НИУ ВШЭ - Нижний Новгород

December 6, 2024

Алгоритмы и структуры данных

Лекция 5. Простые структуры данных

Илья Сергеевич Бычков
ibychkov@hse.ru

Лекция 5.
Простые структуры данных

Простые структуры данных

План лекции

0. План лекции
1. Связный список (linked list)
2. Стек (stack)
3. Очередь (queue)

Связный список

Связный список

Связный список (linked list) - это структура данных, состоящая объектов специального вида, которые называются узлами (nodes). Узлы хранят сами данные и связаны друг с другом с помощью указателей.

Каждый узел содержит одно или несколько полей для хранения данных

Каждый узел содержит указатель на следующий/предыдущий узел

Требуются указатели на первый/последний элементы списка

Связный список
Структура node

Связный список (linked list)

```c
typedef struct node {
    int val;
    struct node * next;
} node_t;
```

(1) Структура данных node, Источник - Ссылка

```c
node_t * head = NULL;
head = malloc(sizeof(node_t));
if (head == NULL) {
    return 1;
}

head->val = 1;
head->next = NULL;
```

(2) Создание node, Источник - Ссылка

Связный список
Операции

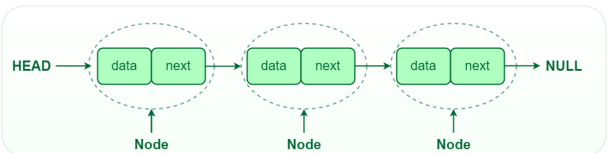
Основные операции

Вставка и удаление

- В начало/конец списка
- До/после определенного значения
- До/после определенного адреса

Поиск

- По значению
- По позиции



(3) Связный список, Источник - Geeks4Geeks

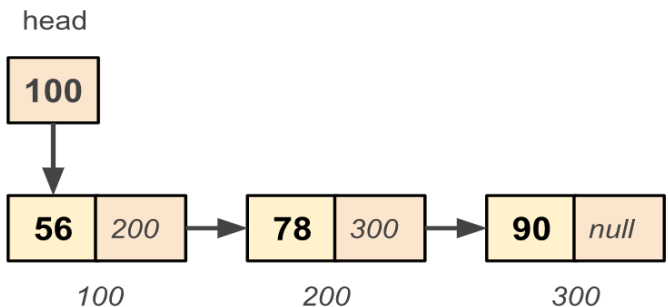
Перебор элементов связного списка

```c
void print_list(node_t * head) {
    node_t * current = head;

    while (current != NULL) {
        printf("%d\n", current->val);
        current = current->next;
    }
}
```

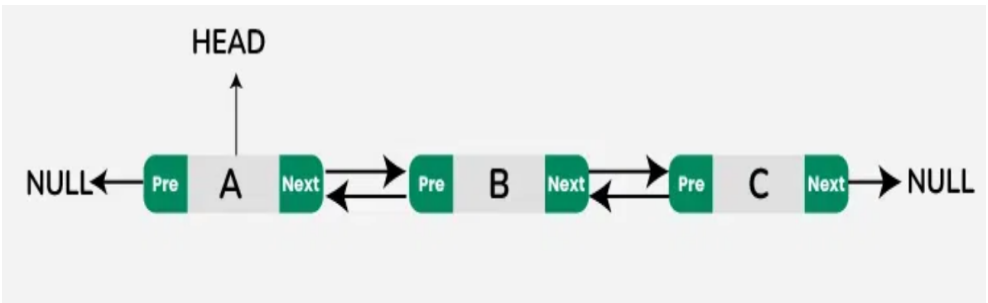
(4) Перебор элементов связного списка, Источник - Ссылка

Односвязный список (singly linked list) - состоит из узлов, которые хранят полезные данные и указатель на следующий



(5) Односвязный список, Источник - Ссылка

Двусвязный список (doubly linked list) - состоит из узлов, которые хранят полезные данные, указатели на предыдущий узел и следующий узел

