Лекция 9. Стек и очередь в связанных списках

Казахский национальный университет имени аль-Фараби

Кафедра компьютерных наук

Алгоритмы и структуры данных

Лектор: старший преподаватель, PhD, Кенжебек Ержан Ғалымжанұлы

Цель и план лекции

Цель лекции

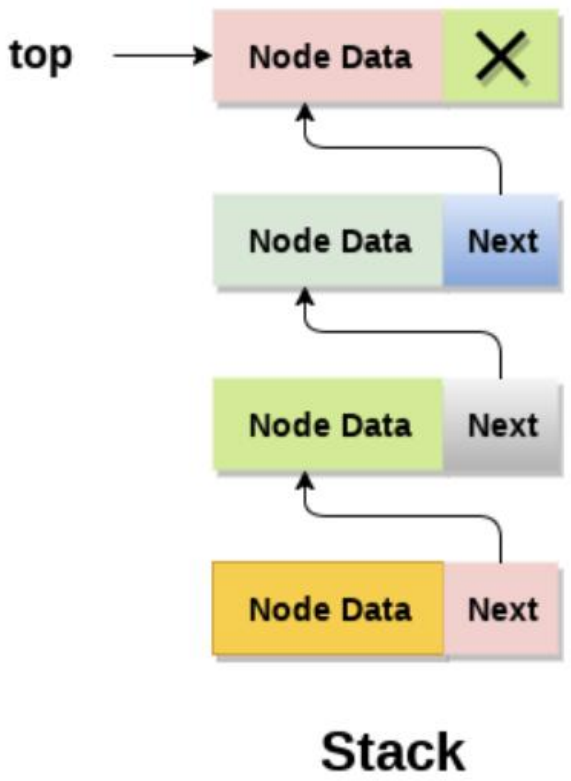
Изучить реализацию стека на основе связанного списка, операции Push, Pop и обход узлов, оценить сложность и рассмотреть области применения стека.

План лекции

1. Структура стека на связанном списке
2. Добавление узла (Push)
3. Удаление узла (Pop)
4. Обход и отображение узлов
5. Анализ сложности и преимущества
6. Примеры применения стека

Реализация стека связанного списка

- Вместо использования массива мы также можем использовать **связанный список** для реализации стека. Связанный список распределяет память динамически. Однако временная сложность в обоих сценариях одинакова для всех операций, т. е. push, pop и peek.
- В реализации стека со связным списком узлы хранятся в памяти несмежными. Каждый узел содержит указатель на свой непосредственный узел-преемник в стеке. Говорят, что стек переполнен, если места, оставшегося в куче памяти, недостаточно для создания узла.
- Самый верхний узел в стеке всегда содержит значение null в поле адреса. Давайте обсудим, как каждая операция выполняется в реализации стека со связным списком.

