Экзамен по ИИП 2 сем

1 Понятие Θ, Ω-нотации. Оценка сложности алгоритма

Θ-нотация: асимптотически точная оценка (верхняя и нижняя границы совпадают)

Ω-нотация: нижняя граница сложности алгоритма.

Оценка сложности алгоритма — это определение того, как объём работы алгоритма зависит от размера входных данных. Сложность оценивают по времени выполнения или по используемой памяти.

Сложность алгоритма в наихудшем случае — функция, определяемая максимальным количеством шагов, необходимым для обработки любого экземпляра задачи. Для сортировок — количество шагов, необходимое для сортировки отсортированной в обратном порядке последовательности.

Сложность алгоритма в наилучшем случае — функция, определяемая минимальным количеством шагов, необходимым для обработки любого экземпляра задачи. Для сортировок — количество шагов, необходимое для сортировки уже отсортированной последовательности.

Сложность алгоритма в среднем случае — функция, определяемая средним количеством шагов, необходимым для обработки всех экземпляров задачи. Для сортировок — количество шагов, необходимое для сортировки случайной после- довательности.

2 Сортировка. Примеры устойчивой и неустойчивой сортировок

Сортировка — процесс упорядочивания множества объектов по какому-либо признаку.

Устойчивой называется сортировка, которая не меняет порядка объектов с одинаковыми ключами.

Некоторые примеры устойчивой сортировки:

Сортировка пузырьком. Алгоритм состоит в повторяющихся проходах по сортируемому массиву. На каждой итерации последовательно сравниваются соседние элементы, и, если порядок в паре неверный, то элементы меняют местами.

Сортировка слиянием. Алгоритм состоит в разделении массива пополам, сортировке половин и их слиянии.

Некоторые примеры неустойчивой сортировки:

Сортировка выбором. Алгоритм ищет наименьший элемент в текущем списке и производит обмен его значения со значением первой неотсортированной позиции. То же самое происходит со вторым элементом с наименьшим значением. Цикл повторяется до тех пор, пока все элементы не займут нужную последовательность.

Быстрая сортировка. Считается одним из самых быстрых алгоритмов сортировки. Работает по принципу «разделяй и властвуй». **Пример устойчивой и неустойчивой сортировки**

Исходный массив:

$$\langle 1,\ 2,\ 5',\ 6,\ 3,\ 4,\ 5'' \rangle$$

Результаты сортировки:

$$\text{Устойчивая: } \langle 1,\ 2,\ 3,\ 4,\ 5',\ 5'',\ 6 \rangle$$
$$\text{Неустойчивая: } \langle 1,\ 2,\ 3,\ 4,\ 5'',\ 5',\ 6 \rangle$$

3 Сортировки за $O(n^2)$: вставка, пузырьком, выбор

Пузырьком: поменяв местами два элемента не возвращаемся назад, а идем дальше. Тогда при одном проходе минимальный элемент окажется на своем месте

8	3	9	6	1	2	5

Начальное состояние

$i = 0$:

8	3	9	1	6	2	5
8	3	1	9	6	2	5
8	1	3	9	6	2	5
1	8	3	9	6	2	5

$i = 1$:

1	8	3	9	2	6	5
1	8	3	2	9	6	5
1	8	2	3	9	6	5
1	2	8	3	9	6	5

$i = 2$:

1	2	8	3	9	5	6
1	2	8	3	5	9	6

1	2	3	8	5	9	6

$i = 3$:

1	2	3	8	5	6	9
1	2	3	5	8	6	9

$i = 4$:

1	2	3	5	6	8	9

Результат:

1	2	3	5	6	8	9

Вставка:Можно сказать, что это модификация гномьей сортировки. Гном идет вперед, пока не найдет неправильно стоящие горшки, запоминает на каком горшке он остановился, меняет горшки местами и идет назад, пока не поставит горшок на место. Однако дальше идет не на шаг вперед, а сразу начинает с горшка, который он запомнил

