ПРИЛОЖЕНИЕ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ИМЕНИ. В.Ф. УТКИНА

Кафедра «Вычислительная и прикладная математика»

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

«Алгоритмы и структуры данных»

Направление подготовки

09.03.04 «Программная инженерия»

Направленность (профиль) подготовки

«Программная инженерия систем искусственного интеллекта»

Уровень подготовки – бакалавриат

Квалификация выпускника – бакалавр

Форма обучения – очная

Срок обучения – 4 года

Рязань

1 ОБЩИЕ ПОЛОЖЕНИЯ

Оценочные материалы – это совокупность учебно-методических материалов и процедур, предназначенных для оценки качества освоения обучающимися данной дисциплины как части основной образовательной программы.

Цель – оценить соответствие знаний, умений и уровня приобретенных компетенций, обучающихся целям и требованиям основной образовательной программы в ходе проведения текущего контроля и промежуточной аттестации.

Основная задача – обеспечить оценку уровня сформированности общекультурных и профессиональных компетенций и индикаторов их достижения, приобретаемых обучающимся в соответствии с этими требованиями.

Текущий контроль успеваемости и промежуточная аттестацияя проводятся с целью определения степени усвоения учебного материала, своевременного выявления и устранения недостатков в подготовке обучающихся и принятия необходимых мер по совершенствованию методики преподавания учебной дисциплины, организации работы обучающихся в ходе учебных занятий и оказания им индивидуальной помощи.

К контролю текущей успеваемости относятся проверка знаний, умений и навыков обучающихся на лабораторных работах, по результатам выполнения и защиты обучающимися индивидуальных заданий, курсовой работы, по результатам выполнения контрольных работ и тестов, по результатам проверки качества конспектов лекций и иных материалов.

В качестве оценочных средств на протяжении семестра используется устные и письменные ответы студентов на индивидуальные вопросы, письменное тестирование по теоретическим разделам курса. Дополнительным средством оценки знаний и умений студентов является отчет о выполнении лабораторных работ и их защита.

По итогам курса обучающиеся сдают зачет (с оценкой). Форма проведения – устный ответ с письменным подкреплением по утвержденным билетам, сформулированным с учетом содержания дисциплины. В билет включается два теоретических вопроса и одна задача. В процессе подготовки к устному ответу студент должен составить в письменном виде план ответа, включающий в себя определения, выводы формул, схемы и пр. Решение задачи предполагает выполнение некоторого кейса с использованием современных информационных технологий и программных средств.

2 ОПИСАНИЕ ПОКАЗАТЕЛЕЙ И КРИТЕРИЕВ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ

Сформированность каждой компетенции в рамках освоения данной дисциплины оценивается по трехуровневой шкале:

1) пороговый уровень является обязательным для всех обучающихся по завершении освоения дисциплины;

2) продвинутый уровень характеризуется превышением минимальных характеристик сформированности компетенций по завершении освоения дисциплины;

3) эталонный уровень характеризуется максимально возможной выраженностью компетенций и является важным качественным ориентиром для самосовершенствования.

Уровень освоения компетенций, формируемых дисциплиной

а) описание критериев и шкалы оценивания тестирования:

Шкала оценивания	**Критерий**
3 балла (эталонный уровень)	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 85 до 100%
2 балла (продвинутый уровень)	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 75 до 84%
1 балл (пороговый уровень)	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 60 до 74%
0 баллов	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 0 до 59%

б) описание критериев и шкалы оценивания теоретического вопроса:

Шкала оценивания	**Критерий**
3 балла	выставляется студенту, который дал полный ответ на вопрос, по-

(эталонный уровень)	казал глубокие систематизированные знания, смог привести примеры, ответил на дополнительные вопросы преподавателя.
2 балла (продвинутый уровень)	выставляется студенту, который дал полный ответ на вопрос, но на некоторые дополнительные вопросы преподавателя ответил только с помощью наводящих вопросов.
1 балл (пороговый уровень)	выставляется студенту, который дал неполный ответ на вопрос в билете и смог ответить на дополнительные вопросы только с помощью преподавателя.
0 баллов	выставляется студенту, который не смог ответить на вопрос

в) описание критериев и шкалы оценивания практического задания:

Шкала оценивания	Критерий
3 балла (эталонный уровень)	Задание решено верно
2 балла (продвинутый уровень)	Задание решено верно, но имеются технические неточности в выполнении
1 балл (пороговый уровень)	Задание решено верно, с дополнительными наводящими вопросами преподавателя
0 баллов	Задание не решено

На зачет выносится: 1 практическое задание и 2 теоретических вопроса. Студент может набрать максимум 9 баллов. Итоговый суммарный балл студента, полученный при прохождении промежуточной аттестации, переводится в традиционную форму по системе «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

Шкала оценивания	Критерий	
отлично (эталонный уровень)	8 – 9 баллов	Обязательным условием является выполнение всех предусмотренных в течение семестра практических заданий и лабораторных работ.
хорошо (продвинутый уровень)	6 – 7 баллов	
удовлетворительно (пороговый уровень)	4 – 5 баллов	
неудовлетворительно	0 – 3 баллов	Студент не выполнил всех предусмотренных в течение семестра текущих заданий

3 ПАСПОРТ ОЦЕНОЧНЫХ МАТЕРИАЛОВ ПО ДИСЦИПЛИНЕ

Контролируемые разделы (темы) дисциплины	Код контролируемой компетенции (или её части)	Наименование оценочного средства
Раздел 1. Введение. Анализ сложности алгоритмов	ОПК-6.1, ОПК-7.1, ОПК-7.2, ОПК-8.1, ПК-15.1, ПК-15.2	Зачет с оценкой
Раздел 2. Алгоритмы сортировки и поиска данных	ОПК-6.1, ОПК-7.1, ОПК-7.2, ОПК-8.1, ПК-15.1, ПК-15.2	Зачет с оценкой
Раздел 3. Фундаментальные структуры данных	ОПК-6.1, ОПК-7.1, ОПК-7.2, ОПК-8.1, ПК-15.1, ПК-15.2	Зачет с оценкой
Раздел 4. Древовидные структуры данных	ОПК-6.1, ОПК-7.1, ОПК-7.2, ОПК-8.1, ПК-15.1, ПК-15.2	Зачет с оценкой
Раздел 5. Алгоритмы на графах	ОПК-6.1, ОПК-7.1, ОПК-7.2, ОПК-8.1, ПК-15.1, ПК-15.2	Зачет с оценкой

Раздел 6. Алгоритмы на строках	ОПК-6.1, ОПК-7.1, ОПК-7.2, ОПК-8.1, ПК-15.1, ПК-15.2	
Раздел 7. Промежуточная аттестация	ОПК-6.1, ОПК-7.1, ОПК-7.2, ОПК-8.1, ПК-15.1, ПК-15.2	
Раздел 8. Курсовая работа	ОПК-6.1, ОПК-7.1, ОПК-7.2, ОПК-8.1, ПК-15.1, ПК-15.2	Защита курсовой работы

4 ТИПОВЫЕ КОНТРОЛЬНЫЕ ЗАДАНИЯ ИЛИ ИНЫЕ МАТЕРИАЛЫ

4.1 Промежуточная аттестация (зачет с оценкой)

ОПК-6: Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов;
ОПК-6.1. Выполняет разработку алгоритмического и программного обеспечения для решения прикладных задач