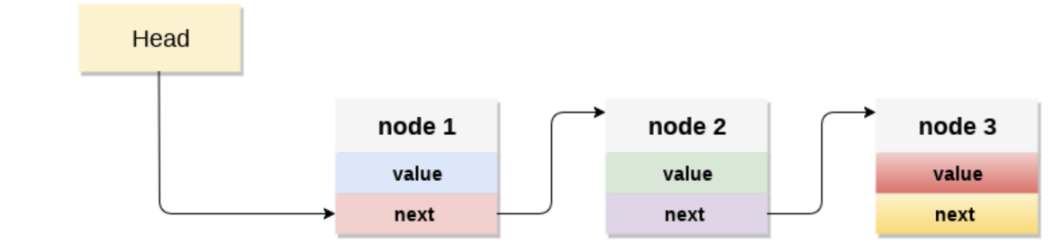


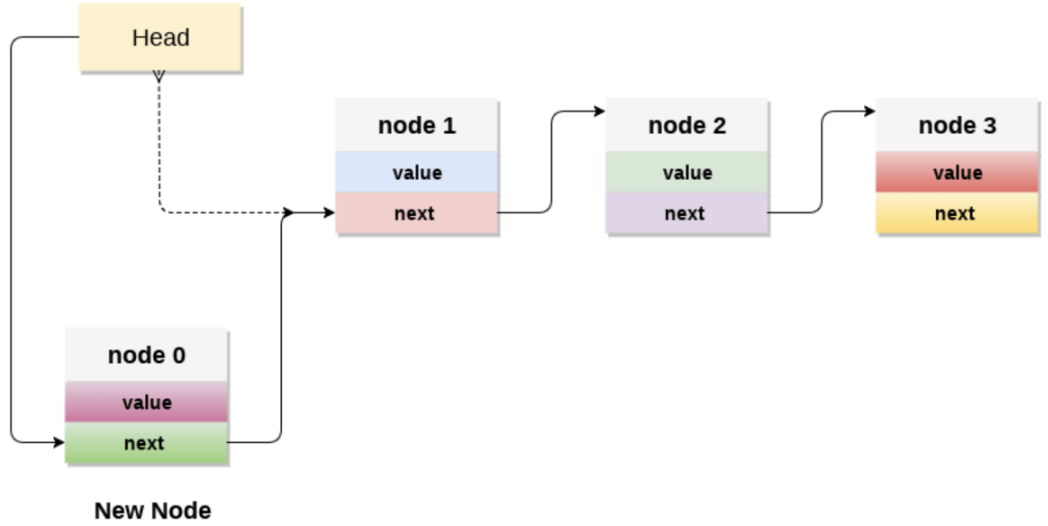
Добавление узла в стек (операция Push)

- Добавление узла в стек называется операцией **push** . Помещение элемента в стек в реализации связанного списка отличается от реализации массива. Чтобы поместить элемент в стек, необходимо выполнить следующие шаги.
    1. Сначала создайте узел и выделите для него память.
    2. Если список пуст, то элемент должен быть помещен в качестве начального узла списка. Это включает в себя присвоение значения части данных узла и присвоение нуля адресной части узла.
    3. Если в списке уже есть какие-то узлы, то нам нужно добавить новый элемент в начало списка (чтобы не нарушать свойства стека). Для этого присвойте адрес стартового элемента полю адреса нового узла и сделайте новый узел стартовым узлом списка.
-

Иллюстрация

Временная сложность: o(1)





Удаление узла из стека (операция POP)

- Удаление узла из вершины стека называется операцией **извлечения (pop)** . Удаление узла из реализации стека в связанном списке отличается от реализации в виде массива. Чтобы извлечь элемент из стека, нам нужно выполнить следующие шаги:
    1. **Проверьте состояние опустошения:** состояние опустошения возникает, когда мы пытаемся извлечь данные из уже пустого стека. Стек будет пустым, если указатель заголовка списка указывает на ноль.
    2. **Отрегулируйте указатель head соответствующим образом:** в стеке элементы извлекаются только с одного конца, поэтому значение, хранящееся в указателе **head**, должно быть удалено, а узел должен быть освобожден. Следующий узел головного узла теперь становится головным узлом.

Отображение(display) узлов (traversing)

- Для отображения всех узлов стека необходимо пройти по всем узлам связанного списка, организованного в виде стека. Для этого нам необходимо выполнить следующие шаги.
  1. Скопируйте указатель головы (head) во временный указатель.
  2. Переместите временный указатель по всем узлам списка и распечатайте поле значения, прикрепленное к каждому узлу.

- **Временная сложность: O(n)**

Анализ сложности

- **Временная сложность:** O(1) для всех push(), pop() и peek(), поскольку мы не выполняем никакого обхода по списку. Все операции мы выполняем только через текущий указатель.
  Вспомогательное пространство: O(N), где N — размер стека.
- В этой реализации мы определяем класс Node, который представляет узел в связанном списке, и класс Stack, который использует этот класс узла для реализации стека. Атрибут head класса Stack указывает на вершину стека (т. е. первый узел в связанном списке).
- Чтобы поместить элемент в стек, мы создаем новый узел с данным элементом и устанавливаем его следующий указатель на текущую главу стека. Затем мы устанавливаем начало стека новому узлу, фактически делая его новой вершиной стека.
- Чтобы извлечь элемент из стека, мы просто удаляем первый узел из связанного списка, устанавливая заголовок стека на следующий узел в списке (т. е. узел, на который указывает следующий указатель текущей головы). Мы возвращаем данные, хранящиеся в исходном головном узле, который представляет собой элемент, удаленный из вершины стека.

