1.

Неориентированный граф можно рассматривать как ориентированный на V вершинах с $2|E|$ ребрами: как в ориентированном графе и их транспонированных (перевернутых).

а)

Да, это верно.

Доказательство:

От противного.

Если ребро минимального веса e не было включено в какое-то MST, то удаление любого из ребер (большего веса) в цикле, образованном после добавления e к MST, привело бы к остовному дереву меньшего веса. Противоречие.

б)

Да, это верно.

От противного.

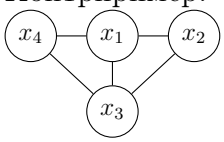
Если ребро e принадлежит разрезу и MST, но не является самым минимальным в разрезе, то можно удалить это ребро и взять другое с минимальным весом, так что получится MST с меньшим весом.

Противоречие.

в)

Нет, это неверно.

Контрпример:



И пусть веса ребер: $w_{1-2} = 1$, $w_{2-3} = 1$, $w_{1-3} = 2$, $w_{1-4} = 3$, $w_{3-4} = 4$

Остовное дерево состоит из ребер: $w_{1-2} = 1$, $w_{2-3} = 1$, $w_{1-4} = 3$

Кратчайший путь между вершинами 3 и 4 – $3 - 1 - 4$, однако он не является частью минимального остовного дерева.

2.

Минимальное остовное дерево – это связный ациклический граф, имеющий минимально возможный вес. MST содержит все вершины исходного графа. При выборе ребер, входящих как в T, так и в связный подграф H, мы получим связный ациклический граф, ведь в обоих графах от каждой вершины есть путь до каждой, так что при взятии ребер, содержащихся в них обоих, это свойство не потеряется. Полученное дерево будет обладать минимальным весом среди всех возможных деревьев графа H, так как его ребра взяты из MST графа G, от которого он индуцирован, так что обеспечено, что не сущетсвует дерева из ребер H с меньшим весом, иначе это ребро можно было удалить и заменить ребром с меньшим весом, принадлежащим H и уменьшить вес MST графа G.

3.

В структре $Union - Find$ операции поиска и объединения стоят $O(\log n)$ (согласно лекции).

Докажем, что за m операций можно реализовать алгоритм Крускала на графе на m ребрах и n вершинах.

Надо привести последовательность операций $Union - Find$, так что будем предполагать, что предобработка сделана (ребра отсортированы, хотя это даже и не портит асимптотику, так как стоит $O(m \log m)$, что можно переписать $O(m \log n)$).

При рассмотрении каждого ребра $e = (v, w)$ мы вычисляем $Find(u)$ и $Find(v)$ и проверяем результаты на равенство, чтобы узнать, принадлежат ли v и w разным множетсвам. Если алгоритм решает включить ребро e в дерево, то операция $Union(Find(u), Find(v))$ объединяет два множетсва. Подробное описание работы этих операций было на лекции.

В ходе этих действий выполняются не более $2m$ операций $Find$ и $n - 1$ операций $Union$, так что итоговая сложность: $O(m \log n)$.

4.

Неориентированный граф можно рассматривать как ориентированный на V вершинах с $2|E|$ ребрами: как в ориентированном графе и их транспонированных (перевернутых).

Алгоритм:

Для начала обработаем вершины из множества U. Пусть u – вершина из множества U, рассмотрим все ребра, инцидитентные ей и выберем ребро с минимальным весом, которые идут из множетсва U в множество $V \setminus U$ (то есть являются перекрестными в этом разрезе), также проделаем со всеми вершинами из множества U. Заметим, что циклов получится не может, так как мы не соединяем ребрами вершины из множества U. Если вдруг нет ребер из какой-то вершины, не между вершинами из U, то алгоритм выдаст сообщение о том, что такого дерева нет. А на вершинах $V \setminus U$ проведем обычный алгоритм Крускала.

Корректность:

Для вершин из множества U мы выбираем индидентные ребра с минимальным весом, причем, как было сказано выше, циклов получится не может, а корректность алгоритма Крускала была доказана на лекции, так что в итоге получаем связный ациклический граф с минимальным весом с листьями из U. Если нет какого-то перекрестного ребра из вершины, принадлежащей множетсву U в $V \setminus U$, то алгоритм сообщает, что таких деревьев нет.