а) типовые тестовые вопросы закрытого типа

1. Какой из перечисленных алгоритмов сортировки имеет в среднем случае временную сложность $O(n\log n)$ и требует $O(1)$ дополнительной памяти?
   Сортировка пузырьком
   Быстрая сортировка (без оптимизации по памяти)
   Пирамидальная сортировка
   Сортировка слиянием (классическая рекурсивная)
2. Что из перечисленного является основным недостатком рекурсивной реализации алгоритма для обработки больших объёмов данных (например, глубины графа > 10^5) в Python?
   Сложность написания кода
   Ограничение на глубину рекурсии и риск переполнения стека вызовов
   Невозможность использования циклов
   Плохая читаемость
3. Какая структура данных из стандартной библиотеки Python наиболее подходит для реализации очереди с двусторонним доступом (deque) в высоконагруженном приложении?
   list с pop(0)
   collections.deque
   queue.Queue
   tuple
4. В упорядоченном массиве из 1 миллиона элементов бинарный поиск выполнит не более $log_2(10^6)$ сравнений. Это примерно:
   10
   20
   100
   1 000 000
5. Для поиска кратчайшего пути во взвешенном графе с неотрицательными весами в практической задаче (тысячи вершин и рёбер) следует использовать:
   Алгоритм Флойда–Уоршелла
   Поиск в глубину
   Алгоритм Дейкстры с приоритетной очередью
   Топологическую сортировку
6. Какая проблема чаще возникает при практическом использовании связного списка, реализованного на Python через ссылки на объекты?
   Невозможность удаления элементов
   Высокое потребление памяти на хранение ссылок и фрагментация кучи
   Отсутствие операций вставки
   Медленный доступ по индексу, но это преимущество

7. Что характерно для динамической реализации структур данных?
   Использование адресных переменных (указателей) для связывания элементов структуры
   Возможность выделения памяти для элементов структуры во время выполнения программы
   Использование массивов как основу реализации
   Распределение памяти под элементы структуры во время компиляции программы
8. Какие утверждения относительно динамической реализации списков являются правильными?
   Каждому элементу списка во время выполнения программы выделяется своя область памяти
   Каждый элемент списка имеет специальное поле с адресом следующего элемента
   Логический порядок следования элементов в списке может не совпадать с физическим размещением элементов в памяти
   Максимальное число элементов в списке должно быть известно заранее
9. Что означает понятие «инвариант цикла» при разработке алгоритмов?
   Условие, которое всегда ложно
   Условие, истинное до входа в цикл и после каждой итерации, помогающее доказать корректность
   Переменная, не меняющая значения
   Запрет на использование break
10. Алгоритм сортировки слиянием с классической рекурсивной реализацией на Python требует **дополнительной** памяти порядка:
    O(1)
    O(logn)
    O(n)
    O(n2)
11. В Python при обработке большого файла (> 10 ГБ) построчное чтение через генератор вместо загрузки всего файла в список – это пример:
    Преждевременной оптимизации
    Использования ленивых вычислений для экономии памяти
    Нарушения принципов ООП
    Увеличения временной сложности до O(n2)
12. Для реализации ассоциативного массива (ключ → значение) с быстрым доступом, вставкой и удалением в среднем за $O(1)$ в Python практическим выбором является:
    list кортежей
    dict
    set
    collections.namedtuple
13. Какой подход повышает практическую пригодность алгоритма при работе с пользовательским вводом?
    Игнорирование любых ошибок
    Проверка входных данных и выбрасывание исключений с понятным сообщением
    Преобразование всех данных к целым числам
    Использование глобальных переменных для состояния
14. В Python операция list.pop(0) имеет временную сложность:
    O(1)
    O(logn)
    O(n)
    O(2n)
15. Что означает, что алгоритм сортировки является «устойчивым»?
    Он не падает при больших объёмах данных
    Он сохраняет относительный порядок равных элементов
    Он использует только O(1) памяти
    Он работает за O(nlogn) в любом случае
16. Для задачи вычисления n-го числа Фибоначчи (n=106) практически применимый по времени и памяти подход:
    Рекурсивное вычисление

Динамическое программирование с массивом размера n
Итеративный алгоритм с двумя переменными
Формула Бине с плавающей точкой

17. Какой алгоритм из нижеперечисленных будет самым производительным, если дан уже отсортированный массив?
    сортировка слиянием
    сортировка вставками
    быстрая сортировка
    пирамидальная сортировка

б) типовые тестовые вопросы открытого типа

1. Дан массив из 10^6 целых чисел. Необходимо найти 10 наибольших элементов (не обязательно отсортированных). Предложите алгоритм с оценкой времени работы и памяти. Почему полная сортировка всего массива – не лучший практический выбор?
   Ожидаемый ответ: использовать алгоритм выбора (quickselect) с частичной сортировкой или кучу (min-heap) размера 10. Сортировка всего массива даст O(n log n) против O(n log 10) ≈ O(n). Память: O(1) или O(10).
2. Напишите итеративную версию функции для обхода бинарного дерева в глубину (`in-order`), используя явный стек. Объясните, в каких случаях итеративная реализация предпочтительнее рекурсивной с точки зрения практической пригодности.
   Ожидаемый ответ: код с `while` и `list` в качестве стека. Рекурсия опасна при большой глубине дерева из-за ограничения `sys.setrecursionlimit` и переполнения стека Python.
3. Разработайте функцию на Python, которая принимает список целых чисел и возвращает новый список, содержащий только уникальные элементы, сохраняя порядок первого вхождения. Приведите примеры для пустого списка, списка из одного элемента, списка со всеми дубликатами. Какие структуры данных вы используете и почему?
   Ожидаемый ответ: использовать `list + set` (`OrderedDict` или обычный `dict`). Обойти граничные случаи. Объяснить, что `set` даёт O(1) проверку наличия.
4. В системе телеметрии приходят события с временными метками. Нужно поддерживать операции: добавление события, удаление всех событий старше 5 минут, получение последнего события. Какие структуры данных предложите? Обоснуйте сложность операций.
   Ожидаемый ответ: коллекция deque (для быстрого добавления и удаления с концов) + отдельный список для таймстампов. Или двунаправленная очередь с хранением (timestamp, data) + периодическая очистка от старых.
5. Объясните недостатки наивного алгоритма поиска подстроки (по одному символу). Какой алгоритм предложите для поиска в большом тексте (10^9 символов) короткого шаблона (длина 10-20) с гарантией линейного времени?
   Ожидаемый ответ: наивный – O(n*m). KMP дает O(n+m). KMP хорош для коротких шаблонов и потоковых данных.
6. В Python операция append() для списка работает в среднем за O(1). Объясните механизм изменения размера динамического массива. Почему с практической точки зрения лучше заранее выделить память [None]*N, если размер известен?
   Ожидаемый ответ: список перевыделяет память с коэффициентом (обычно 1.125), копируя элементы. Амортизированная сложность O(1), но каждый перевыделение – дорогое. Предварительное выделение памяти позволяет избежать лишних копирований.
7. Программа обрабатывает список из 100 элементов. Начинающий разработчик заменил все стандартные for на map и filter с лямбдами, полагая, что это ускорит код. Правильно ли это с точки зрения практической пригодности?
   Ожидаемый ответ: неверно – преждевременная оптимизация усложняет читаемость. Для 100 элементов разница незаметна. Главное – ясность и поддерживаемость.
8. Что произойдёт с производительностью, если в цикле к строке s = s + ch (накопление строки) для 100 000 итераций? Предложите исправление с оценкой сложности.
   Ожидаемый ответ: каждая конкатенация создаёт новую строку за O(len(s)), итого $O(n^2)$. Исправить: использовать `list.append(ch) + ''.join(list)` за O(n).
9. Какое ограничение применимости алгоритма Дейкстры? **Ожидаемый ответ**: алгоритм Дейкстры не работает с отрицательными ребрами.

