

Программа письменной части экзамена (магистратура)

1. Динамический массив. Амортизационный анализ. Амортизированное (учетное) время добавления элемента в динамический массив. Двусвязный и односвязный список. Операции. Объединение списков. Стек. Очередь. Дек. Двоичная куча. АТД “Очередь с приоритетом”.
2. Сортировка вставками, выбором. Сортировка слиянием. Сортировка с помощью кучи. Слияние K отсортированных массивов с помощью кучи. Нижняя оценка времени работы для сортировок сравнением. Быстрая сортировка. Выбор опорного элемента. Доказательство среднего времени работы. TimSort.
3. MSD, LSD. Сортировка строк. Binary QuickSort. Поиск k -ой порядковой статистики методом QuickSelect (без доказательства среднего времени работы). Поиск k -ой порядковой статистики за линейное время.
4. Деревья, родительские, дочерние узлы, листья. Количество ребер. Обходы в глубину. pre-order, post-order и in-order для бинарных деревьев. Обход в ширину. Дерево поиска. Поиск ключа, вставка, удаление.
5. Необходимость балансировки. Три типа самобалансирующихся деревьев. Декартово дерево. Оценка средней высоты декартового дерева при случайных приоритетах (без доказательства). Построение за $O(n)$, если ключи упорядочены. Основные операции над декартовым деревом. АВЛ-дерево. Красно-черное дерево.
6. Хеш-функции. Остаток от деления, мультипликативная. Деление многочленов - CRC. Обзор криптографических хеш-функций. CRC*, MD*, SHA*. Полиномиальная. Ее использование для строк. Метод Горнера для уменьшения количества операций умножения при ее вычислении.
7. Хеш-таблицы. Понятие коллизии. Метод цепочек (открытое хеширование). Метод прямой адресации (закрытое хеширование). Двойное хеширование.
8. Общая идея жадных алгоритмов. Задача о рюкзаке. Общая идея последовательного вычисления зависимых величин. Идея введения подзадач (декомпозиции) для решения поставленной задачи. Восходящее ДП. Нисходящее ДП, кэширование результатов.
9. Вычисление чисел Фибоначчи. Нахождение наибольшей возрастающей подпоследовательности за $O(N^2)$ и за $O(N \log N)$. Количество способов разложить число N на слагаемые. Количество способов разложить число N на различные слагаемые. Нахождение наибольшей общей подпоследовательности. Методы восстановления ответа в задачах динамического программирования. Расстояние Левенштейна. Персистентные структуры данных. Версии, возможность модифицировать любую версию. Примеры использования.

10. Персистентный стек. Персистентное дерево поиска. Добавление, удаление узла. Повороты. Рекуррентные выражения. Способы доказательства оценок: метод подстановки и метод разворачивания суммы. Мастер-теорема (без доказательства).
11. Ориентированный граф, псевдограф. Неориентированный граф, псевдограф. Связность в неор. графе, компоненты связности. Слабая и сильная связность в ор. графе. Компоненты слабой, сильной связности. Обход в глубину. Цвета вершин. Времена входа и выхода. Лемма о белых путях. Проверка связности неориентированного графа. Поиск цикла в неориентированном и ориентированном графе. Топологическая сортировка. Нахождение компонент сильной связности. Алгоритм Косарайю с доказательством корректности. Алгоритм Тарьяна с доказательством корректности. Компоненты реберной двусвязности. Мосты. Поиск мостов.
12. Критерий существования Эйлера пути и цикла в ориентированном и неориентированном графе. Поиск эйлера пути и цикла. Алгоритм Дейкстры. Доказательство корректности. Оценка времени работы. Дерево кратчайших путей.
13. Алгоритм A^* . Условие монотонности на эвристику. Допустимость эвристики. Примеры эвристик. Двусторонний алгоритм Дейкстры. Алгоритм Форда-Беллмана.
14. Нахождение кратчайших путей с учетом циклов отрицательного веса. Алгоритм Флойда. Доказательство. Восстановление пути. Нахождение цикла отрицательного веса. Алгоритм Джонсона. Добавление фиктивного корня и фиктивных ребер для запуска алгоритма Форда-Беллмана.
15. Остовное дерево. Построение с помощью обхода в глубину и в ширину. Определение минимального остовного дерева. Алгоритм Прима. Аналогия с алгоритмом Дейкстры. Алгоритм Крускала.
16. Алгоритм Форда-Фалкерсона. Поиск минимального разреза. Алгоритм Эдмондса-Карпа. Слоистая сеть. Алгоритм Диница. Оценка времени работы. Паросочетания. Паросочетание в двудольном графе. Максимальное паросочетание. Сведение поиска максимального паросочетания к поиску максимального потока. Алгоритм Куна.
17. RSQ и RMQ. Дерево отрезков. Обработка запросов от листьев. Обработка запросов от корня. Изменение значения в массиве, обновление дерева отрезков. Множественные операции. LCA. Метод двоичного подъема. Сведение LCA к задаче RMQ. Сведение RMQ к задаче LCA. Декартово дерево по неявному ключу. Интерфейс быстрого массива: Доступ к элементу в позиции i , Вставка элемента в позицию i , Удаление элемента из позиции i , Конкатенация двух массивов, разделение массива на два.

