Экзамен по предмету “Алгоритмы и Структуры данных”.

Вариант 2

Чобану Артём I1902

1. Определить понятия Структуры данных (СД) и Динамической СД. Привести примеры. (5 p.)

**Структура данных** – единица, содержащее множество однотипных данных, организованных по определённой структуре. **Пример**: массив, дерево.

**Динамическая структура данных** – структура данных, не имеющая фиксированной длины – т.е. её длина может изменяться. **Пример**: односвязный список.

1. Задан одномерный массив: {-2, -1, 1, 3, 5, 7, 10, 11}.
2. Изложить метод бинарного (двоичного) поиска. (5 p.)

**Метод бинарного поиска** подразумевает наличие отрезка индексов, внутри которого находится искомый элемент. Каждую итерацию находится середина этого отрезка, и отсекается часть, в которой нет ответа. Метод может работать только с отсортированным массивом.

1. Применить указанный метод для поиска элемента, имеющего значение 1. (5 p.)

Пусть l – левая граница отрезка, а r – правая граница. l = 0, r = 7, так как последний элемент имеет индекс 7. Пусть m – середина отрезка, и a[i] – обозначение элемента с данным индексом в массиве.

m = (r-l)/2 = 3

Так как a[m] = a[3] = 3 < a[r], присваиваем r = m = 3;

Далее работаем на отрезке [l,r] = [0,3]

m = 1. a[m] = -1 > a[l] => присваиваем l = m = 1.

Получили отрезок [1,3]

m = 2. a[m] = a[2] = 1 => ответ.

1. Заданы элементы: my, file, off, copy, open, draw, delete, up, esc, split.
2. Построить дискретную Хэш функцию в виде таблицы пар значений: исходный ключ – значение построенной дискретной Хэш функции. (6 p.)

Хеш функция для строк должна сложить номера символов, умноженные на некую константу, возведённую в степень индекса символа в строке.

Для строки “file”:

HashCode(“file”) = charAt(‘f’) \* c ^ 0 + charAt(‘i’) \* c^1 + charAt(‘l’) \* c^2 + charAt(‘e’) \* c^3,

Где charAt – функция, возвращающая номер символа

C – любое натуральное число

b) Объяснить пошагово поиск первого и последнего элемента из заданного массива. (6 p.) c) Объяснить неудачный поиск для несуществующего элемента заданного массива. (3 p.) d) Вычислить среднюю теоретическую и практическую сложность поиска в Хэш таблице.

(4 p.)

1. a) Объяснить для заданного массива сортировку пирамидальным методом, согласно первой букве элементов массива. (9 p.)

**Алгоритм пирамидальной сортировки** использует бинарное дерево сортировки. Каждую итерацию дерево заново перестраивается. Каждый элемент с индексом i должен быть не меньше чем элементы с индексами 2i+1 и 2i+2. Далее, элемент из корня переносится в конец не отсортированной части массива. Далее элемент считается отсортированным, а оставшаяся часть – нет.

Пусть задан одномерный массив: {c, a, b, i, d, e}.

Ожидаемый отсортированный массив: {a, b, c, d, e, i}.

Перестраивание: {i, e, d, b, c, a}. Здесь i – самый большой элемент. Меняем местами с последним элементом: {a, e, d, b, c, i}. Далее неотсортированная часть массива выглядит следующим образом: a, e, d, b, c

Перестраивание: {e, a, d, b, c, i} (перестраивается не отсортированный массив, то есть на отрезке индексов [0,4]). Элемент e меняется местами с c: {c, a, d, b, e, i}

**Итог сортировки**: {a, b, c, d, e, i}

b) Оценить количество сравнений и перестановок. (7 p.)