ОПК-7: Способен применять в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой;
ОПК-7.1. Демонстрирует знание основных концепций, принципов, теорий и фактов, связанных с информатикой
ОПК-7.2. Применяет в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой

1. Если алгоритм работает за время $\Theta(n\log n)$ в среднем и за $\Theta(n^2)$ в худшем случае, что это означает практически?

Время выполнения всегда не более $n\log n$

На случайных данных производительность будет близка к $n\log n$, но возможны плохие входы с квадратичным временем

Время выполнения всегда не менее $n\log n$

Алгоритм не может быть использован на больших данных

2.Определите асимптотическую сложность алгоритма

```
def modif_arr(arr):
    n = len(arr)
    for num in arr:
        num =  num ** 2
    ans = 0
    for i in range(n):
        for j in range(i, n):
            ans += arr[i] + arr[j]
    return arr
```

O(n)

$O(n^2)$

O(nlogn)

O(logn)

$O(2^n)$

$O(n^2/2)$

среди перечисленных вариантов нет верного

3.Определите асимптотическую сложность алгоритма

```
def f(n):
    if n == 0:
        return 1
    return n * f(n - 1)
```

O(n)

$O(n^2)$

O(nlogn)

O(logn)

$O(2^n)$

$O(n^2/2)$

среди перечисленных вариантов нет верного

4.Определите асимптотическую сложность алгоритма

```
n = int(input())
i = 1
while i <= n:
    print(i)
    i = i * 2
```

O(n)

$O(n^2)$
O(nlogn)
O(logn)
$O(2^n)$
$O(n^2/2)$
среди перечисленных вариантов нет верного

5.Какая структура данных используется для диспетчеризации задач в операционной системе?
Двоичное дерево
Таблица
Список
Стек
Очередь

6. Формула, определяющая количество сравнений N элементов массива при быстрой сортировке:
N^2
N-1
$N\log_2(N)$
N(N-1)
N(N-1)/2

7. Основное требование, предъявляемое к массиву для возможности выполнения двоичного поиска:
упорядоченность
малый размер
неупорядоченность
большой размер
нет особых требований

8. Имеется идеально сбалансированное двоичное дерево поиска, содержащее целые числа. Просмотр дерева даёт следующий результат: 2, 4, 6, 8, 10, 12, 14. Какой способ просмотра дерева использовался?
Симметричный
Прямой
Обратный
Поузловой

9.Какова асимптотика удаления последнего элемента из очереди, состоящей из *n* элементов?
O(n)
$O(n^2)$
O(nlogn)
O(logn)
O(1)
$O(2^n)$
$O(n^2/2)$
среди перечисленных вариантов нет верного

10.Какова асимптотика удаления последнего элемента из односвязного списка, состоящей из *n* элементов?
O(n)
$O(n^2)$
O(nlogn)
O(logn)
O(1)
$O(2^n)$
$O(n^2/2)$
среди перечисленных вариантов нет верного

11.Какова асимптотика добавления элемента в начало дека, состоящего из *n* элементов?
O(n)
$O(n^2)$
O(nlogn)
O(1)
O(logn)

$O(2^n)$
$O(n^2/2)$
среди перечисленных вариантов нет верного

12.Какова асимптотика добавления элемента в двоичное дерево поиска, состоящего из *n* элементов?

O(n)
$O(n^2)$
O(nlogn)
O(1)
O(logn)
$O(2^n)$
$O(n^2/2)$
среди перечисленных вариантов нет верного

13.Чем отличается кольцевой список от линейного?

В кольцевом списке последний элемент является одновременно и первым
В кольцевом списке указатель последнего элемента пустой
В кольцевых списках последнего элемента нет
В кольцевом списке указатель последнего элемента не пустой

14.В каких направлениях можно перемещаться в кольцевом двунаправленном списке?

Влево и вправо
Влево
Вправо

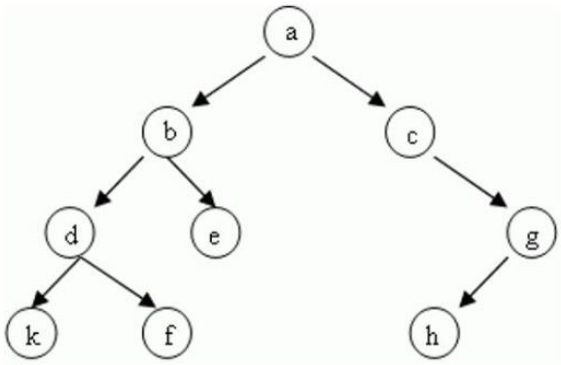
15.Дерево называется полным бинарным, если степень исходов вершин равна:

2 или 0
2
M или 0
M

16. В двоичном дереве поиска при удалении узла, содержащего два дочерних узла, узел заменяется на элемент, являющийся _____, после чего сам элемент рекурсивно удаляется из дерева.

минимальным элементом из правого поддерева
максимальным элементом из левого поддерева
минимальным элементом из левого поддерева
максимальным элементом из правого поддерева
максимальным элементом среди всех имеющихся листьев
минимальным элементом среди всех имеющихся листьев

17.Какое из следующих перечислений вершин бинарного дерева ***T***, изображенного на рисунке, представляет его обход в обратном (суффиксном) порядке?



abefdcghk
kdbfehgca
abdkfecgh
kdfebhgca
kfdebhgca

18.Какие утверждения справедливы относительно недвоичных деревьев?

вершины могут иметь любое число потомков
недвоичное дерево можно описать с помощью двоичного

недвоичное дерево нельзя описать с помощью двоичного

19.Свойство дерева: количество рёбер в любом связном ациклическом графе равно:
Количеству вершин
Количеству вершин минус 1
Количеству вершин плюс 1
Количеству листьев

20.Стандартным способом устранения рекурсии при поиске в глубину является использование:
массива
очереди;
стека;
циклического списка

21.При поиске в ширину используется:
массив
очередь
стек
циклический список

22.Граф – это структура данных, реализующая отношение
«многие ко многим»
«многие к одному»
«один ко многим»

23.Если граф задан матрицей смежности и не имеет петель, то на главной диагонали у него:
всегда стоят нули
всегда стоят единицы
обязательно должна быть хотя бы одна единица
обязательно должен быть хотя бы один ноль
главная диагональ не определяет наличие петель в графе

24.Выберите верные утверждения
Матрица смежности неориентированного графа является симметричной
Матрица смежности неориентированного графа меняется при транспонировании
Матрица смежности неориентированного графа не меняется при транспонировании
Матрица смежности неориентированного графа не является симметричной

25.Постройте, начиная с вершины a, обход графа G=(V, E) в глубину, в котором соседи каждой вершины рассматриваются в порядке, определенном ее списком смежности. Какая из следующих нумераций вершин ему соответствует? Неориентированный граф G=(V, E) задан с помощью списков смежности:

L_a: c, d, b L_b: a, f, g L_c: a, d, e
L_d: a, c, e L_e: c, d L_f: b
L_g: b, i, h L_h: g, i L_i: g, h