8	3	9	6	1	2	5

$i = 1$

3	8	9	6	1	2	5

$i = 2$

$i = 3$

3	8	6	9	1	2	5
3	6	8	9	1	2	5

$i = 4$

3	6	8	1	9	2	5
3	6	1	8	9	2	5
3	1	6	8	9	2	5
1	3	6	8	9	2	5

Выбор: находим минимум из элементов массива от i+1 до конца, меняем местами i- тый и минимальный элементы

Шаг	Состояние массива						
0	8	3	9	6	1	2	5
1	1	3	9	6	8	2	5
2	1	2	9	6	8	3	5
3	1	2	3	6	8	9	5
4	1	2	3	5	8	9	6
5	1	2	3	5	6	9	8
6	1	2	3	5	6	8	9

4 Быстрая сортировка и сортировка слиянием

Быстрая сортировка

1. Выбираем опорный элемент.

2. Меняем местами элементы так, чтобы слева были меньшие опорного, справа — большие.

3. Рекурсивно вызываем функцию для подмассивов.

Сортировка слиянием

Если есть две отсортированные последовательности, то результатом слияния должна быть отсортированная последовательность. Алгоритм слияния можно объяснить на примере двух колод карт. Обе колоды отсортированы, лежат рубашками вниз. Берется верхние карты из каждой колоды, сравниваются, меньшая откладывается, берется следующая карта из этой колоды.

Рассмотрим на примере. Есть два отсортированных массива: 2, 5, 9 и 1, 3, 4. У первого массива красным цветом показан счетчик, у второго — синим. Зеленым цветом показан счетчик результирующего массива. В результирующим массив записывается минимальный из этих элементов.

2	5	9		1	3	4		1					
2	5	9		1	3	4		1	2				
2	5	9		1	3	4		1	2	3			
2	5	9		1	3	4		1	2	3	4		
2	5	9		1	3	4		1	2	3	4	5	9

5 Понятие кучи. Пирамидальная сортировка

Куча — это специальное бинарное дерево, обладающее следующими свойствами:

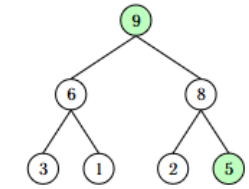
1. Дерево почти полное, то есть, заполнено на всех уровнях, кроме последнего. На последнем уровне элементы заполня ются слева направо, пока не закончатся элементы.

2. Значение узла всегда больше значений его потомков. Следовательно, корнем дерева будет будет максимальный элемент

Пиромидальная сортировка

На вершине кучи всегда находится максимальный элемент. Следовательно, алгоритм пирамидальной сортировки заключает- ся в следующем: меняем местами первый и последний элемент и перестраиваем пирамиду, уменьшая количество элементов на 1

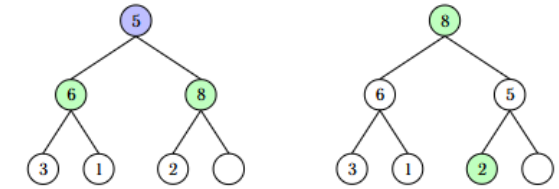
Пирамида выше уже построена. Рассмотрим на этом примере пирамидальную сортировку:



Массив выглядит следующим образом:

9	6	8	3	1	2	5

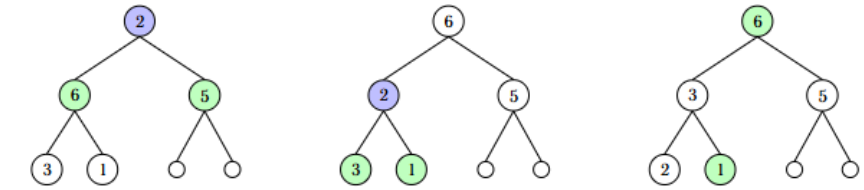
Делаем замену и перестраиваем пирамиду:



Массив выглядит следующим образом:

8	6	5	3	1	2	**9**

Делаем замену и перестраиваем пирамиду:



Массив выглядит следующим образом:

6	3	5	2	1	**8**	**9**

Сорян, я заебался если честно эту хрень делать в LaTeX, продолжение в ворде