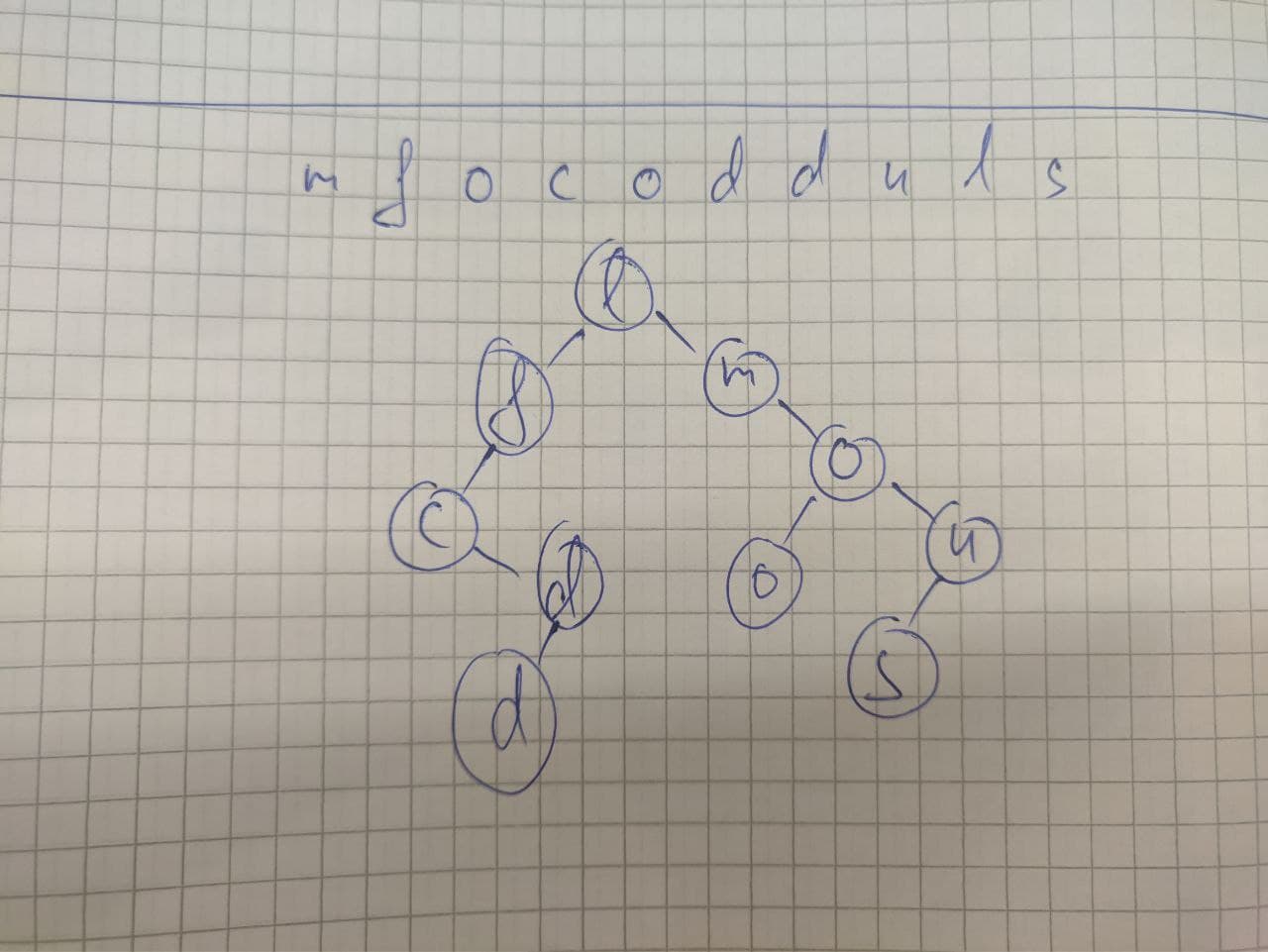
1. Оценить сложность сортировки для заданного метода. (2 p.)

**Сложность**: **O(nlogn)** так как операция перестраивания дерева имеет сложность **O(n)** и

1. a) Построить двоичное дерево поиска для заданных элементов. (5 p.)

Массив из первых букв элементов my, file, off, copy, open, draw, delete, up, esc, split выглядит следующим образом: {m, f, o, c, o, d, d, u, e, s}