Вариант 1

a	b	c	d	e	f	g	h	i
1	2	7	8	9	3	4	5	6

Вариант 2

a	b	c	d	e	f	g	h	i
1	5	2	3	4	6	7	9	8

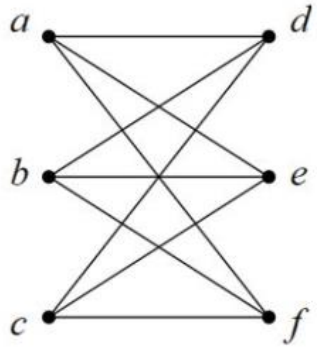
Вариант 3

a	b	c	d	e	f	g	h	i
1	5	3	2	4	6	7	9	8

Вариант 4

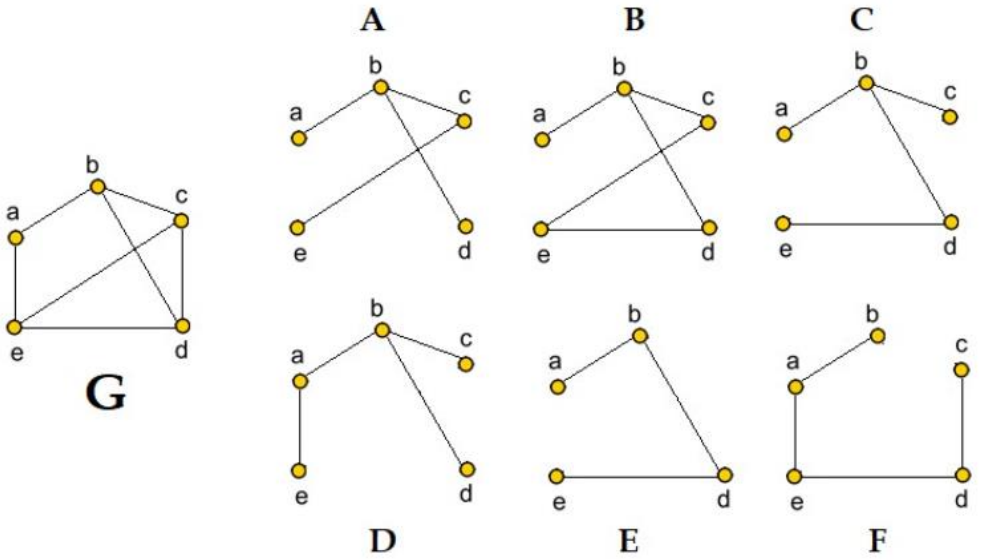
a	b	c	d	e	f	g	h	i
1	5	2	3	4	6	7	8	9

26.Что из приведенного ниже является простым путем в графе, изображенном на рисунке:



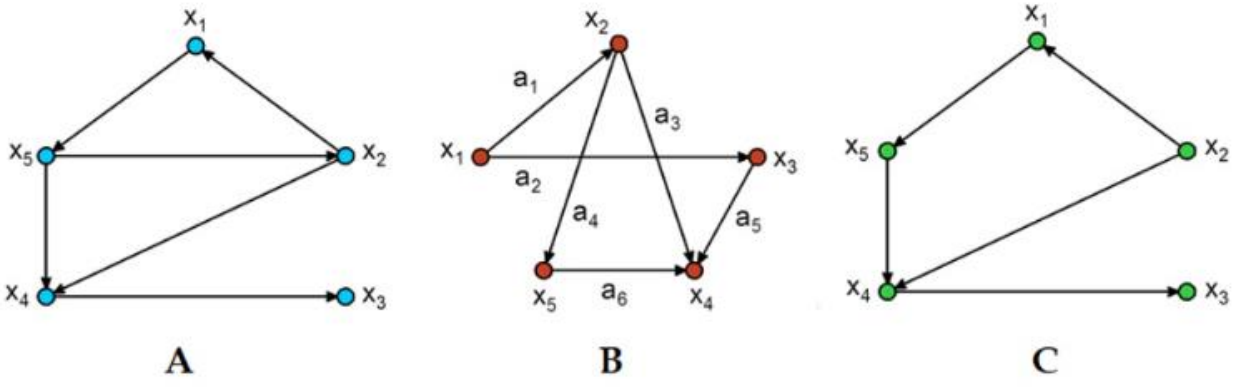
aecfbdafc
aebfcd
aecbd
aecdaec
aebecfbd

27.Дан граф G. Какие из представленных графов являются остовными подграфами графа G?



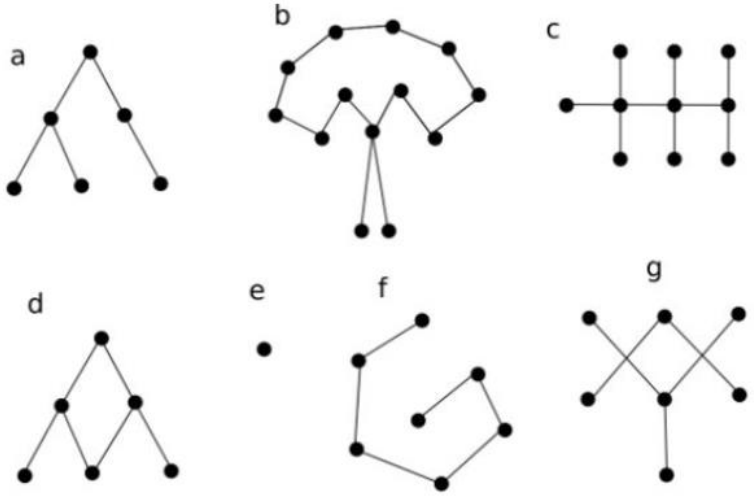
A
B
C
D
E
F
G

28.Какие из приведенных на рисунке графов являются двудольными?



A
B
C

29. Какие из изображенных на рисунке графов являются деревьями?



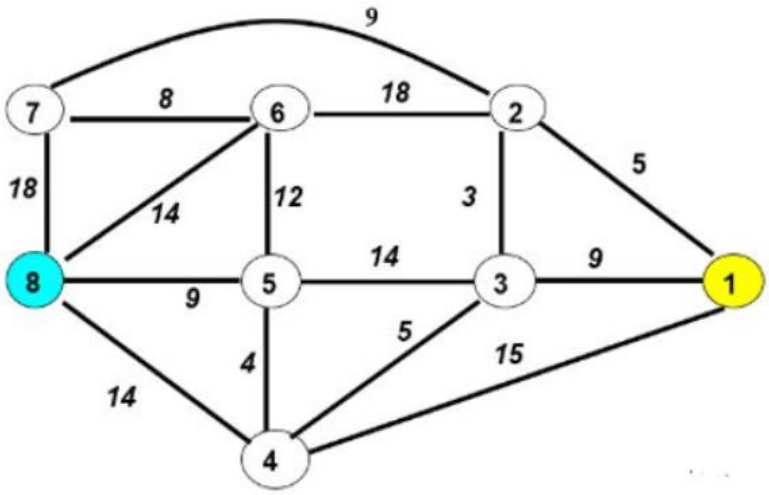
a
b
c
d
e
f
g

30. Алгоритм Дейкстры основан на:
парадигме «разделяй и властвуй»
динамическом программировании
жадном подходе
поиске с возвратом

