояния (в количестве ребер) между указанными в скобках вершинами. Затем запустим BFS из вершины b и получим $mindist[b, u]$, $mindist[b, v]$. Далее, если верно одно из равенств:

$$mindist[a, b] = \begin{bmatrix} mindist[a, u] + mindist[v, b] + 1 \\ mindist[a, v] + mindist[u, b] + 1 \end{bmatrix}$$

то ответ «да», иначе ответ «нет». Это действительно так, поскольку, если одно из равенств выполнено, то можно составить путь, включающий данное ребро и являющийся минимальным путем из a в b.

Задача 2

Изменим алгоритм BFS из Кормена (стр. 632), который работает на графах с ребрами веса один и находит кратчайшие пути из истока до всех остальных вершин, так, чтобы наш алгоритм искал кратчайшие пути в графе с весами ребер 0 и 1.

```
for Каждой вершины u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for Каждой вершины v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

Рис. 1: Обычный BFS

Пусть мы находимся на вершине графа u в процессе работы обычного BFS. Рассмотрим ребро (u, v). Вершина v находится на таком же расстонии $H(u)$ от истока, если вес этого ребра равен нулю, либо на «следующем» расстояни $H(u)+1$, если вес ребра один. Обычный алгоритм на обычном графе хранит в очереди вершины, находящиеся на максимум двух соседних расстояниях либо на одном и расположенные в порядке возрастания раччтояний до них из истока. Т. е. если мы извлекаем из очереди вершину u, то в ней остаются вершины на расстояниях $H(u)$ либо $H(u)+1$. Если вес ребра (u, v) равен нулю, то при добавлении вершины v в очередь, распологаем ее в начале очереди и не увеличиваем $v.d$ (минимальное расстояние до нее), иначе повторяем действие обычного алгоритма (добавление вершины в конец и $v.d = u.d+1$). Массив предков заполняется также. В итоге, свойтсво очереди алгоритма BFS сохраняется, откуда следует минимальность найденных путей.

Задача 4

а) В графе с отрицательными циклами мы не можем говорить про циклы минимального веса, поэтому модифицируем алгоритм Флойда-Уоршелла для того, чтобы проверить граф на наличие отрицательного цикла и вернуть вычисленные веса минимальных путей при его отсутствии. Для этого запустим алгоритм Флойда-Уоршелла на произвольном графе и затем проверим диагональные элементы полученнной матрицы расстояний на наличие отрицательных значений. Если таковых нет, возвращаем матрицу результатов, являющуюся правильным ответом, иначе в графе есть цикл отрицательного веса. Это так, поскольку изначально значения (i, i) на диагонали равны нулю, а в процессе работы они могут обновиться, только если найден путь меньшей длины $(i \cdots k \cdots i)$, т. е. отрицательной длины, а это и есть цикл отрицательной длины.

б) Запустив алгоритм, получаю на каждой итерации всех трех вложенных циклов одну и ту же матрицу (измененний нет вообще), то есть $d^0 = d^1 = d^2 = d^3$.

Задача 5

b)Запустив алгоритм на графе из вершины D, находим, что метки расстояний вершин не будут меняться после первой итерации по внешнему циклу и будут равны $[0, 7, 15, 19, 9, 11, 13, 14]$.

c) Чтобы на каждой фазе менялись метки расстояний необходимо все вершины расположить в одну линию и соединить соседние вершины ребрами (граф в виде цепочки). Т. к. на i-той итерации внешнего цикла алгоритма все пути длины $\leq i$ прорелаксированы, то чтобы получить min расстояние между началом и концом графа-цепочки нужно повторить все $|V| - 1$ итерации внешнего цикла.

Задача 6

Вершина в конденсации орграфа соответсвует компоненте сильной связности этого графа. Всякая компонента сильной связности графа содержит циклы: если две вершины принадлежат одной компоненте сильной связности, то по определению есть путь с началом в первой вершине и концом во второй вершине и другой путь с началом во второй вершине и концом в первой вершине, т. е. есть цикл, содержащий данные две вершины (для любых двух вершин в компоненте сильной связности есть цикл, проходящий через обе эти вершины $*$).

