ЛАБОРАТОРНАЯ РАБОТА №7

ОЦЕНКА ЭМПИРИЧЕСКОЙ И АСИМПТОТИЧЕСКОЙ СЛОЖНОСТИ АЛГОРИТМОВ

Цель. Актуализация знаний и приобретение практических умений по определению вычислительной сложности алгоритмов.

А. ЭМПИРИЧЕСКАЯ СЛОЖНОСТЬ АЛГОРИТМА

Задание 1. Определить эффективный алгоритм из двух предложенных, используя оценку практической и теоретической сложности каждого из алгоритмов и их ёмкостную сложность.

Пусть имеется задача: дан массив из n элементов целого типа, удалить из массива все значения, равные заданному пользователем (ключевому).

Примечание: удаление состоит в уменьшении размера массива («эффективного» размера). Удаление осуществляется путем сдвига элементов массива к его началу с сохранением порядка следования (как до удаляемого, так и следующих после удаляемого).

Например, нужно удалить из массива все значения, равные 2.

Исходный массив: n=10; A(n)={1,2,3,2,2,2,5,2,2,2}.

Результат: n=3; A(n)={1,3,5}.

Два алгоритма решения этой задачи:

x-массив, n – количество элементов в массиве, key – удаляемое значение

Алгоритм 1.

```
delFirstMetod(x,n,key) {
i←1
while (i<=n) do
        if x[i]=key then
                //удаление
                 for j←i to n-1 do
                 x[j] ←x[j+1]
               od
               n←n-1
      else
             i←i+1
     endif
od
}
```

Алгоритм 2

```
delOtherMetod(x,n,key) {
j←1
for i←1 to n do
  x[j]=x[i];
  if x[i]!=key then
     j++
  endif
od
n←j
}
```

Требования к выполнению задания 1:

1. Для каждого алгоритма привести в отчёте следующие результаты:
   1.1. Мат. модель решения поставленной задачи.
   1.2. На основе псевдокода, используя теоретический подход, для каждого из алгоритмов определить:
      a. Что будет ситуациями лучшего, среднего и худшего случаев.

b. ***Функции роста*** времени работы алгоритмов.
c. По полученной функции роста – количество $T_т$ критических операций (как сумму сравнений $C_т$ + перемещений элементов в памяти $M_т$) при n = 10, 100, 1000.
d. Вычислительную сложность алгоритма (как ***порядок роста*** функции роста).

1.3. Реализовать алгоритмы в виде функций и отладить на массиве при n=10 для ситуаций лучшего, среднего и худшего случаев.

1.4. Включить в функции п.1.4. операторы, подсчитывающие число выполненных критических операций $T_п$ как сумму сравнений ($C_п$) + перемещений элементов в памяти ($M_п$).

1.5. Реализовать дополнительные возможности:
a. заполнение массива:
- вручную с клавиатуры,
- датчиком случайных чисел в заданном диапазоне,
- заданным значением (например, ключевым);
b. вывод массива на экран;
c. подсчёт времени сортировки в миллисекундах (в main).

1.6. Протестировать алгоритм при n = 10, 100, 1000, 10000 в случаях: все элементы должны быть удалены (худший случай), случайное заполнение (средний случай), ни один элемент не удаляется (лучший случай).

Примечание: для обеспечения равных условий при сравнении работы двух алгоритмов в среднем случае (случайное заполнение) следует предусмотреть, чтобы они обрабатывали одинаковые входные массивы.

1.7. Представить результаты эмпирического исследования в таблице (по примеру таблицы 2), указав для каждого n время работы алгоритмов в мс, $T_т$ – количество критических операций согласно теоретическим расчетам (п. 1.3.c) и $T_п$ – практически полученное количество операций при выполнении алгоритма (по счётчикам).

Примечание: в ситуациях лучшего и худшего случаев $T_т = T_п$.

1.8. Определить ёмкостную сложность алгоритмов.

2. Предложите способ как привести длину массива после удаления ключевых элементов к «эффективной» (для экономного расходования памяти).
3. Сделать в отчёте вывод – какой алгоритм в каком случае эффективнее.

Задание 2. Оценить эмпирически вычислительную сложность алгоритма простой сортировки на массиве, заполненном случайными числами (средний случай).