б) типовые тестовые вопросы открытого типа

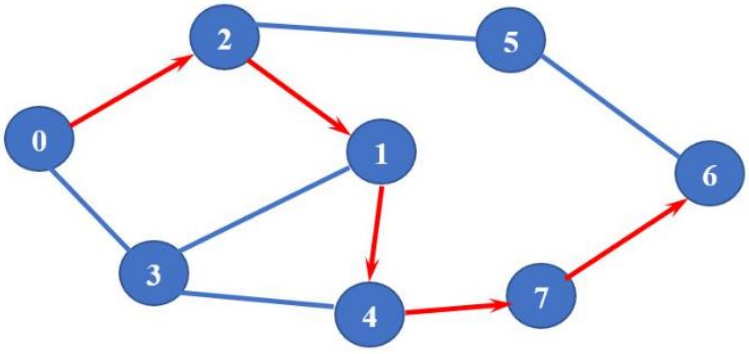
1. Массив, где N – число элементов массива сортируется “пузырьковым” методом. За сколько проходов по массиву самый “лёгкий” элемент в массиве окажется вверху? **Ответ**: за 1 проход.
2. Производится пузырьковая сортировка массива из 6 элементов, причём массив упорядочен в обратном порядке. Сколько будет выполнено перестановок? ***Ответ***: 30.
3. Записать выражение в постфиксной форме (в записи ответа токены разделите пробелами): **(32 – 6 * 5) * (2 * 6 / 4 + 1) / (7 – 10 / 2)**. ***Ответ***: 32 6 5 * - 2 6 * 4 / 1 + * 7 10 2 / - /
4. Определите значение выражения, записанного в постфиксной форме: 6 3 * 18 3 / – 2 5 * 18 3 / 2 * + * 12 / 5 * 4 / ***Ответ***: 27,5
5. Сколько указателей используется в односвязном кольцевом списке? ***Ответ***: один.
6. В чём суть бинарного поиска? ***Ответ***: нахождение элемента массива осуществляется путём деления массива пополам каждый раз, пока элемент не найден.
7. В чем суть правила обхода дерева в обратном направлении? ***Ответ:*** сначала обрабатывается левое поддерево, потом - правое поддерево, потом - корень поддерева.
8. Какие ситуации возможны при удалении вершины из дерева поиска? ***Ответ:*** удаляемая вершина не имеет потомков, удаляемая вершина имеет только одного потомка, удаляемая вершина имеет двух потомков.
9. Из элементов списка: **[12, 8, 16, 14, 19, 5, 9, 15, 7, 13, 1]** было построено двоичное дерево поиска. В каком порядке будут выведены элементы дерева при обходе в ширину (укажите элементы через пробел)? ***Ответ:*** 12 8 16 5 9 14 19 1 7 13 15
10. Сколько вершин в полном бинарном дереве высоты 5? ***Ответ:*** 63
11. Какие операции характерны при использовании очереди? ***Ответ:*** добавление элемента в конец очереди, удаление элемента из начала очереди.
12. Какие поля должен содержать каждый элемент динамической очереди? ***Ответ:*** поле-указатель с адресом соседнего элемента.

13. Какие возможны способы реализации списков? ***Ответ:*** динамически в виде однонаправленного списка, динамически в виде двунаправленного списка, статически на основе адресных указателей элементов массивов.
14. Элемент дерева, который не ссылается на другие, называется: ***Ответ***: листом.
15. Элемент дерева, на который не ссылаются другие, называется: ***Ответ***: корнем.
16. Высотой дерева называется: ***Ответ***: максимальная длина пути от корня до листа.
17. Узлам (или вершинам) графа можно сопоставить: ***Ответ***: объекты.
18. Какие существуют способы представления графа? ***Ответ***: графическое изображение, матрица смежности, указание списка вершин и списка ребер(дуг), матрица инцидентности.
19. В чем состоит общая идея поиска в глубину графах? ***Ответ***: поиск начинаем с некоторой фиксированной вершины v0. Затем выбираем произвольную вершину u, смежную с v0, и повторяем просмотр от u. Предположим, что находимся в некоторой вершине v. Если существует ещё не просмотренная вершина u, u-v, то рассматриваем её, затем продолжаем поиск с нее. Если не просмотренной вершины, смежной с v, не существует, то возвращаемся в вершину, из которой попали в v, и продолжаем поиск (если v=v0, то поиск закончен).
20. Найдите кратчайший путь между вершинами 1 и 8, используя алгоритм Дейкстры.



Ответ: 26

21. Обход графа, изображенного на рисунке, осуществлялся по красным стрелкам в направлении от вершины № 0 до вершины № 6. Запишите содержимое массива предков, разделяя элементы этого массива одним пробелом. Полагаем, что если у вершины нет предка, то в массиве предков записано число -1.



Ответ: -1 2 0 -1 1 -1 7 4

22. Определите число компонент связности в неориентированном графе G=(V, E), где V={1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, E={(2,5), (6,9), (4,7), (6,3), (5,9), (1,10), (5,3), (7,8), (4,8)}? ***Ответ***: 3
23. Записать префикс-функцию от строки "**bacbabacbac**", разделяя значения одним пробелом. Формат ответа: 0 0 1 2 0 0 0 1 2 3 4 5 1 2 0. ***Ответ***: 0 0 0 1 2 1 2 3 4 5 3
24. Записать *z*-функцию от строки "**acbacbacaac**", разделяя значения одним пробелом. Считаем, что z[0] = 0, например: 0 0 3 0 1 1 1 2 0. ***Ответ***: 0 0 0 5 0 0 2 0 1 2 0

ОПК-8: Способен осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий.
ОПК-8.1. Владеет средствами поиска, хранения, обработки и анализа информации из различных источников и баз данных

а) типовые тестовые вопросы закрытого типа

1. Два отсортированных списка целых чисел (каждый по 10^6 элементов) из двух разных баз данных необходимо объединить в один отсортированный список. Какой алгоритм обеспечит минимальную временную сложность?
   - Добавить все элементы в один список и вызвать list.sort()
   - **Использовать алгоритм слияния двух отсортированных массивов (merge)**
   - Вставить элементы второго списка в первый бинарным поиском
   - Использовать сортировку пузырьком
2. В базе данных хранится массив записей, отсортированных по ключу. Операция вставки новой записи выполняется редко, а поиск – очень часто. Какую структуру данных предпочтительнее использовать для хранения, чтобы поиск оставался эффективным?
   - Связный список
   - Дек (collections.deque)
   - **Отсортированный список с бинарным поиском (и вставкой за O(n))**
   - Хеш-таблица без порядка
3. Система обрабатывает задания из очереди с приоритетами. Какая структура данных из изученных обеспечивает добавление задания и извлечение задания с максимальным приоритетом за O(log n)?
   - Стек
   - Обычная очередь (FIFO)
   - Дек
   - **Куча (heap)**
4. Требуется найти все вхождения короткого паттерна (до 100 символов) в большом тексте (10^7 символов). Какой алгоритм обработки строк обеспечит линейное время O(n+m) в худшем случае?
   - Наивный поиск (сравнение посимвольно)
   - **Алгоритм Кнута-Морриса-Пратта (KMP)**
   - Бинарный поиск
   - Сортировка строк
5. В социальной сети для поиска всех друзей пользователя на расстоянии не более 3 шагов (в графе дружбы) наиболее подходящим алгоритмом является:
   - Поиск в глубину (DFS) без ограничений
   - **Поиск в ширину (BFS) с ограничением глубины**
   - Топологическая сортировка
   - Алгоритм Дейкстры
6. В приложении карт для поиска кратчайшего пути между двумя точками с неотрицательными весами рёбер (расстояниями) используется:
   - Поиск в ширину (BFS)
   - **Алгоритм Дейкстры с приоритетной очередью (кучей)**
   - Алгоритм Флойда-Уоршелла
   - Топологическая сортировка
7. Бинарный поиск в отсортированном массиве из 1 000 000 элементов в худшем случае выполняет примерно:
   - 10 сравнений
   - **20 сравнений**
   - 100 сравнений
   - 1 000 000 сравнений
8. Для хранения графа с миллионом вершин и десятью миллионами рёбер (разреженный граф) наиболее экономичной по памяти структурой является:
   - Матрица смежности (миллион × миллион)

Список рёбер (массив пар)
Список смежности (для каждой вершины – список соседей)
Матрица инцидентности