Отсортированный: {c, d, d, f, m, o, o, s, u}



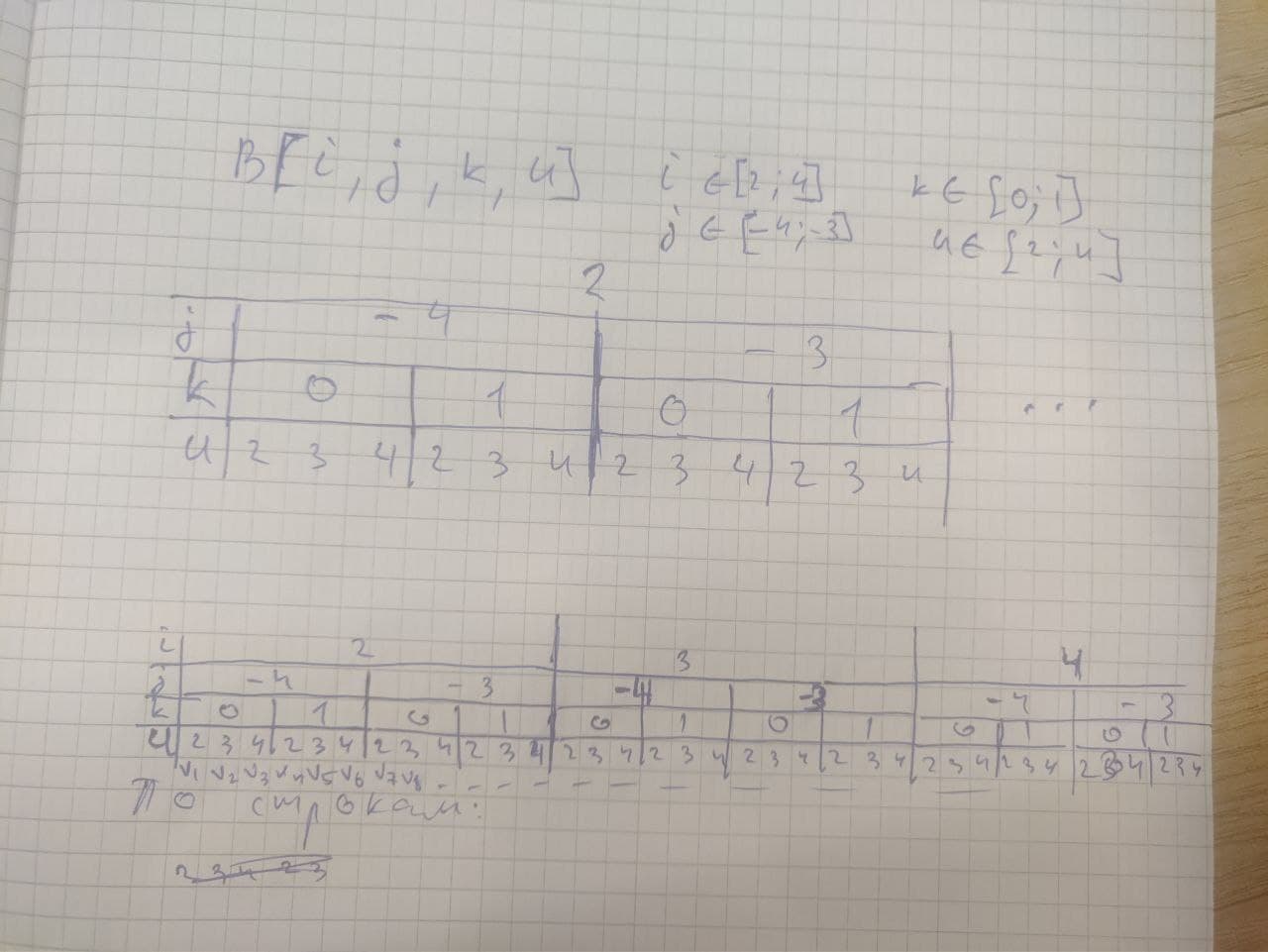
b) Перечислить вершины дерева согласно центрированному (inorder) обходу, указывая вставку и удаление из стека. (8 p.)

**Обход**: d, d, c, f, l, m, o, o, s, u

c) Вычислить среднюю сложность поиска в этом двоичном дереве (4 p.)

Сложность составляет O(nlogn), так как дерево содержит n элементов и nlogn уровней.

1. a) Показать размещение в ОП элементов массива B[2..4, -4..-3, 0..1, 2..4] “по строкам”.(5 p.)



b) Представить логическую структуру массива. (5 p.)

c) Построить определяющий вектор (дескриптор) для заданного массива. (8 p.)

