Отчет

**Теоретическая часть**

1. Программный измерительный монитор — это инструмент, используемый для наблюдения, записи и анализа различных аспектов работы программного обеспечения.

Список данных - платформа для мониторинга и анализа данных, которая позволяет отслеживать производительность приложений, инфраструктуры и пользовательский опыт.

Новая Реликвия - инструмент для мониторинга производительности приложений, предоставляющий данные в реальном времени о работе веб- и мобильных приложений.

Прометей - система мониторинга и оповещения с открытым исходным кодом, предназначенная для сбора и обработки метрик в режиме реального времени.

2. Производительность - время отклика, загрузка процессора, использование памяти и другие показатели, влияющие на скорость и эффективность работы приложения.

Доступность - процент времени, в течение которого приложение доступно для пользователей, и частота возникновения сбоев или простоев.

Использование ресурсов - мониторинг использования ресурсов, таких как процессор, память, дисковое пространство и сетевые соединения.

3. Запуск программы с монитором - установка и настройка программного измерительного монитора. Запуск приложения вместе с монитором для начала сбора данных.

Сбор данных - монитор начинает собирать данные о различных аспектах работы приложения, таких как производительность, использование ресурсов и доступность

Интерпретация результатов - анализ собранных данных для выявления проблем, узких мест и других аспектов, требующих внимания.

**Практическая часть**

1. import random

#Генерация массива из 10,000,000 случайных чисел

random\_array = [random.randint(1, 100000000) for \_ in range(10000000)]

#Сортировка массива

random\_array.sort()

Сохранение массива в файл

with open('sorted\_array.txt', 'w') as f:

for number in random\_array:

f.write("%s\n" % number)

2. def linear\_search(arr, target):

for i in range(len(arr)):

if arr[i] == target:

return i

return -1

arr = list(range(1, 1000001)) # Создаем массив от 1 до 1,000,000

target = 1000000 # Крайний элемент

result = linear\_search(arr, target)

print(f"Индекс элемента {target}: {result}")

Дальше только открыть PowerShell и выполните следующую команду:

Measure-Command {python test6.py}

2) Изменяем эту часть

arr = list(range(1, 101)) # Создаем массив от 1 до 100

target = 100 # Крайний элемент

result = binary\_search(arr, target)

print(f"Индекс элемента {target}: {result}")

Выполняем эту команду:

Measure-Command {python test7.py}

|  |  |  |  |
| --- | --- | --- | --- |
| Алгоритм | Реальное время (ms) | Использование памяти | Пиковое использование памяти |
| Линейный поиск | 45.2 | 50.7 | 38.1 |
| Бинарный поиск | 0.05 | 48.3 | 38.1 |

3. 1)Бинарный поиск работает быстрее на больших данных по сравнению с линейным поиском. Причина заключается в разнице временной сложности этих алгоритмов

Линейный поиск имеет временную сложность это означает, что в худшем случае алгоритм должен проверить каждый элемент в массиве, что может быть очень медленно для больших массивов.

Бинарный поиск тоже имеет временную сложность это означает, что алгоритм делит массив пополам на каждом шаге, значительно сокращая количество необходимых проверок.

2) Оба алгоритма, линейный и бинарный поиск, используют примерно одинаковое количество памяти, поскольку они не требуют дополнительных структур данных для хранения промежуточных результатов. Однако в некоторых реализациях бинарный поиск может использовать немного меньше памяти, поскольку он не требует хранения индексов или дополнительных переменных для перебора всего массива.

3) time:

1.Не может измерять время выполнения многопоточных или многопроцессорных программ с учётом времени, проведённого в ожидании.

2.Точные измерения могут быть затруднены из-за контекстных переключений операционной системы.

memory\_profiler:

1.Использует дополнительную память для отслеживания, что может привести к изменению профиля памяти.

2.Не всегда точно отражает использование памяти для встроенных объектов или объектов, которые ведут себя по-разному в зависимости от окружения.

Вывод: Линейный поиск подходит для неотсортированных массивов или списков. Эффективен для небольших наборов данных, когда сортировка данных нецелесообразна. Бинарный поиск применим только к отсортированным массивам или спискам. Эффективен для больших наборов данных, так как значительно сокращает число необходимых сравнений.

Применение измерительных мониторов критически важно в системах реального времени, где необходимо минимизировать задержки, в критически важных приложениях, таких как медицинские устройства или финансовые системы, где ошибки могут привести к серьёзным последствиям и при разработке и тестировании программного обеспечения, чтобы обеспечить соблюдение временных ограничений и оптимизацию производительности.