9. Какое утверждение о стеке и очереди является верным:
   Стек работает по принципу LIFO (последний вошёл – первый вышел), очередь – FIFO
   Стек работает по принципу FIFO (первый вошёл – первый вышел), очередь – LIFO
   Обе структуры работают по принципу LIFO
   Обе структуры работают по принципу FIFO
10. Какой алгоритм сортировки гарантирует сложность O(n log n) в худшем случае и использует O(1) дополнительной памяти?
    Быстрая сортировка (quicksort)
    Сортировка слиянием (mergesort)
    Сортировка кучей (heapsort)
    Сортировка пузырьком

б) типовые тестовые вопросы открытого типа

1. У вас есть два отсортированных списка чисел (каждый по миллиону элементов) из двух разных баз данных. Требуется получить объединённый отсортированный список без дубликатов. Какой алгоритм использовать? Обоснуйте выбор. ***Ожидаемый ответ***: слияние двух отсортированных списков (алгоритм `merge` из сортировки слиянием) за O(n+m). Поддерживаются два указателя, на каждом шаге выбирается меньший элемент, дубликаты пропускаются.
2. В систему анализа поступает непрерывный поток целых чисел. Требуется в любой момент времени быстро (за O(log n)) получать максимум потока. Какую структуру данных для этого нужно использовать? ***Ожидаемый ответ***: куча с поддержкой макимума.
3. У вас есть база данных авиарейсов: (*город_отправления*, *город_назначения*, *время_перелёта*). Необходимо для заданной пары городов найти минимальное время пути с учётом возможных пересадок. Какую структуру данных для графа вы выберете? Какой алгоритм примените? Объясните, как обрабатывать случай, когда рейсы могут иметь разную стоимость. ***Ожидаемый ответ***: Список смежности (adjacency list) – `dict: город -> list[(город, время)]`. Алгоритм Дейкстры с приоритетной очередью (куча). Если веса неотрицательные – Дейкстра; если есть отрицательные – алгоритм Форда-Беллмана.
4. В системе сборки программного обеспечения модули зависят друг от друга (граф зависимостей). Требуется определить порядок сборки, чтобы модуль собирался только после всех его зависимостей. Какой алгоритм на графе это позволяет сделать? ***Ожидаемый ответ***: топологическая сортировка.
5. В систему мониторинга поступают данные о температуре каждую секунду. Нужно вычислять скользящее среднее за последние 60 секунд и одновременно выявлять выбросы (значения, отклоняющиеся от среднего более чем на 3 сигмы). Какую структуру данных использовать для хранения последних 60 значений? Почему `list` не подходит? ***Ожидаемый ответ***: `collections.deque` с `maxlen`=60. Добавление O(1), удаление старых автоматически. Список с `pop(0)` – O(n). Среднее вычисляется суммированием (можно хранить сумму и обновлять при добавлении/удалении).
6. В социальной сети пользователи хранятся в БД с таблицей друзей (*user_id*, *friend_id*). Найдите всех друзей пользователя на расстоянии не более 2 шагов (друзья и друзья друзей), исключая самого пользователя. Какой обход графа выберете и почему? Оцените сложность по времени и памяти. ***Ожидаемый ответ***: BFS с ограничением глубины 2. Используем очередь. Множество `visited` для предотвращения зацикливания. BFS гарантирует нахождение всех вершин на заданном расстоянии. Сложность O(V+E).
7. Система обработки задач имеет задания с приоритетами (1 – наивысший, 5 – низший). Задания поступают в произвольном порядке. Нужно всегда обрабатывать задание с наивысшим приоритетом (и наименьшим временем поступления среди одинаковых приоритетов). Какую структуру данных использовать и почему? *Ожидаемый ответ*: куча (`heapq`) с кортежем (*приоритет*, *время_поступления*, *задача*). В Python heapq реализует `min`-кучу. Операции добавления и извлечения – O(log n).

8. В базе данных имеется текстовое поле длиной до 10^6 символов. Необходимо быстро проверить, встречается ли в нём заданный короткий шаблон (до 100 символов). Наивный алгоритм будет слишком медленным. Какой алгоритм поиска подстроки вы предложите и почему? ***Ожидаемый ответ***: алгоритм Кнута-Морриса-Пратта (KMP) – линейное время O(n+m). Или Бойер-Мур (лучше на практике для больших текстов). Рекомендуется KMP для гарантированной линейной сложности.
9. Матрица смежности графа с миллионом вершин имеет всего около миллиона рёбер. Хранить полную матрицу (10^{12} элементов) нереально. Какую структуру данных вы выберете для хранения такого графа в памяти? ***Ожидаемый ответ***: Список смежности в виде `dict: вершина -> set/список соседей`.
10. Сервер генерирует логи ошибок. Нужно хранить последние 1000 сообщений об ошибках в памяти для быстрого отображения и анализа. При этом количество ошибок может быть огромным, но хранить все нельзя. Какую структуру данных выбрать? ***Ожидаемый ответ***: `collections.deque` с `maxlen=1000`. Добавление в конец O(1), автоматическое удаление старых.

ПК-15: Способен применять язык программирования Python для решения задач в области ИИ (PL-1)
ПК-15.1. Разрабатывает и отлаживает прикладные решения разной сложности и для разного круга конечных пользователей с использованием языка программирования Python, тестирует, испытывает и оценивает качество таких решений
ПК-15.2. Осуществляет выбор инструментов разработки на Python, приемлемых для создания прикладной системы обработки научных данных, машинного обучения и визуализации с заданными требованиями.

а) типовые тестовые вопросы закрытого типа

1. При подготовке обучающей выборки для модели машинного обучения требуется отсортировать 10 миллионов объектов по значению признака. Какой алгоритм сортировки в Python наиболее эффективен с точки зрения производительности и практичности?
   Пузырьковая сортировка
   Быстрая сортировка, реализованная вручную
   Встроенная list.sort() (Timsort)
   Сортировка слиянием, написанная на Python?
2. В задаче поиска ближайшего соседа среди одномерных эмбеддингов (отсортированный список из 10^6 чисел) требуется найти элемент, ближайший к заданному числу. Какой алгоритм следует применить?
   Линейный поиск
   Бинарный поиск (bisect модуль)
   Поиск в ширину
   Сортировка кучей
3. При обучении нейронной сети с огромным количеством классов (100 000) необходимо быстро получать 5 классов с наибольшими вероятностями на выходе `softmax`. Какую структуру данных следует использовать?
   Стек
   Связный список
   Дек
   Min-кучу размера 5
4. В алгоритме случайного леса каждое дерево решений представляет собой бинарное дерево. Какое свойство обеспечивает эффективный поиск решения для нового примера?
   Высота дерева логарифмическая (при сбалансированности)
   Все листья находятся на одном уровне
   Каждый узел имеет не более 10 потомков
   Дерево хранится в виде списка смежности
5. При поиске кратчайшего пути в невзвешенном графе (например, решение головоломки «пятнашки») гарантированно находит решение с минимальным числом шагов?
   Поиск в глубину (DFS)

Поиск в ширину (BFS)
Жадный поиск
Алгоритм Дейкстры (при положительных весах)

6. При реализации LSTM или GRU для обработки временных рядов часто требуется окно последних N входных векторов. Какая структура данных Python наиболее эффективна для хранения такого скользящего окна?
   list с pop(0)
   numpy.array с изменяемым размером
   set
   collections.deque с maxlen
7. При предобработке текста для модели NLP (удаление стоп-слов, стемминг) часто требуется проверять, является ли слово стоп-словом. Какая структура данных обеспечивает проверку за O(1) в среднем?
   Список стоп-слов
   Дерево поиска
   Множество (set)
   Очередь
8. При построении графа знаний (онтологии) с отношениями «является подклассом» необходимо проверить отсутствие циклов и определить порядок наследования. Для этого используется:
   Поиск в ширину (BFS)
   Топологическая сортировка
   Алгоритм Дейкстры
   Сортировка слиянием
9. При настройке гиперпараметра, имеющего монотонное влияние на метрику качества (например, *learning rate*), для быстрого нахождения оптимального значения можно использовать:
   Линейный поиск
   Бинарный поиск (поиск по монотонной функции)
   Сортировку кучей
   Поиск в глубину
10. В экспертных системах для хранения и поиска правил вида «ЕСЛИ условие ТО действие» часто используется структура, обеспечивающая быстрый поиск по левой части. Что это за структура?
    Хеш-таблица
    Стек
    Дерево решений (из базовых – дерево)
    Очередь