1. Составить функцию простой сортировки одномерного целочисленного массива A[n], используя алгоритм согласно варианту индивидуального задания

(столбец *Алгоритм заданий 2 и 3* в таблице 1*)*. Провести тестирование программы на исходном массиве n=10.

2. Используя теоретический подход, определить для алгоритма:

   a. Что будет ситуациями лучшего, среднего и худшего случаев.

   b. Функции роста времени работы алгоритма от объёма входа для лучшего и худшего случаев.

Таблица 1. Варианты индивидуальных заданий

№	Алгоритм заданий 2 и 3	Алгоритм заданий 4 и 5
1	Простой вставки (***Insertion sort***)	Простого выбора ***(Selection sort)***
2	Простого обмена («пузырек», ***Exchange sort***)	Простой вставки (***Insertion sort***)
3	Простого выбора ***(Selection sort)***	Простого обмена («пузырек», ***Exchange sort***)

3. Провести контрольные прогоны программы массивов случайных чисел при n = 100, 1000, 10000, 100000 и 1000000 элементов с вычислением времени выполнения T(n) – (в миллисекундах/секундах). Полученные результаты свести в сводную таблицу 2.

4. Провести эмпирическую оценку вычислительной сложности алгоритма, для чего предусмотреть в программе подсчет фактического количества критических операций $T_п$ как сумму сравнений $C_п$ и перемещений $M_п$. Полученные результаты вставить в сводную таблицу 2.

Примечание: столбец $T_т$ для среднего случая оставим незаполненным.

Таблица 2. Сводная таблица результатов

n	**T(n), мс**	**$T_т$=C+M**	**$T_п=C_п+M_п$**
100			
1000			
10000			
100000			
1000000			

5. Построить график функции роста $T_п$ этого алгоритма от размера массива n.

6. Определить ёмкостную сложность алгоритма.

7. Сделать вывод об эмпирической вычислительной сложности алгоритма на основе скорости роста функции роста.

Задание 3. Оценить вычислительную сложность алгоритма простой сортировки в наихудшем и наилучшем случаях.

1. Провести дополнительные прогоны программы на массивах при n = 100, 1000, 10000, 100000 и 1000000 элементов, отсортированных:

   a. строго в убывающем порядке значений, результаты представить в сводной таблице по формату *Таблицы 2*;

b. строго в возрастающем порядке значений, результаты представить в сводной таблице по формату *Таблицы 2*;

2. Сделать вывод о зависимости (или независимости) алгоритма сортировки от исходной упорядоченности массива.

Задание 4. Сравнить эффективность алгоритмов простых сортировок

1. Выполнить разработку и программную реализацию второго алгоритма согласно индивидуальному варианту в столбце *Алгоритм заданий 4 и 5* из таблицы 1.
2. Аналогично заданиям 2 и 3 сформировать таблицы с результатами эмпирического исследования второго алгоритма в среднем, лучшем и худшем случаях в соответствии с форматом Таблицы 2 (на тех же массивах, что и в заданиях 2 и 3).
3. Определить ёмкостную сложность алгоритма от n.
4. На одном сравнительном графике отобразить функции $T_п(n)$ двух алгоритмов сортировки в худшем случае.
5. Аналогично на другом общем графике отобразить функции $T_п(n)$ двух алгоритмов сортировки для лучшего случая.
6. Выполнить сравнительный анализ полученных результатов для двух алгоритмов.

Требования к выполнению заданий 2-4:

1. **В отчёте по заданию 2:**
   1.1. Привести алгоритм сортировки по методу варианта *Алгоритм заданий 2 и 3* на случайно заполненном массиве. Представить результаты тестирования при n=10.
   1.2. Описать процесс определения функции роста метода сортировки.
   1.3. Представить сводную таблицу результатов выполнения сортировки по указанным объемам по формату *Таблицы 2* на случайно заполненном массиве для всех указанных объемов.
   1.4. Представить график согласно заданию, выполненный с помощью любого вспомогательного ПО. График должен: быть подписан, быть наглядным, с подписанными и размеченными координатными осями.
2. **В отчёте по заданию 3.**
   2.1. Представить сводную таблицу результатов при применении алгоритма к массиву, упорядоченному: по возрастанию, по убыванию.
   2.2. Представить код программы и результат работы при n=10.
   2.3. Сделать вывод о вычислительной сложности алгоритма.