18. Префикс-функция. Тривиальный алгоритм нахождения. Алгоритм Кнута-Морриса-Пратта. Z-функция. Тривиальный алгоритм нахождения. Линейный поиск Z-функции. Суффиксный массив. Суффиксное дерево.

Обход дерева в порядке post-order

Ограничение времени	0.2 секунды
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Дано число $N \leq 10^4$ и последовательность целых чисел из $[-2^{31}..2^{31}]$ длиной N . Требуется построить бинарное дерево, заданное наивным порядком вставки. Т.е., при добавлении очередного числа K в дерево с корнем $root$, если $root \rightarrow Key \leq K$, то узел K добавляется в правое поддерево $root$; иначе в левое поддерево $root$. Выведите элементы в порядке post-order (снизу вверх).

Рекурсия запрещена.

Пример

Ввод	Вывод
10 7 2 10 8 5 3 6 4 1 9	1 4 3 6 5 2 9 8 10 7

Шаблон с ?

Ограничение времени	0.01 секунда
Ограничение памяти	12Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Шаблон поиска задан строкой длины m , в которой кроме обычных символов могут встречаться символы “?”. Найти позиции всех вхождений шаблона в тексте длины n . Каждое вхождение шаблона предполагает, что все обычные символы совпадают с соответствующими из текста, а вместо символа “?” в тексте встречается произвольный символ.

Время работы - $O(n + m + Z)$, где Z - общее -число вхождений подстрок шаблона “между вопросиками” в исходном тексте. $m \leq 5000$, $n \leq 2000000$.

Пример

Ввод	Вывод
ab??aba ababacaba	2

Минимальное остовное дерево

Ограничение времени	0.2 секунды
Ограничение памяти	10Mb
Ввод	стандартный ввод или <code>kruskal.in</code>
Вывод	стандартный вывод или <code>kruskal.out</code>

Дан неориентированный связный граф. Требуется найти вес минимального остовного дерева в этом графе.

- Вариант 1. С помощью алгоритма Прима.
- Вариант 2. С помощью алгоритма Крускала.
- Вариант 3. С помощью алгоритма Борувки.

Ваш номер варианта прописан в ведомости.

Формат ввода

Первая строка содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($1 \leq n \leq 20000$, $0 \leq m \leq 100000$).

Следующие m строк содержат описание ребер по одному на строке.

Ребро номер i описывается тремя натуральными числами b_i , e_i и w_i — номера концов ребра и его вес соответственно ($1 \leq b_i, e_i \leq n$, $0 \leq w_i \leq 100000$).

Формат вывода

Выведите единственное целое число - вес минимального остовного дерева.

Пример 1

Ввод	Вывод
------	-------

4 4 1 2 1 2 3 2 3 4 5 4 1 4	7
---	---

Сумма длин до остальных

Ограничение времени	0.1 секунда
Ограничение памяти	5Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Дано невзвешенное дерево. Расстоянием между двумя вершинами будем называть количество ребер в пути, соединяющем эти две вершины. Для каждой вершины определите сумму расстояний до всех остальных вершин. Время работы должно быть $O(n)$.

Формат ввода

В первой строке записано количество вершин $n \leq 10000$. Затем следует $n - 1$ строка, описывающая ребра дерева. Каждое ребро - это два различных целых числа - индексы вершин в диапазоне $[0, n-1]$. Индекс корня – 0. В каждом ребре родительской вершиной является та, чей номер меньше.

Формат вывода

Выход должен содержать n строк. В i -ой строке выводится сумма расстояний от i -ой вершины до всех остальных.

Пример

Ввод	Вывод
3 0 1 1 2	3 2 3