Сложность:

Просмотр всех вершин из U и поиск для каждой вершины перекрестного ребра минимального веса занимает $O(|U| \log |U|)$, а алгоритм Крускала: $O(|E| \log |V \setminus U|)$, так что в итоге получим $O(|E| \log |V|)$.

5.

Алгоритм:

Построим миниальное остовное дерево для каждого оператора. Будем брать вершины вышек каждого оператора и вершину, принадлежащую общегородской сети. Веса ребер между вышками, которые принадлежат оператору – c_{ij}, а между вершиной, принадлежащей оператору, и вершиной общегородской сети – ребро минимального веса среди всех c_{ij}, где j – вершина общегородской сети, а i – вершина вышки оператора. То есть, находим для каждой вершины оператора ребро минимального веса, соединяющее

ее и вершину из общегородской сети. Число ребер, необходимое для этих двух MST деревьев и будет минимальным количеством соединений минимальной стоимости. Для построения MST будем использовать алгоритм Крускала.

Корректность:

MST – связный ациклический граф минимального веса, то есть, его ребра и есть соединения минимальной стоимости и количество этих соединений минимально, но сохранено условие связности: из любой вышки сигнал может пройти до любой другой вышки. Алгоритм предполагает связь не только между своими вышками, которые нужно оплачивать, но и между своей вышкой и вышкой общегородской сети, а веса ребер между вышками общегородской сети можно считать нулевыми. Таким образом, эти ребра нулевого веса будут предпочтительнее и таким образом достигнется оптимальность решения.

Сложность:

Пусть множество вершин и ребер в общегородской сети: V_0, U_0, множество вершин операторов: V_1 и V_2 соответсвенно. При работе алгоритма Крускала будем считать, что граф полный на V_1 и V_2 вершинах, чтобы выбрать оптимальные соединения. Число ребер в полном графе на n вершинах: $\frac{n(n-1)}{2}$. Таким образом, сложность алгоритма Крускала: $(|V_1|^2 + U_0) \log (V_1 + U_0) + (|V_2|^2 + U_0) \log (V_2 + U_0)$. Если число вершин вышек примерно одинаковое у операторов и в сети общегородской, то получим: $O(|V|^2 \log |V|)$.

6.

2-приближенный алгоритм означает, что ошибка оптимальности пути будет не более, чем в 2 раза.

Данная задача аналогична задаче о коммивояжере.

Алгоритм:

Строим минимальное остовное дерево на вершинах, являющихся складом и пунктами, куда нужно попасть курьеру. Будем считать, что у нас полный граф на этих вершинах так, чтобы можно было найти ребро минимального веса. Строить будем по алгоритму Прима. Затем по списку вершин, которые посещаются при обходе при создании MST создадим гамильтонов цикл (посещает каждую вершину один раз), то есть, цикл, который посещает вершины в порядке перечисления в списке.

Корректность:

В алгоритме строится гамильтонов цикл, так что посещение и возврат в исходную точку гарантируется по определению. Докажем, что данный алгоритм ошибается в оптимальности пути, не более, чем в 2 раза.

Пусть H^* – оптимальный путь, который является турниром. Так как при удалении ребра получится остовное дерево, вес минимального остовного дерева T равен нижней границе веса оптимального пути, то: $w(T) \leq w(H^*)$.

При полном обходе дерева T составляется список вершин, которые посещаются впервые, если к ним происходит возврат после посещения поддерева (обозначим этот обход W). Поскольку при полном обходе каждое ребро проходится ровно по 2 раза, то $w(W) = 2w(T)$.

Откуда будет следовать, что $w(W) \leq 2w(H^*)$, то есть, вес полного обхода превышает вес оптимального турнира не более, чем в 2 раза.

Согласно неравенству треугольника посещение любой вершины в обходе W можно отменить, при этом вес не возрастет. Таким образом, многократно это повторяя, можно исключить все посещения каждой вершины, кроме первого, тогда будем иметь, что: $w(H) \leq w(W)$.

Таким образом, $w(H) \leq 2w(H^*)$, что и требовалось доказать.

Сложность:

Было показано на предыдущих лекциях для алгоритма Дейкстры в случае плотных графых, что его сложность: $O(|V|^2)$, каким и является граф в этой задаче, а алгоритм Прима в точности повторяет его, за исключением процедуры коррекции процедуры релаксации. Таким образом, сложность алгоритма $\mathbf{O(|V|^2)}$.