Гавриленко Арсений *777*

Домашняя работа 7

1 Сортировки

Задача 1. Наше орудие - неравенство треугольника. Отсортируем за O(n log n), дальше будем по особому проверять неравенство треугольника. Начнем с самого большго элемента в отсоритрованном массиве, дальше я придумал только за квадрат(.

Задача 2.

Задача 3. Используем MergeSort В цикле алгоритма слияния левая и правая половины сортируются по возрастанию, и мы хотим объединить их в один отсортированный массив. Обратите внимание, что все элементы в правой части имеют более высокие индексы, чем те, что расположены слева.(по построению алгоритма) Предположим массив [leftIndex] > массив [rightIndex]. Это означает, что все элементы в левой части, следующие за элементом с индексом leftIndex, также больше, чем текущий в правой части (потому что левая уже отсортирована по возрастанию). Таким образом, текущий элемент в правой части генерирует инверсии numberOfElementsInTheLeftSide - leftIndex + 1, поэтому добавьте это в свой глобальный счет инверсии.

Как только алгоритм завершит выполнение, вы получите свой ответ, а сортировка слияния - O (n log n) в худшем случае.

2 Кучи

Задача 1. Прочитал Кормена по кучам, увидел доказательство, потом увидел дз - писать его сюда смысла навреное нет.

Задача 2. Удалим все элементы, которые дальше чем на K от корня. Очевидно среди пересечение множества удаленных и K минимальных - нулевое. Тогда для подстриженной кучи просто будем K раз вызывать Heapify, то есть брать минимум из корня и находить новый минимум(перестраивать кучу). Тогда асимптоика $O(KlogK)$, высота дерева - K, операцию вызывам K раз.

Задача 3.

3 BFS DFS

Задача 1. Поиском в ширину пройдемся по графу и удалим ребра с весом 0 (схлопнем). Потом еще раз применим алгоритм поиска в ширину, но заходя в каждую вершину будем в ней выставлять счетчик количества переходов до нее. Для корня счетчик = 0, для смежных с ним - 1 и тд.. Корректность <=> Корректность BFS, асимптотика = $O(V + E)$. как у BFS.

Задача 2. Модифицируем DFS: Произведём серию поисков в глубину в графе. Т.е. из каждой вершины, в которую мы ещё ни разу не приходили, запустим поиск в глубину, который при входе в вершину будет красить её в серый цвет, а при выходе - в чёрный. И если поиск в глубину пытается пойти в серую вершину, то это означает, что мы нашли цикл (если граф неориентированный, то случаи, когда поиск в глубину из какой-то вершины пытается пойти в предка, не считаются).

Корректность Корректность следует из построения. Если есть цикл, мы его найдем.

Асимптотика Такая же как и у DFS - O(V+E).
Сам цикл можно восстановить проходом по массиву предков.

Задача 3. Возможно я не понял задачу, но почему бы нам не запустить алгоритм задачи 2 из каждой вершины графа, каждый раз обновляя цикл, если нашли меньший, чем был до этого? Корректность следует из задачи 2. Асимптотика очень жирная, поэтому тоже очевидно проходит.

Задача 4. 1. Добавим ко всем ребрам по 1, тогда кратчайщий путь будет проходить по тем же ребрам, что и до этого.
2. Удалим ребра нулевого веса.
3. Задача сводится к первой, только после того как мы найдем все пути от вершины А до вершины Б - нужно будет выбрать кратчайший. Сделаем так - будем всегда хранить минимальный, если новый найденный короче - обновим минимальный. Самый короткий из них будет точно простой, потому что иначе был бы более короткий путь из которого убрали цикл(так как все ребра положительного веса).

Корректность Корректность сводится к корректности первой задачи(которя сводится к BFS). Задача корректна, так как по построению нет ребер отрицательной длины и , построив несколько путей из А в Б мы прото выбрали кратчайший.ъ

Асимптотика O(V+E) + O (E) - добавление веса. Итого O(V+E).

4 Дейкстра

Задача 1. Запустим Дейкстру из всех вершин множества T, то есть исток - набор T. Его найдем за O(V). Построим алгоритм:

- Заведём массив d[], в котором для каждой вершины v будем хранить текущую длину d[v] кратчайшего пути . Изначально d[s]=0, для всех s - из T, а для всех остальных вершин эта длина равна бесконечности.
- Создадим |T| массивов, которые для каждой вершины будут говорить, начатый из какой вершины из T ДЕкйстра дошел до этой.
- На очередной итерации выбирается вершина v с наименьшей величиной d[v] среди ещё не помеченных, понятно, что какие то |T| итераций будут выбираться вершины из T.
- Если пришли в вершину из T, то пометим ее как посещенную всеми другими из T. Если же пришли в вершину не из T , то пометим ее посещенной только той вершиной из T из которой мы пришли сюда. Затем проводим релаксацию всех смежных вершин и завершаем итерацию.

После того как отработает наш новый Дейкстра, линейно за O(V) пройдем по всем массивам и выберем минимумы расстояний для каждой пары.

Задача 2. (Аналогично 1ой)
Асимптотика как у оригинального дейкстры, говорит нам, что нужно управиться за константное количество Дейкстр. Построим такой алгоритм. Рассмотрим каждую пару цветов, таких пар $\frac{4*3}{2} = 6$. РАссмотрим конкретную одну пару , пусть красный и синий. Модифицируем Дейкстру:

- Исток - множество всех красных вершин. Перебрать все вершины и выбрать множество красных - O(E).
- По сути мы запускаем Дейкстру как бы с середины. Стандартный алгоритм, считаем новое расстоения, если оно больше чем текущие в вершине, меняем.
- Так получим расстояния от красных до всех остальных, пройдем опять за O(E) по всем синим и выберем минимальное расстояние.

Таким образом, за 6 Дейкстр мы решим задачу.