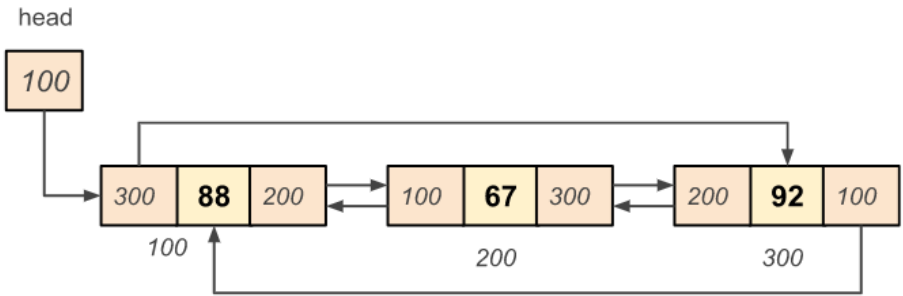


(6) Двусвязный список, Источник - Geeks4Geeks

Связный список

Вариации связных списков

Кольцевой связный список (circular linked list) - разновидность связного списка, при которой первый элемент указывает на последний, а последний — на первый



(7) Односвязный список, Источник - Ссылка

Связный список

Сложность операций

Односвязный список (singly linked list)

Тип операции	Только Head	Head + Tail
Взять первый	O(1)	
Взять последний	O(n)	O(1)
Вставка в начало	O(1)	
Вставка в конец	O(n)	O(1)
Вставка до заданного	O(n)	
Вставка после заданного	O(1)	
Удаление первого	O(1)	
Удаление последнего	O(n)	O(n)
Удаление по значению	O(n)	
Поиск элемента	O(n)	

(8) Односвязный список - сложность, Источник - Этот курс

Связный список

Сложность операций

Двусвязный список (doubly linked list)

Тип операции	Только head	Head + Tail
Взять первый	O(1)	
Взять последний	O(n)	O(1)
Вставка в начало	O(1)	
Вставка в конец	O(n)	O(1)
Вставка до заданного	~~O(n)~~ O(1)	
Вставка после заданного	O(1)	
Удаление первого	O(1)	
Удаление последнего	O(n)	~~O(n)~~ O(1)
Удаление по значению	O(n)	
Поиск элемента	O(n)	

(9) Двусвязный список - сложность, Источник - Этот курс

Связные списки - Преимущества и недостатки

Переполнение невозможно, только если закончилась память

Простые и эффективные операции вставки и удаления

При работе с большими объектами перемещение указателей проще чем копирование

Использую дополнительную память для указателей

Нет эффективного произвольного доступа

Менее эффективны в использовании кэш-памяти

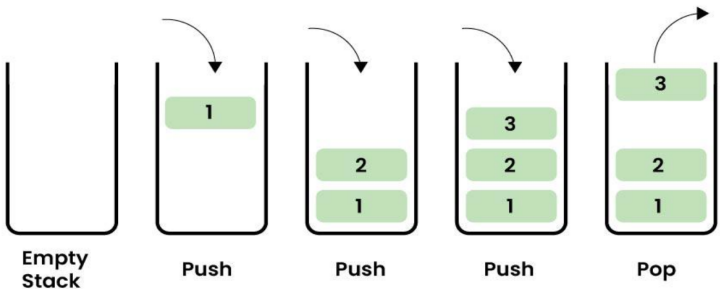
(10) Связные списки - Преимущества и недостатки, Источник - Этот курс

Простые структуры данных

План лекции

0. План лекции
1. Связный список (linked list)
2. Стек (stack)
3. Очередь (queue)

Стек (Stack) - это структура данных, работающая по принципу “последним добавлен – первым возвращен” – Last In First Out (LIFO)



(11) Принцип LIFO, Источник - Geeks4Geeks

Стек (Stack)

```
// A structure to represent a stack
struct Stack {
    int top;
    unsigned capacity;
    int* array;
};
```

(12) Стек, реализация на массиве, Источник - Этот курс

Стек

Операции

Основные операции

Доступ

- Получение последнего элемента без удаления

Вставка

- В конец стека

Удаление

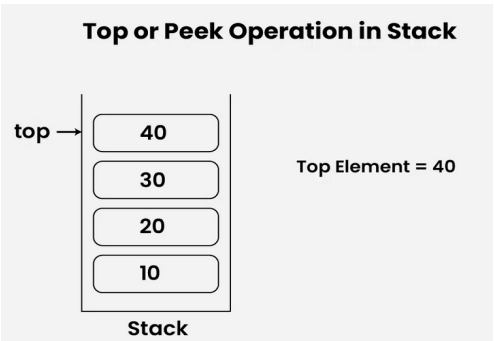
- Получение последнего элемента с удалением

Дополнительно

- Пуст ли стек?
- Заполнен ли стек?