a[0] = 4 – размерность

a[1] = 2 – левая граница первого индекса

a[2] = 4 – правая граница первого индекса

a[3] = -4 – левая граница второго индекса

a[4] = -3 – правая граница второго индекса

a[5] = 0 – левая граница третьего индекса

a[6] = 1 – правая граница третьего индекса

a[7] = 2 – левая граница четвёртого индекса

a[8] = 4 – правая граница четвёртого индекса

a[13] = 1

a[12] = a[8]-a[7] = 2

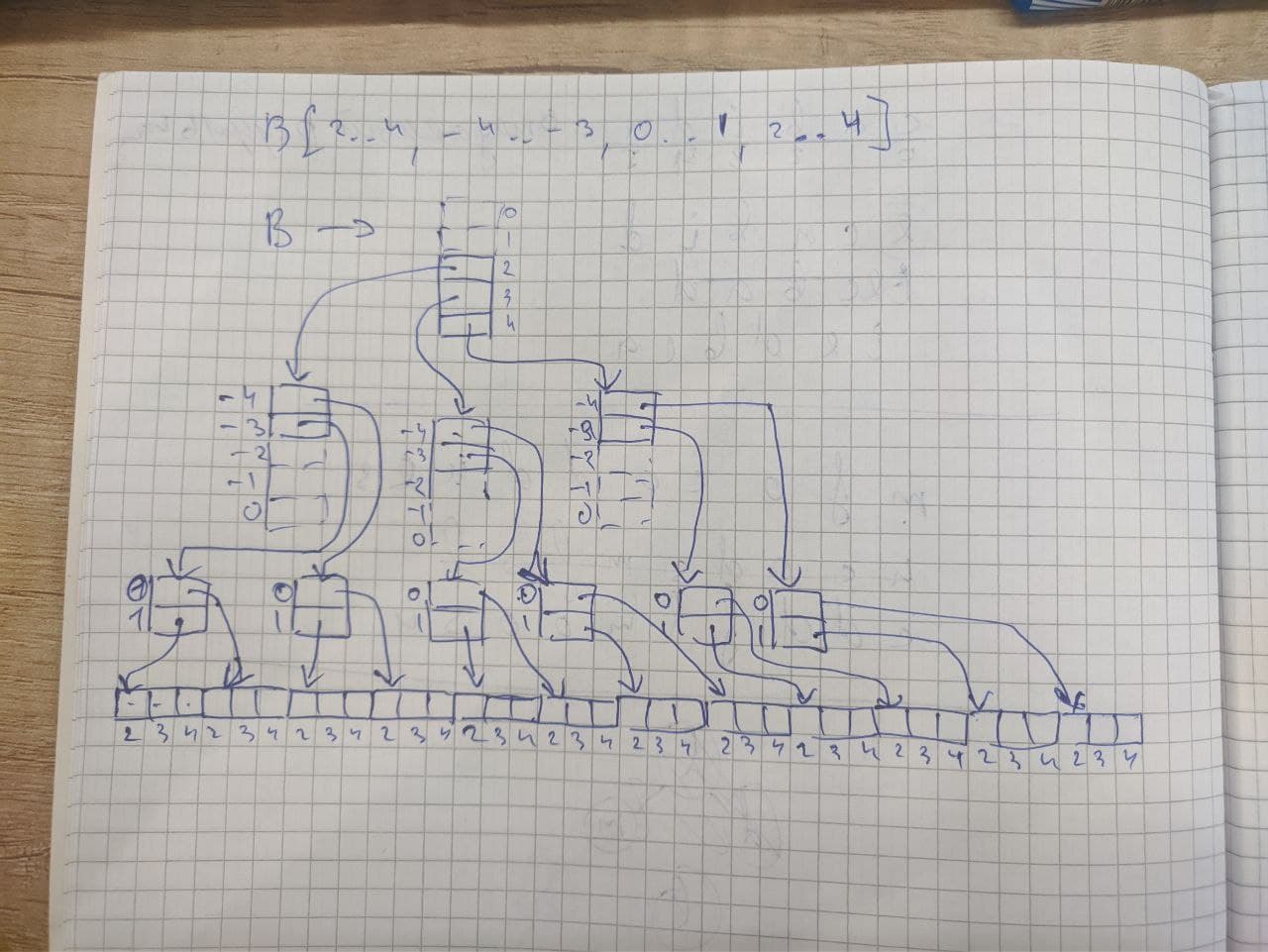
a[11] = (a[6]-a[5]) \* a[12] = (1 – 0) \* 2 = 2

a[10] = (a[4]-a[3]) \* a[11] = (-3 + 4) \* 2 = 2

a[14] = a[1] \* a[10] + a[3] \* a[11] + a[5] \* a[12] + a[7] \* a[13] =

= 2 \* 2 + -4 \* 2 + 0 + 2 \* 1 = 4 – 8 + 2 = -2

1. Построить вектор Айлиффа для заданного массива. (8 p.)



e) Изобразить схему доступа к 2 различным элементам (на основе вектора Айлиффа). (5 p.)

