ЛЕКЦИЯ 05
СПИСКИ

АЛГОРИТМИЗАЦИЯ И
ПРОГРАММИРОВАНИЕ

ЛЕКТОР ФУРМАВНИН С.А.

ПОНЯТИЕ СПИСКА

Связный список — структура данных, состоящая из элементов, содержащих помимо собственных данных ссылки на следующий и/или предыдущий элемент списка.

ИДЕЯ ОДНОСВЯЗНОГО СПИСКА

Идея односвязного списка довольно проста: каждый узел содержит указатель на следующий

```
struct ListNode {
      int val;
      ListNode *next;
      ListNode(int x) : val(x), next(nullptr) {}
};
```

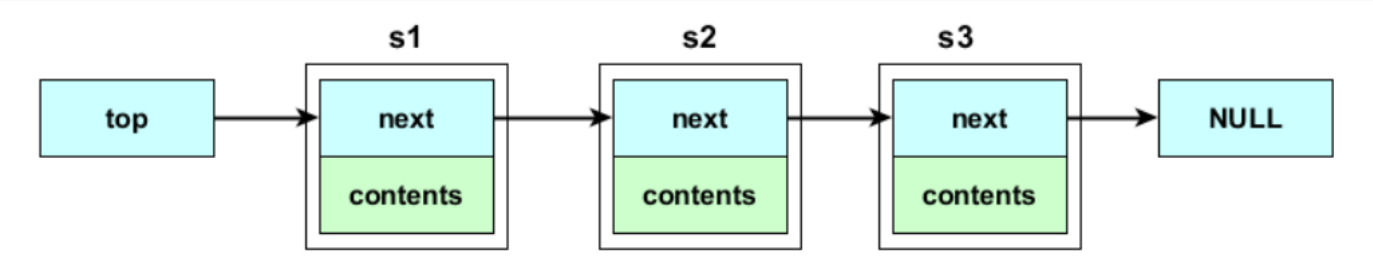
Узлы такого рода динамических структур данных обычно выделяются в куче.

```
ListNode *top = new ListNode(1);
top->next = new ListNode(2);
```

ИДЕЯ ОДНОСВЯЗНОГО СПИСКА

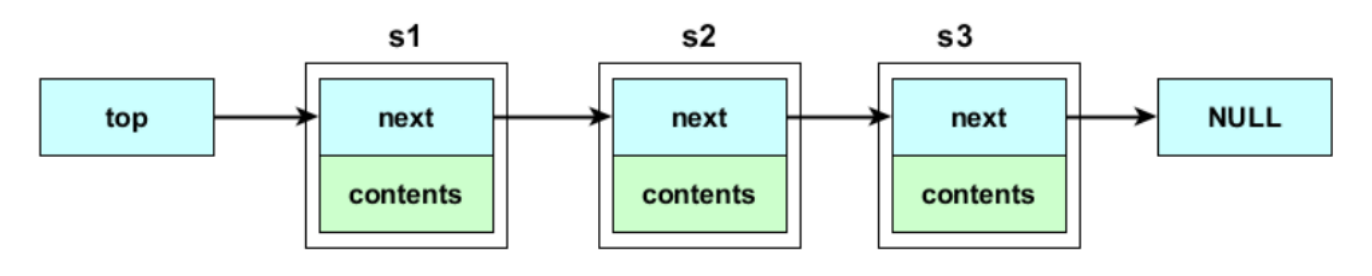
Идея односвязного списка довольно проста: каждый узел содержит указатель на следующий

```
struct ListNode {
      int val; ListNode *next;
      ListNode(int x) : val(x), next(nullptr) {}
};
```



ИДЕЯ ОДНОСВЯЗНОГО СПИСКА

Идея односвязного списка довольно проста: каждый узел содержит указатель на следующий



Динамические структуры данных крайне полезны

По традиции мы начнём их изучение со списков (хотя они как раз сами по себе не слишком полезны, зато очень просты)

ЗАДАЧА. РАБОТА СО СПИСКАМИ

- Ваша задача: написать две функции
- Построить односвязный список из файлового ввода, такой, что все чётные числа идут в начале, а все нечётные в конце

```
ListNode *read_list(FILE *inp);
```

- Файл представляет собой просто целые числа разделённые через пробел

```
1 65 78 2 34
```

- Удалить односвязный список, получив указатель на первый элемент

```
void delete_list(ListNode *top);
```

ИНТЕРФЕЙС ЦИКЛИЧЕСКОГО СПИСКА

Основная структура ничем не отличается от обычного односвязного списка

```
struct ListNode {
     int val; ListNode *next;
     ListNode(int x) : val(x), next(nullptr) {}
};
```

Основные операции:

```
// добавить элемент после указанного
ListNode *cl_push(ListNode *pre, int d);
// удалить элемент после указанного
ListNode *cl_pop(ListNode *pre);
```

