Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра КСУП

Лабораторная работа №2

По дисциплине «Структуры данных»

Тема: «Двусвязный список. Сравнение времени работы операция вставки и удаления в двусвязном списке.»

Выполнил :
Студент гр. 583-1
___Дрягин Д.А.
Проверил:
Доцент КСУП
_____Калентьев А.А.

Томск 2024

Оглавление

Введение

Целью данной работы является разработка и исследование структуры данных «Двусвязный список», а также создание набора функций для работы с ней. Двусвязный список — это динамическая структура данных, которая позволяет эффективно выполнять операции вставки и удаления элементов, обеспечивая доступ как к предыдущему, так и к следующему элементу. Практическое применение двусвязных списков охватывает множество областей, таких как системы управления базами данных, операционные системы, компиляторы и другие системы, где необходимы быстрые операции вставки и удаления элементов.

Область исследования данной практики заключается в сравнении времени выполнения основных операций (вставка и удаление элементов) для различных размеров двусвязного списка.

В рамках работы будет проведено исследование производительности, в ходе которого будет измерено время выполнения операций для списков различной длины с использованием встроенных средств языка программирования. Для каждой операции будет рассчитана алгоритмическая сложность в нотации O(n), что позволит сделать выводы о преимуществах и недостатках данной структуры данных.

Описание структуры кода

1. **ListItem.h –** Содержит реализацию структур узла двусвязного списка и самого двусвязного списка.
2. **List.h и List.cpp** — содержат реализацию методов для работы со списком:
    - **Вставка элемента**.
    - **Удаление элемента по значение или по индексу.**
    - **Вставка в начало или конец**.
    - **Вставка перед или после заданного элемента.**
    - **Сортировка списка.**
    - **Линейный поиска**
3. **LaboratoryWork2.cpp** — предоставляет пользователю меню для выполнения различных операций над двусвязным списком:
    - Добавление в начало, конец или произвольную позицию,
    - Удаление элементов,
    - Линейный поиск.
    - Сортировка.
    - Измерение времени удаления или вставки.

Это позволяет гибко работать с данными, включая вставку или удаление элементов в любой части списка.

1 Основная часть

1.1. Теоретические основы

Двусвязный список состоит из узлов, каждый из которых содержит данные и два указателя: на предыдущий и следующий узел. Это позволяет легко перемещаться по списку в обоих направлениях и выполнять операции вставки и удаления с минимальными затратами времени.

1.2. Методика исследования

Для проведения эксперимента был реализован двусвязный список на языке C++. Измерение времени выполнения операций вставки и удаления осуществлялось с использованием стандартной библиотеки <chrono>. Были протестированы различные размеры списка: 500, 1000, 5000, 10000, 50000 и 100000 элементов.

1.3. Результаты измерений времени для вставки и удаления в двусвязном списке.

Размер списка	Время вставки	Время удаления
500	0.000093	0.000041
1000	0.0001	0.000066
5000	0.0004	0.00038
10000	0.0009	0.0007
50000	0.005	0.0047
100000	0.0096	0.01

На графиках ниже показано изменение времени выполнения операций удаления и вставки в зависимости от размера списка, а также общий график для вставки и удаления. Графики выполнены через библиотеку MatPlotLib на языке Python:

• Описание графика: На этом графике(Рисунок 1.1) представлено время, затраченное на вставку элементов в связный список в зависимости от его размера. Ось X отображает размер списка (количество элементов), а ось Y — время вставки в секундах.

• Наблюдения: Как видно из графика, время вставки увеличивается линейно с ростом размера списка. Это связано с тем, что вставка в начало списка имеет сложность O(1), но если мы рассматриваем вставку в конец списка без указателя на последний элемент, то она будет иметь сложность O(n), так как необходимо пройти весь список, чтобы дойти до конца списка.