Если отношение частичного порядка строгое, то циклов в графе нет. Иначе, рассмотрев два соседних элемента цикла, получаем противоречие двух противоположных знаков отношения частичного порядка: одно из них соответствует ребру между этимим вершинами, а второе получаем в силу транзитивности ОЧП по остальным ребрам в цикле. Таким образом при строгом ОЧП в графе нет компонент сильной связности из более чем одной вершины, а значит, конденсация графа равна самому графу и в конденсации то же число вершин.

Если ОЧП нестрогое, то циклы возможны, но тогда в силу свойства нестрогого ОЧП: $(a \leq b) \wedge (b \leq a) \Rightarrow (a = b)$, получаем, что все вершины в одном цикле равны. Из $*$ получаем, что компонента сильной связности состоит из вершин с равными значениями. Тогда количество вершин в конденсации графа равно числу различных значений вершин графа, т. к. вершины с каким-то определенным значением лежат в одной компоненте связности.

Задача 7

a) Рассмотрим шаги произвольного поиска в глубину. Пусть первой из двух вершин u и v в обходе встретилась вершина v, тогда из нее никак обход не дойдет до вершины u, в силу ее недостижимости, а значит, вершина v закроется раньше вершины u, котораяя будет открыта точно после того, как закроется v. Если же первой будет открыта u, то в силу DFS и того, что v достижима из нее, она будет открыта после u, затем будет закрыта, а после этого (эти шаги не обязательно последовательны) будет закрыта вершина u, когда все вершины достижимые из нее уже закрыты. В итоге, $f[u] > f[v]$.

b) Поскольку пункт a) верен для любых двух вершин графа, то после упорядочивания по убыванию меток закрытия все ребра орграфа будут указывать в сторону убывания метки закрытия. Если есть хотя бы одно ребро направленное противоположно, то получаем цикл в графе (пусть есть обратное ребро (u, v), тогда есть путь из v в u, т. к. v есть предок u, и есть ребро (u, v), а значит, есть цикл). Противоречие с ацикличностью графа. Значит, все ребра направлены в сторону убвания метки закрытия.

Задача 8

a) Рассмотрим две произвольные вершины одной компоненты сильной связности исходного графа u и v. Поскольку в ней есть путь из v в u, то в инвертированном графе есть путь из u в v и наоборот. Значит, две вершины остаются взаимодостижимыми, если они обе принадлежат одной компоненте связности. Иначе, если u принадлежит другой компоненте связности, то в исходном графе нет пути $(u \to v)$ или $(v \to u)$, значит, в инвертированном графе эти вершины тоже не будут принадлежать одной компоненте связноти. В итоге, инвертирование ребер не меняет компонент сильной связности орграфа.

b) Шаги алгоритма Косараю есть разворачивание ребер за $O(|E|)$, DFS на инвертированном графе за $O(|V| + |E|)$, DFS на исходном графе за $O(|V| + |E|)$. В итоге, временная сложность алгоритма $O(|V| + |E|)$. Инвертированный граф будем хранить в новой матрице, также нужны массив меток закрытия и массив для компонент сильной связности, каждый размера $|V|$. В итоге нужно $O(|V|)^2$ ячеек памяти.

c) Конденсация любого графа есть ациклический орграф, потому что если есть цикл в конденсации графа, то есть цикл содержащий хотя бы одну вершину, принадлежащую каждой компоненте сильной связности этого графа, а значит, каждые две вершины из компонент сильной связности, соединенных этим циклом взаимно достижимы, т. е. принадлежат одной компоненте сильной связности, что противоречит данной конденсации (цикл в цонденсации схлопнется в одну вершину). Т. к. конденсация есть ациклический граф, то на ней всегда можно провести топологическую сортировку.