б) типовые тестовые вопросы открытого типа

1. При подготовке данных для обучения модели градиентного бустинга (например, CatBoost) требуется отсортировать таблицу из 50 миллионов строк по числовому признаку «возраст». Какую сортировку в Python вы выберете и почему? Оцените сложность по времени и памяти. ***Ожидаемый ответ***: Использовать встроенную list.sort() или sorted() (Timsort), так как она написана на C и работает быстрее любой ручной реализации. Сложность O(n log n), память O(n) в худшем случае (Timsort требует дополнительной памяти)..
2. При автоматическом подборе гиперпараметров часто требуется откат к предыдущей конфигурации. Какую структурой данных вы предложите для хранения гиперпараметров? ***Ожидаемый ответ***: стек.
3. Нейросеть выдала 1 миллион оценок (score) для товаров. Нужно быстро отобрать 10 товаров с наибольшими оценками, не сортируя весь список. Какую структуру данных вы предложите? ***Ожидаемый ответ***: Min-кучу размера 10. Она позволяет держать 10 лучших элементов, обрабатывая все данные за O(n log 10).
4. В задаче поиска кратчайшего пути для робота на поле (без весов) используется алгоритм BFS. Какую структуру данных нужно использовать для очереди обрабатываемых клеток? ***Ожидаемый ответ***: Очередь (FIFO – первым пришёл, первым ушёл). Она гарантирует, что сначала будут обработаны клетки, ближайшие к старту.
5. При обработке временных рядов (например, курса акций) для LSTM-модели нужно хранить последние 60 значений. Какая структура данных из Python автоматически удаляет старые

значения при добавлении новых. ***Ожидаемый ответ***: collections.deque с параметром maxlen=60. Она добавляет и удаляет с концов за O(1).

6. В системе рекомендаций нужно быстро получить эмбеддинг (вектор) пользователя по его ID. Какая структура данных в Python даёт доступ к данным по ключу в среднем за O(1)? ***Ожидаемый ответ***: Словарь (dict). Он использует хеш-таблицу.
7. В графе знаний классы связаны отношением «является подклассом». Нужно расположить классы так, чтобы каждый класс шёл после своих родителей. Как называется этот алгоритм? ***Ожидаемый ответ***: Топологическая сортировка.
8. В задаче поиска кратчайшего пути в лабиринте для робота используется BFS. Почему DFS (поиск в глубину) не гарантирует кратчайший путь? ***Ожидаемый ответ***: DFS сначала уходит вглубь по одной ветке и может найти длинный путь, даже если рядом есть короткий. BFS проверяет все пути одновременно, увеличивая «волну» от старта.
9. В задаче построения маршрута на карте (дороги имеют разную длину) нужно найти самый короткий путь. Какой алгоритм на графе для этого используется? ***Ожидаемый ответ***: алгоритм Дейкстры. Он работает с взвешенными графами (неотрицательные веса).
10. В алгоритме A* (поиск пути с эвристикой) нужно на каждом шаге выбирать узел с наименьшей стоимостью. Какая структура данных позволяет быстро извлекать минимум? ***Ожидаемый ответ***: Приоритетная очередь (реализованная через кучу, heapq). Поис минимума – O(log n).

4.2 Типовые вопросы к зачету по дисциплине (3-й семестр)

Раздел 1. Введение. Анализ сложности алгоритмов

1. Понятия тип данных, структура данных и абстрактный тип. Основные структуры данных.
2. Особенности работы с динамическими массивами (различие между `size` и `capacity`).
3. Подходы к оценке сложности алгоритма. Нотация Big O. Другие виды оценок. Для чего необходимо понимание нотации Big O.
4. Понятие временной и асимптотической сложности алгоритма. Оценка пространственной эффективности.
5. Правила вычисления времени выполнения программ.
6. Наиболее распространенные асимптотики: константная, логарифмическая, сублинейная, линейная, линейно-логарифмическая, квадратичная, экспоненциальная, факториальная. Примеры программ на разные виды асимптотик.

Раздел 2. Алгоритмы сортировки и поиска данных

7. Что такое сортировка, каковы ее цель и задача. Какие показатели влияют на эффективность сортировки? Свойства алгоритмов сортировки. Требования к методам сортировки.
8. Сортировка простым выбором: реализация, оценка эффективности.
9. Пузырьковая сортировка: реализация, оценка эффективности, подходы к улучшению алгоритма.
10. Сортировка простыми вставками: реализация, оценка эффективности.
11. Сортировка слиянием: реализация, оценка эффективности.
12. Быстрая сортировка: рекурсивная реализация, оценка эффективности. Определение барьерного элемента и зависимость эффективности сортировки от его выбора.
13. Быстрая сортировка: не рекурсивная реализация, оценка эффективности.
14. Сортировка подсчетом: ограничения на использование, реализация, оценка эффективности. Применение алгоритма к сортировке отрицательных чисел.
15. Определение числа сравнений и числа присваиваний в алгоритмах сортировки.
16. Понятие компаратора и его использование при сортировке.
17. Понятие инверсии. Программа определения числа инверсий.
18. Реализация линейного поиска элемента в массиве, недостаток линейного поиска. Задачи, решаемые линейным поиском.
19. Бинарный поиск элемента в массиве: основная идея и реализация. Сложность бинарного поиска.
20. Левый и правый поиск в неубывающем массиве.

21. Программная реализация поиска единицы в массиве из нулей и единиц. Использование идеи бинарного поиска по ответу для монотонных функций.
22. Вещественный бинарный поиск: его особенности.

Раздел 3. Фундаментальные структуры данных

23. Стек: определение, основные и дополнительные операции, способы реализации. Определение асимптотической сложности операций над стеком.
24. Программная реализация структуры данных стек.
25. Понятие правильной скобочной последовательности (ПСП). Алгоритм проверки ПСП.
26. Инфиксные, префиксные и постфиксные записи арифметических выражений. Преобразование инфиксного выражения в префиксное и постфиксное.
27. Преобразование инфиксной записи с полностью расставленными скобками в префиксную и постфиксную формы.
28. Алгоритм преобразования инфиксной записи в постфиксную.
29. Стек с поддержкой некоторой операции: особенности реализации.
30. Очередь: определение, основные и дополнительные операции, способы реализации. Определение асимптотической сложности операций над очередью.
31. Программная реализация структуры данных очередь.
32. Идея реализации очереди на двух стеках.
33. Дек: определение, основные и дополнительные операции, способы реализации. Определение асимптотической сложности операций над деком.
34. Программная реализация структуры данных дек.
35. Реализация линейных структур в языке программирования.
36. Понятие узла в динамических структурах. Реализация класса `Node`. Виды узлов и типы структур данных.
37. Реализация основных операций над односвязным списком: вставка элемента в начало и конец списка, проверка списка на пустоту, определение размера списка. Техника обхода связного списка.
38. Реализация основных операций над односвязным списком: поиск значения, удаление элемента с ключом `key`, удаление последнего элемента.
39. Реализация стека на односвязном списке.