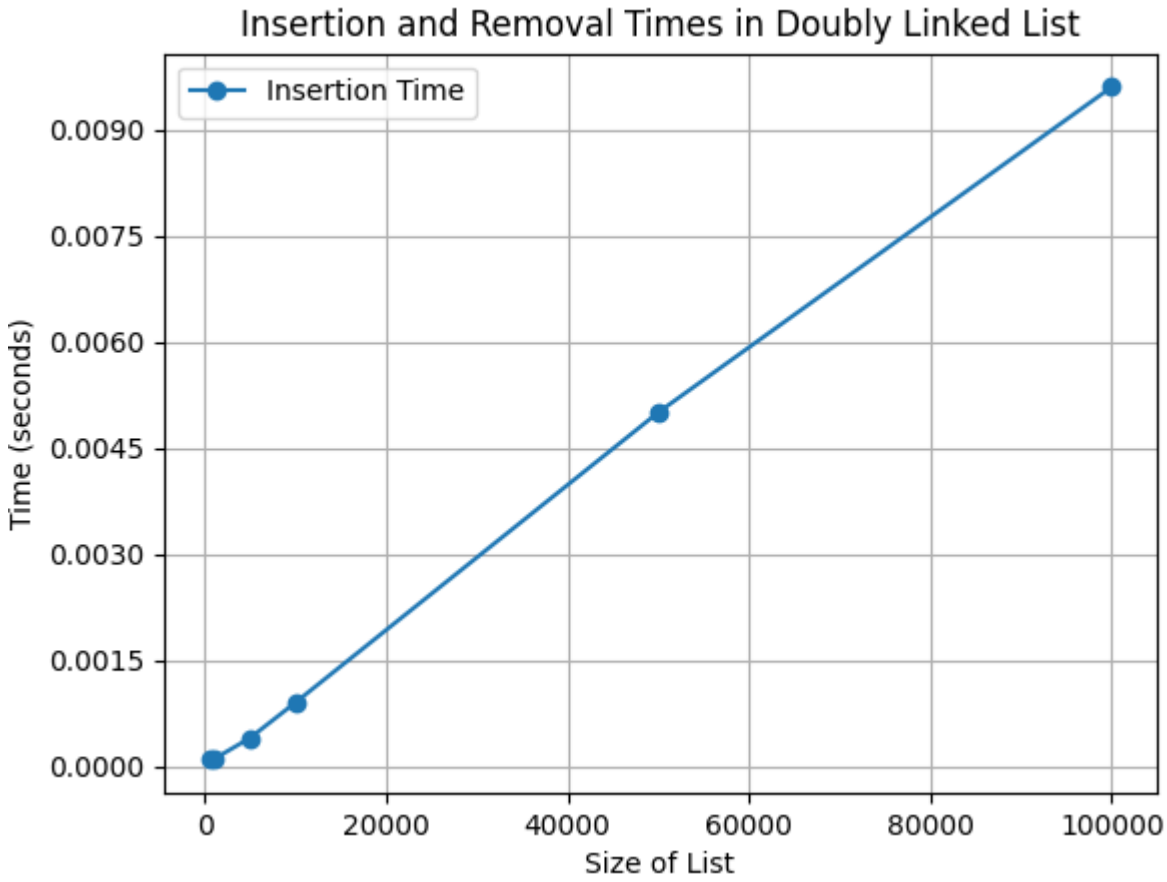

Рисунок 1.1 — График вставки для двусвязного списка

• Описание графика: На этом графике представлено время, затраченное на удаление элементов из связного списка в зависимости от его размера. Ось X показывает размер списка, а ось Y — время удаления в секундах.

• Наблюдения: Время удаления также увеличивается линейно с увеличением размера списка. Удаление из начала списка имеет сложность O(1), так как мы просто обновляем указатель на первый элемент. Однако, если мы рассматриваем удаление из конца списка без указателя на последний элемент, то это будет иметь сложность O(n), так как необходимо пройти весь список, чтобы найти предпоследний элемент.