Доступ

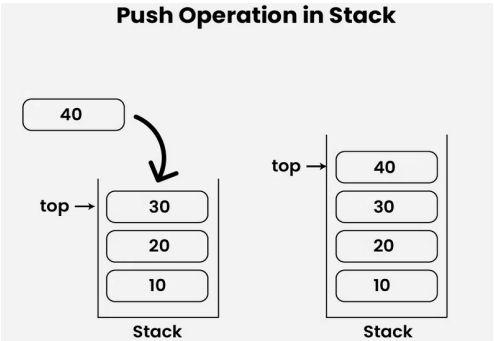
- Получение последнего элемента без удаления



(13) Операция peek, Источник - Geeks4Geeks

Вставка

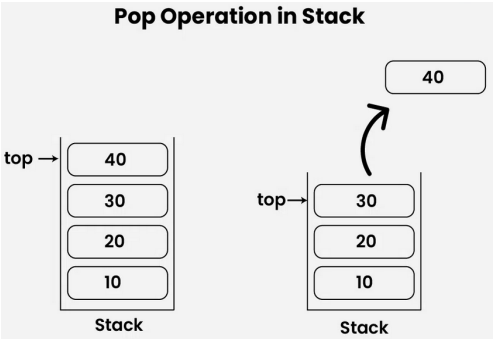
- В конец стека



(14) Операция push, Источник - Geeks4Geeks

Удаление

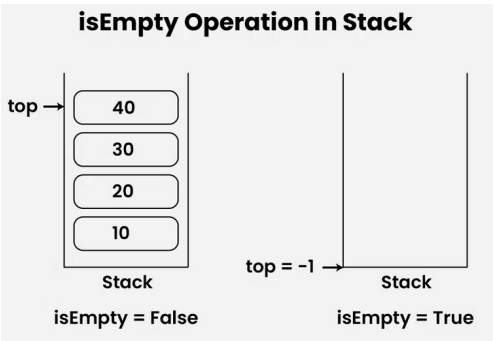
- Получение последнего элемента с удалением



(15) Операция pop, Источник - Geeks4Geeks

Дополнительно

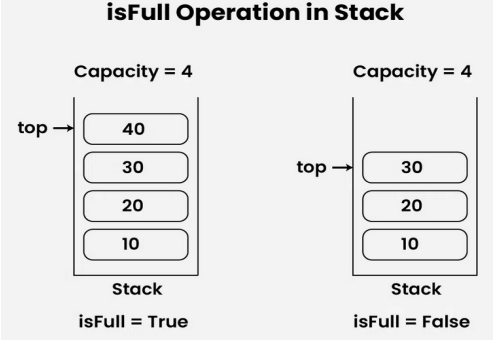
- Пуст ли стек?



(16) Операция isEmpty, Источник - Geeks4Geeks

Дополнительно

- Полон ли стек?



(17) Операция isFull, Источник - Geeks4Geeks

Пример использования

```
struct Stack* stack = createStack(100);

push(stack, 10);
push(stack, 20);
push(stack, 30);

if (!isEmpty(stack))
    printf("%d popped from stack\n", pop(stack));
```

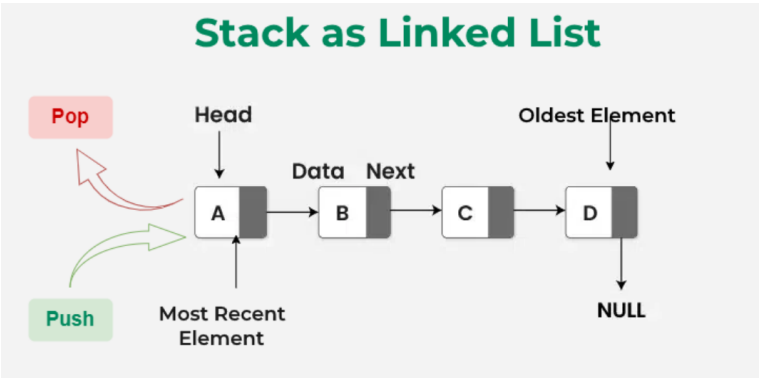
(18) Операция isFull, Источник - Geeks4Geeks