Раздел 4. Древовидные структуры данных

40. Понятие приоритетной очереди. Куча как структура данных: понятие кучи, особенности хранения, основные операции.
41. Реализация кучи. Процедура просеивания вниз и вверх.
42. Пирамидальная сортировка.
43. Python библиотека `heapq`. Пример использования.
44. Класс `PriorityQueue` библиотеки `queue`. Пример использования.
45. Примеры иерархических структур данных. Дерево как структура данных. Терминология деревьев (корень, узел, ветвь, лист, родитель, предок, братья, путь, поддерево, степень узла, степень дерева, высота дерева).
46. Бинарное (двоичное) дерево. Бинарное дерево поиска (BST). Где используются BST? Сбалансированные и несбалансированные деревья. Методы балансировки двоичных деревьев.
47. Реализация бинарного дерева поиска (добавление, удаление, поиск элемента).
48. Алгоритмы обхода дерева поиска в глубину. Реализация алгоритмов обхода дерева поиска в глубину: префиксная форма, постфиксная форма, вывод по возрастанию, вывод по убыванию.
49. Алгоритм обхода дерева поиска в ширину. Реализация алгоритма.

Раздел 5. Алгоритмы на графах

50. Понятие графа. Вершина. Ребро. Петля. Диаграмма неориентированного и ориентированного графа.
51. Способы задания графа. Пустой граф, полный граф. Плоский граф. Примеры.

52. Инцидентность вершины и ребра графа. Понятие кратных ребер. Примеры.
53. Мультиграф и псевдограф. Простой граф. Примеры.
54. Смежные вершины. Смежные ребра. Степень вершины. Изолированная и висячая вершина. Примеры.
55. Путь, простой путь. Цикл, простой цикл. Достижимость вершины.
56. Обратный граф для неориентированного и ориентированного графа. Примеры.
57. Понятие маршрута. Цепь. Простая цепь. Примеры.
58. Понятие подграфа. Остовной граф. Двудольный граф. Примеры.
59. Виды графов: ориентированный, неориентированный, цикличный, ацикличный. Примеры.
60. Виды графов: взвешенный, невзвешенный, связный, несвязный. Примеры. Понятие компоненты связности.
61. Дерево, лес, Регулярный граф, разреженные и плотные графы.
62. Представление графов в памяти: список ребер.
63. Представление графов в памяти: матрица смежности.
64. Представление графов в памяти: список вершин.
65. Сравнение способов представления графов.
66. Понятие степеней вершины для ориентированного графа (полустепень исхода, полустепень захода). Примеры.
67. Пути и циклы в ориентированном графе. Примеры.
68. Понятие сильной связности. Компонента сильной связности.
69. Матрица смежности невзвешеного и взвешенного графа. Примеры.
70. Матрица инцидентности неориентированного графа. Пример.
71. Матрица инцидентности ориентированного графа. Пример.
72. Визуализация неориентированного графа с использованием библиотеки networkx.
73. Визуализация ориентированного графа с использованием библиотеки networkx.
74. Алгоритм обхода графа в глубину (DFS).
75. DFS с покраской вершин в бело-серо-черный цвет.
76. Определение цикла в ориентированном графе.
77. Определение цикла в неориентированном графе.
78. Алгоритм обхода графа в ширину (BFS).
79. BFS с восстановлением пути.
80. Сравнение BFS и DFS. Задачи, решаемые этими алгоритмами.
81. Топологическая сортировка. Идея, применение, программная реализация.
82. Алгоритм Дейкстры. Идея, применение, программная реализация (без кучи).
83. Алгоритм Дейкстры. Идея, применение, программная реализация (с кучей).
84. Восстановление пути в алгоритме Дейкстры. Идея, программная реализация.
85. Алгоритм Флойда поиска кратчайшего пути в графе с отрицательными весами.
86. Восстановление пути в алгоритме Флойда. Идея, программная реализация.

Раздел 6. Алгоритмы на строках

87. Базовые определения: подстрока, префикс, суффикс, собственный префикс/суффикс. Наивный алгоритм поиска подстроки в строке.
88. Определение префикс-функции строки. Алгоритм подсчета префикс-функции строки.
89. Определение z-функции строки. Алгоритм подсчета z-функции строки.
90. Алгоритм Кнута-Морриса-Пратта (поиск подстроки в строке).
91. Понятие хеш-функции. Требования к хеш-функции. Понятие коллизии, методы разрешения коллизий.
92. Хеширование строк с использованием обратного полиномиального хеша.
93. Хеширование строк с использованием прямого полиномиального хеша.

4.3 Типовые задачи для проведения зачета по дисциплине (3-й семестр)

Задача 1. Кратчайший путь

Дан неориентированный связный взвешенный граф. Найдите кратчайшее расстояние от первой вершины до всех вершин.

Формат ввода.

В первой строке входного файла два числа: n и m ($2 \le n \le 30000$, $1 \le m \le 400000$), где n – количество вершин графа, а m – количество ребер. Следующие m строк содержат описание ребер. Каждое ребро задается стартовой вершиной, конечной вершиной и весом ребра. Вес каждого ребра – неотрицательное целое число, не превосходящее 10^4. В графе могут быть кратные ребра.

Формат вывода

Выведите n чисел – для каждой вершины кратчайшее расстояние до нее.

Задача 2. Пирамидальная сортировка

В этой задаче вам необходимо организовать структуру данных *Heap* для хранения целых чисел и с её помощью отсортировать заданный массив. Эту задачу нужно решить без использования встроенных алгоритмов/структур данных для сортировок.

Формат ввода

В первой строке входного файла задано одно натуральное число n ($1 \le n \le 100000$). Во второй строке задан массив a размера n, где $-10^9 < a_i < 10^9$.

Формат вывода

В выходной файл необходимо вывести отсортированный массив a.

Задача 3. Стек (с поддержкой максимума)

Стек – абстрактная структура данных, поддерживающая операции `push` и `pop`. Несложно реализовать стек так, чтобы обе эти операции работали за константное время. В данной задаче ваша цель – расширить интерфейс стека так, чтобы он дополнительно поддерживал операцию `max` и при этом чтобы время работы всех операций по-прежнему было константным.

Формат ввода

Первая строка содержит число запросов q. Каждая из последующих q строк задаёт запрос в одном из следующих форматов: `push v`, `pop`, `max` ($1 \le q \le 400000$, $0 \le v \le 100000$).

Формат вывода

Для каждого запроса max выведите (в отдельной строке) текущий максимум на стеке.

4.4 Типовые темы курсовых работ

Тема 1. Оценка схожести текстов с помощью расстояния Левенштейна
Тема 2. Персистентные структуры: реализация и практическое использование в задачах ИИ
Тема 3. Строковые структуры: реализация и практическое использование в задачах ИИ
Тема 4. Корневые деревья: реализация и практическое использование в задачах ИИ
Тема 5. Дерево отрезков: реализация и практическое использование

Оператор ЭДО ООО "Компания "Тензор"

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ

ПОДПИСАНО ЗАВЕДУЮЩИМ КАФЕДРЫ	**ФГБОУ ВО "РГРТУ", РГРТУ,** Овечкин Геннадий Владимирович, Заведующий кафедрой ВПМ	**07.10.26** 00:09 (MSK)	Простая подпись
ПОДПИСАНО ЗАВЕДУЮЩИМ ВЫПУСКАЮЩЕЙ КАФЕДРЫ	**ФГБОУ ВО "РГРТУ", РГРТУ,** Овечкин Геннадий Владимирович, Заведующий кафедрой ВПМ	**07.10.26** 00:09 (MSK)	Простая подпись