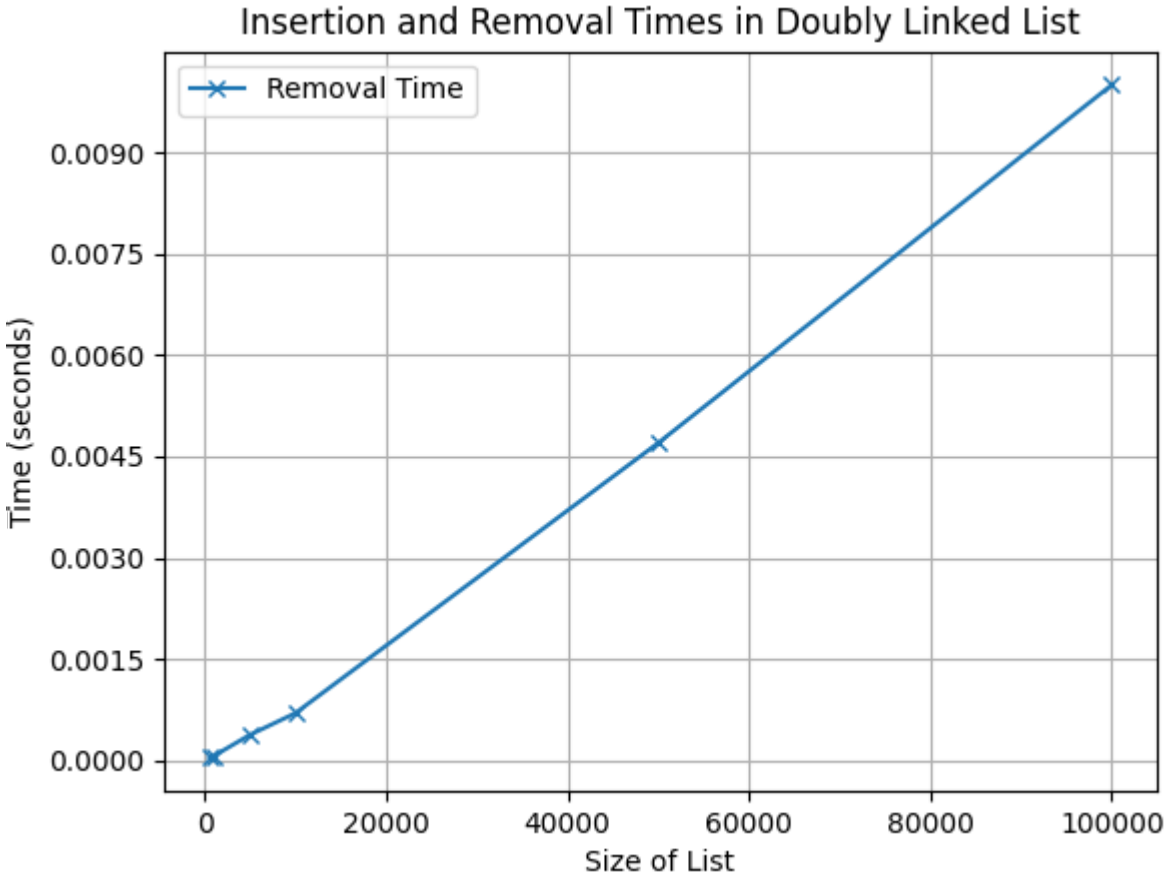

Рисунок 1.2 — График удаления элементов двусвязного списка

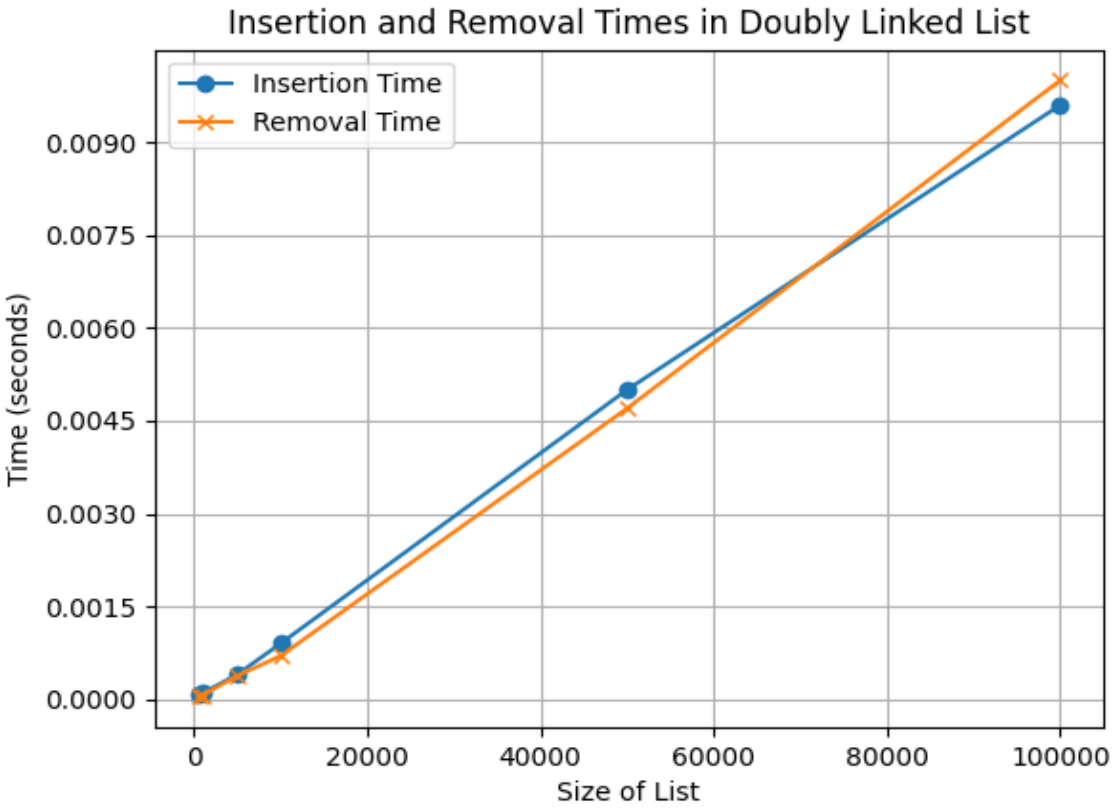

Рисунок 1.3 — Общий график вставки и удаления для двусвязного списка

1.4. Сложность алгоритмов

Теперь опишем сложность каждой операции:

1. Вставка элемента:

• Вставка в начало списка:

• Сложность: O(1)

• Объяснение: Мы создаем новый узел и перенаправляем указатель на голову списка. Эта операция не зависит от размера списка.

• Вставка в конец списка:

• Сложность: O(n)

• Объяснение: Если у нас нет указателя на последний элемент, нам нужно пройти весь список, чтобы добраться до конца и добавить новый узел.

2. Удаление элемента:

• Удаление из начала списка:

• Сложность: O(1)

• Объяснение: Мы просто обновляем указатель на голову списка, что занимает постоянное время.

• Удаление из конца списка:

• Сложность: O(n)

• Объяснение: Нам нужно пройти весь список, чтобы найти предпоследний элемент и удалить последний узел. Это требует времени, пропорционального размеру списка.

Заключение

В ходе работы было проведено исследование временных характеристик операций вставки и удаления в двусвязном списке. Результаты показали, что время выполнения обеих операций увеличивается линейно с увеличением размера списка. Время вставки оказалось несколько меньшим по сравнению с временем удаления, что согласуется с теоретическими ожиданиями.

Данная работа позволила лучше понять работу двусвязных списков и их применение в программировании.