Стек (Stack)

```
typedef struct Node {
    int data;
    struct Node* next;
} node;
```

(19) Стек, реализация на списке, Источник - Этот курс

Стек (Stack)



(20) Стек, реализация на списке, Источник - Geeks4Geeks

Стек (Stack)

Тип операции	Stack
Получить элемент	O(1)
Вставка в конец	O(1)
Удаление последнего	O(1)
Проверка пустоты	O(1)
Проверка заполненности	O(1)

(21) Стек, сложность, Источник - Этот курс

Задачи на использование стека

Определение правильности скобочной последовательности

Дана строка состоящая из скобок разных видов ( ) [ ] { }. Необходимо сообщить, является ли данная скобочная последовательность правильной.

()()()[][]{()} – правильная последовательность

(((()))){{}}[{}] – правильная последовательность

{[)} – неправильная последовательность

}{)(– неправильная последовательность

Простые структуры данных

План лекции

0. План лекции
1. Связный список (linked list)
2. Стек (stack)
3. Очередь (queue)

Очередь (Queue) - это структура данных, работающая по принципу “последним добавлен – первым возвращен” – Last In First Out (LIFO)

(22) Принцип FIFO, Источник - Geeks4Geeks

Очередь

Очередь

Основные операции

Доступ

- Получение первого элемента без удаления
- Получение последнего элемента без удаления

Вставка

- В конец очереди

Удаление

- Удаление из начала очереди

Дополнительно

- Пуста ли очередь?
- Заполнена ли очередь?

Очередь (Queue)

```
// A structure to represent a queue
struct Queue
{
    int front, rear, size;
    unsigned capacity;
    int* array;
};
```

(23) Очередь - реализация на массиве, Источник - Geeks4Geeks

Операция создания очереди

```
// function to create a queue
// of given capacity.
// It initializes size of queue as 0
struct Queue* createQueue(unsigned capacity)
{
    struct Queue* queue = (struct Queue*)malloc(
        sizeof(struct Queue));
    queue->capacity = capacity;
    queue->front = queue->size = 0;

    // This is important, see the enqueue
    queue->rear = capacity - 1;
    queue->array = (int*)malloc(
        queue->capacity * sizeof(int));
    return queue;
}
```

(24) Создание очереди - реализация на массиве, Источник - Geeks4Geeks

Операция enqueue

```
// Function to add an item to the queue.
// It changes rear and size
void enqueue(struct Queue* queue, int item)
{
    if (isFull(queue))
        return;
    queue->rear = (queue->rear + 1)
                  % queue->capacity;
    queue->array[queue->rear] = item;
    queue->size = queue->size + 1;
    printf("%d enqueued to queue\n", item);
}
```

(25) Enqueue - реализация на массиве, Источник - Geeks4Geeks

Очередь

Реализация на связном списке

```c
// Define the structure for a node of the linked list
typedef struct Node {
    int data;
    struct Node* next;
} node;

// Define the structure for the queue
typedef struct Queue {
    node* front;
    node* rear;
} queue;
```

(26) Очередь на связном списке, Источник - Geeks4Geeks

Очередь

Пример использования

```
struct Queue* queue = createQueue(1000);

enqueue(queue, 10);
enqueue(queue, 20);
enqueue(queue, 30);
enqueue(queue, 40);

printf("%d dequeued from queue\n\n", dequeue(queue));

printf("Front item is %d\n", front(queue));
printf("Rear item is %d\n", rear(queue));
```

(27) Очередь - пример, Источник - Geeks4Geeks

Очередь

Сложность операци (время)

Operations	Complexity
Enqueue(insertion)	O(1)
Deque(deletion)	O(1)
Front(Get front)	O(1)
Rear(Get Rear)	O(1)
IsFull(Check queue is full or not)	O(1)
IsEmpty(Check queue is empty or not)	O(1)

(28) Сложность операций с очередью, Источник - Geeks4Geeks