Приложение 2 к РПД
Структуры и алгоритмы обработки данных
09.03.01 Информатика и вычислительная техника
Направленность (профиль)
Виртуальные технологии и дизайн
Форма обучения – очная
Год набора – 2021

ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

1. Общие сведения

1.	Кафедра	Математики, физики и информационных технологий
2.	Направление подготовки	09.03.01 Информатика и вычислительная техника
3.	Направленность (профиль)	Виртуальные технологии и дизайн
4.	Дисциплина (модуль)	Б1.О.17.02 Структуры и алгоритмы обработки данных
5.	Форма обучения	очная
6.	Год набора	2021

2. Перечень компетенций

– **ОПК-8**: Способен разрабатывать алгоритмы и программы, пригодные для практического применения

3. Критерии и показатели оценивания компетенций на различных этапах их формирования

Этап формирования компетенции (разделы, темы дисциплины)	Формируемая компетенция	Критерии и показатели оценивания компетенций			Формы контроля сформированности компетенций
		Знать:	**Уметь:**	**Владеть:**	
Структуры данных	ОПК-8	основные понятия курса «Структуры и алгоритмы обработки данных»	проектировать структуры данных, наиболее подходящие для поставленной задачи	навыками использования различных структур данных	Опрос на практических занятиях. Лабораторные работы 1-2
Методы сортировки и поиска	ОПК-8	основные методы разработки машинных алгоритмов и программ	реализовывать изученные алгоритмы и структуры данных для представления информационных объектов средствами языков программирования высокого уровня	навыками программной реализации классических алгоритмов	Опрос на практических занятиях. Лабораторные работы 3-5
Алгоритмы обработки данных	ОПК-8	основные подходы к решению «трудно решаемых» задач	экспериментально исследовать эффективность алгоритмов и программ	навыками оценки теоретической сложности алгоритмов	Опрос на практических занятиях. Лабораторные работы 6-8

Шкала оценивания в рамках балльно-рейтинговой системы: «неудовлетворительно» – 60 баллов и менее; «удовлетворительно» – 61-80 баллов; «хорошо» – 81-90 баллов; «отлично» – 91-100 баллов

4. Критерии и шкалы оценивания

4.1 Каждая лабораторная работа оценивается в зависимости от полноты и самостоятельности выполнения от 1 до 6 баллов.

4.2 Каждая практическая работа оценивается в зависимости от полноты и самостоятельности выполнения от 0 до 1 баллов.

4. Типовые контрольные задания и методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

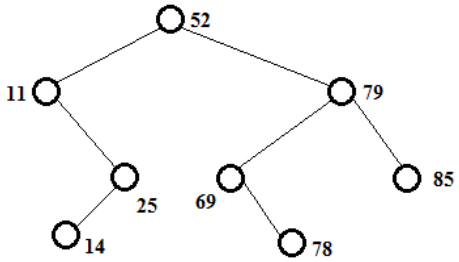
Задача. Быстрый поиск
По данной последовательности ключей: 52, 79, 11, 69, 78, 85, 94, 65
а) сформировать дерево поиска,
б) удалить корневой элемент.

Решение.
Дана последовательность ключей: 52, 79, 11, 69, 78, 85, 25, 14

а) Первый ключ – 52 – помещаем в корень. Далее, спустившись по пути поиска от корня до вершины V с нулевым указателем, создаем новую вершину, "цепляя" её

- слева от вершины V, если новый ключ меньше ключа вершины V, или
- справа от вершины V, если новый ключ больше ключа вершины V.

Проделывая *поочерёдно* эту процедуру для **каждого элемента** входной последовательности, получим следующее бинарное дерево:



б) В случае исключения элемента из дерева поиска возможны три ситуации: удаляется

- *концевая вершина*
- *вершина с одним потомком*
- *вершина с двумя потомками*.

Первые две ситуации не вызывают трудностей и обрабатываются одинаково. Пусть удаляется вершина V, имеющая предка U и одного потомка Z. Тогда достаточно позаботиться о том, чтобы вершина U ссылалась теперь на Z, а не на V, после чего командой free(<adres V>) очистить участок памяти, занятый вершиной V.

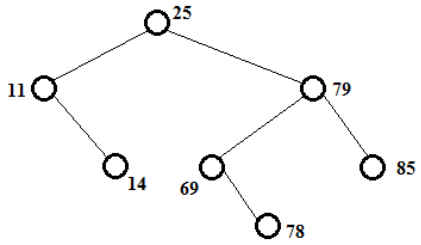
Сложность третьей ситуации в том, что с помощью одной ссылки (от предка удаляемой вершины) нельзя указать сразу два направления (к обоим потомкам удаляемой вершины). В этом случае удаляемый элемент заменяют либо на *самый правый элемент его левого поддерева*, либо на *самый левый элемент его правого поддерева*. Каждый из этих элементов имеет не более одного потомка, иначе он не был бы самым правым (левым), причем их перенос на место удаляемого элемента не нарушает структуру *дерева поиска*.

Рассмотрим более подробно процесс преобразования дерева поиска после удаления вершины с двумя потомками. Пусть удаляется вершина V, имеющая двух потомков. Тогда поиск замещающего элемента W (самого правого элемента левого поддерева вершины V) осуществляется следующим образом: спускаемся вдоль самой правой ветви левого поддерева вершины V до тех пор, пока не найдем вершину W с нулевым правым указателем. После этого заменяем вершину V на W, "цепляя" единственного потомка вершины W (если он есть) к предку этой вершины.

Корневой элемент с ключом 52 имеет двух потомков. Ключ 52 можно заменить

- либо на *самый правый элемент левого поддерева* – 25,
- либо на *самый левый элемент правого поддерева* – 69.

Заменим, например, на 25. При этом единственного потомка вершины V с ключом 25 – вершину с ключом 14 – цепляем к предку вершины V, т.е. к вершине с ключом 11:



Вопросы к экзамену:

1. Алгоритмы. Временная и пространственная сложность алгоритма.
2. Стек и дек. Операции, определённые над стеком и деком. Машинное представление.
3. Очередь. Операции, определённые над очередью. Машинное представление очереди.
4. Список с двумя связями. Операции, определенные над этим списком. Машинное представление списка с двумя связями
5. Множество. Представление множества в виде массива, линейного списка и характеристического вектора.
6. Бинарные деревья. Операции, определенные над бинарными деревьями. Машинное представление деревьев.
7. Сортировка *включением*: сортировка простым включением, Метод Шелла. Временная сложность алгоритмов.
8. Сортировка *включением*: сортировка слиянием и ее реализация с помощью рекурсии. Временная сложность алгоритма.
9. Сортировка *обменами*: пузырьковая сортировка, быстрая сортировка. Временная сложность алгоритмов.
10. Последовательный поиск в линейном списке и не отсортированном массиве. Дихотомический (логарифмический) поиск в отсортированном массиве. Временная сложность алгоритмов.
11. Дерево поиска. Процедуры поиска, включения и исключения элементов для дерева поиска.
12. Хеширование. Поиск, включение и исключение элементов.
13. Полиномиальные алгоритмы. Задачи классов P и NP. NP-полные задачи.
14. Жадные алгоритмы – алгоритм поиска оптимального остовного дерева,
15. Жадные алгоритмы – кодирование Хаффмена,
16. Жадные алгоритмы – поиск кратчайшего пути в графе.
17. Псевдополиномиальные алгоритмы - задача о рюкзаке.
18. Псевдополиномиальные алгоритмы - задача коммивояжера.
19. Алгоритм быстрого дискретного преобразования Фурье.
20. Перемножение длинных чисел.