Преимущества

- **Преимущества реализации стека с использованием односвязного списка включают в себя:**
  - **Динамическое распределение памяти** . Размер стека можно увеличивать или уменьшать динамически путем добавления или удаления узлов из связанного списка без необходимости заранее выделять фиксированный объем памяти для стека.
  - **Эффективное использование памяти.** Поскольку узлы в односвязном списке имеют только указатель «следующий», а не «предыдущий», они используют меньше памяти, чем узлы в двусвязном списке.
  - **Простая реализация** . Реализация стека с использованием односвязного списка проста и может быть выполнена с использованием всего нескольких строк кода.
  - **Универсальность** : односвязные списки можно использовать для реализации других структур данных, таких как очереди, связанные списки и деревья.
  - Таким образом, реализация стека с использованием односвязного списка — это простой и эффективный способ создания динамической структуры данных стека в Python.

Примеры стека в реальном времени

- Стеки используются в различных реальных сценариях, где требуется структура данных «последним пришел — первым обслужен» (LIFO). Вот несколько примеров применения стеков в реальном времени:
- **Стек вызовов функций** . Когда функция вызывается в программе, адрес возврата и все параметры функции помещаются в стек вызовов функций. Стек позволяет функции выполняться и возвращаться к вызывающей функции в порядке, обратном их вызову.
- **Операции отмены/повтора.** Во многих приложениях, таких как текстовые редакторы, редакторы изображений или веб-браузеры, функции отмены и повтора реализуются с использованием стека. Каждый раз, когда выполняется действие, оно помещается в стек. Когда пользователь хочет отменить последнее действие, верхний элемент стека извлекается, и действие отменяется.
- **История браузера.** Веб-браузеры используют стеки для отслеживания посещенных пользователем страниц. Каждый раз, когда посещается новая страница, ее URL-адрес помещается в стек. Когда пользователь нажимает кнопку «Назад», последний посещенный URL-адрес извлекается из стека, и пользователь перенаправляется на предыдущую страницу.
- **Оценка выражений** . Стеки используются в компиляторах и интерпретаторах для оценки выражений. Когда выражение анализируется, оно преобразуется в постфиксную запись и помещается в стек. Затем постфиксное выражение оценивается с использованием стека.
- **Стек вызовов в рекурсии:** когда вызывается рекурсивная функция, ее вызов помещается в стек. Функция выполняется и вызывает саму себя, и каждый последующий вызов помещается в стек. Когда рекурсия заканчивается, стек извлекается, и программа возвращается к предыдущему вызову функции.
- Таким образом, стеки широко используются во многих приложениях, где требуются функции LIFO, такие как вызовы функций, операции отмены/повтора, история браузера, оценка выражений и рекурсивные вызовы функций.

Пример реализации



stack_by_linked_list.cpp

Output

```
44 -> 33 -> 22 -> 11
Top element is 44
22 -> 11
Top element is 22
```

Выводы

1. В реализации на списке узлы хранятся в памяти несмежно, а указатель head указывает на вершину стека.
2. Push создаёт новый узел и делает его вершиной; Pop проверяет состояние опустошения, сдвигает head и освобождает узел.
3. Операции push, pop и peek выполняются за O(1), обход всех узлов — за O(n).
4. Размер стека изменяется динамически и ограничен лишь свободной памятью кучи.
5. Стек применяется для вызовов функций, рекурсии, отмены/повтора действий и вычисления выражений.

Контрольные вопросы

1. Как устроен узел стека, реализованного на связанном списке?
2. Опишите шаги операции Push.
3. Как выполняется Pop и когда возникает состояние опустошения?
4. Какова сложность операций push, pop, peek и обхода стека?
5. Какие преимущества даёт реализация стека на списке по сравнению с массивом?
6. Приведите примеры практического применения стека.

Список литературы

Основная литература

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. 3-е изд. — М.: Вильямс, 2013.
2. Вирт Н. Алгоритмы и структуры данных. — М.: ДМК Пресс, 2010.
3. Седжвик Р., Уэйн К. Алгоритмы на Java. 4-е изд. — М.: Вильямс, 2013.

Дополнительная литература

4. Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. — Addison-Wesley, 1983.
5. Страуструп Б. Программирование: принципы и практика с использованием C++. 2-е изд. — М.: Вильямс, 2016.
6. Бхаргава А. Грокаем алгоритмы. — СПб.: Питер, 2017.

Электронные ресурсы

7. VisuAlgo — визуализация алгоритмов и структур данных: visualgo.net
8. Справочник по C++ и STL: en.cppreference.com