3. **В отчёте по заданию 4.**
   1. Привести алгоритм сортировки по методу *Алгоритм заданий 4 и 5*. Представить результаты тестирования при n=10.
   2. Описать процесс определения функции роста метода сортировки.
   3. Представить сводные таблицы результатов выполнения сортировки по указанным объемам по формату *Таблица 2*. Для среднего, худшего и лучшего случаев.
   4. Представить графики в соответствии с заданиями. Каждый график должен: быть подписан, быть наглядным, с подписанными и размеченными координатными осями.
   5. Сделать вывод о том, какой из алгоритмов эффективнее на основе данных эмпирического анализа.

Б. АСИМПТОТИЧЕСКАЯ СЛОЖНОСТЬ АЛГОРИТМА

Задание 5. Получить асимптотическую оценку простого алгоритма сортировки

1. На основе определений соответствующих нотаций получите асимптотическую оценку вычислительной сложности простого алгоритма сортировки согласно индивидуальному варианту по столбцу *Алгоритм заданий 4 и 5* таблицы 1:

- в O-нотации (оценка сверху) для анализа худшего случая;
- в Ω-нотации (оценка снизу) для анализа лучшего случая.

Примечание: здесь математически доказывается существование коэффициентов, при которых истинны соответствующие нотациям неравенства.

2. Получите (если это возможно) асимптотически точную оценку вычислительной сложности алгоритма в нотации θ.

3. Привести справочную информацию о вычислительной сложности одного из усовершенствованных и одного из быстрых алгоритмов сортировки.

Для типа сортировки **обменом** усовершенствованным вариантом является шейкерная сортировка с условием Айверсона; быстрая сортировка – Хоара.

Для типа сортировки **вставками** усовершенствованным вариантом является сортировка Шелла; в качестве быстрой сортировки возьмите сортировку слиянием (фон Неймана).

Для типа сортировки **выбором** усовершенствованным вариантом является турнирная (или пирамидальная) сортировка; в качестве быстрой сортировки возьмите сортировку слиянием (фон Неймана).

Общие результаты свести в таблицу по типу таблицы 3.

Сделать вывод о наиболее эффективном алгоритме из этих трёх.

Таблица 3. Сводная таблица результатов

Алгоритм[1]	Асимптотическая сложность алгоритма			
	O-нотация	Ω-нотация	θ-нотация	Ёмкостная сложность
Простой				
Усоверш-й				
Быстрый				

Требования к выполнению задания 5:

1. Привести для заданного алгоритма математический вывод асимптотической оценки каждого вида (сверху, снизу, асимптотически точную).
2. Воспользуйтесь справочной информацией и сформируйте в отчёте сравнительную таблицу асимптотических оценок сложности простого, усовершенствованного и быстрого алгоритмов сортировок.
3. Приведите в отчёте обоснованный вывод о том, какая из трёх рассмотренных сортировок эффективнее.

В общих выводах по лабораторной работе сформулируйте итоги её выполнения в свободной форме.

ЛИТЕРАТУРА:

1. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. – СПб: Питер, 2017. – 288 с.
2. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир, 1985. – 406 с.
3. Кнут Д.Э. Искусство программирования, том 3. Сортировка и поиск, 2-е изд. – М.: ООО «И.Д. Вильямс», 2018. – 832 с.
4. Седжвик Р. Фундаментальные алгоритмы на C++. Анализ/Структуры данных/Сортировка/Поиск. – К.: Издательство «Диасофт», 2001. – 688 с.
5. AlgoList – алгоритмы, методы, исходники [Электронный ресурс]. URL: http://algolist.manual.ru/ (дата обращения 15.03.2022).
6. Алгоритмы – всё об алгоритмах / Хабр [Электронный ресурс]. URL: https://habr.com/ru/hub/algorithms/ (дата обращения 15.03.2022).
7. НОУ ИНТУИТ | Технопарк Mail.ru Group: Алгоритмы и структуры данных [Электронный ресурс]. URL: https://intuit.ru/studies/courses/3496/738/info (дата обращения 15.03.2022).

[1] В этом столбце укажите названия соответствующих сортировок.