ЗАДАЧА. ЦИКЛИЧЕСКИЙ РАЗМЕН

- Входной файл содержит сумму для размена, количество номиналов и все номиналы размена
- Ваша задача считав его со стандартного ввода вывести на стандартный вывод правильное количество использованных монет
- Пример. Входные данные:

```
11 3 1 3 4
```

- Выход:

```
3
```

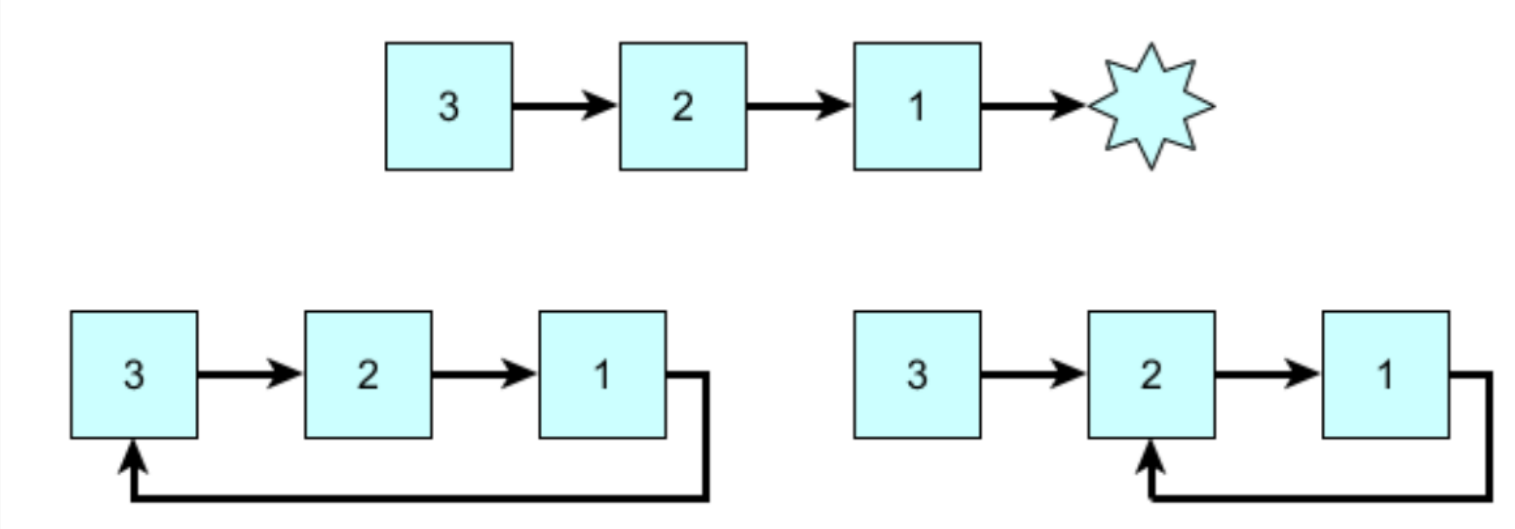
- Сумма для размена может быть очень большой, используйте циклический буфер

ОБСУЖДЕНИЕ

- Тот факт, что циклический список мало чем отличается от обычного должен, в принципе, вас насторожить

ОБСУЖДЕНИЕ

- Тот факт, что циклический список мало чем отличается от обычного должен, в принципе, вас насторожить



ЗАДАЧА. ЕСТЬ ЛИ ПЕТЛЯ?

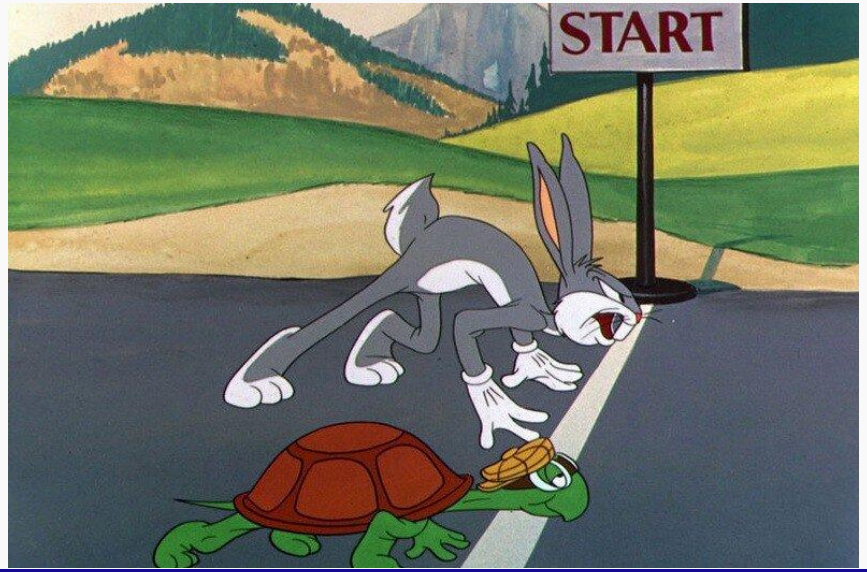
На входе есть односвязный список:

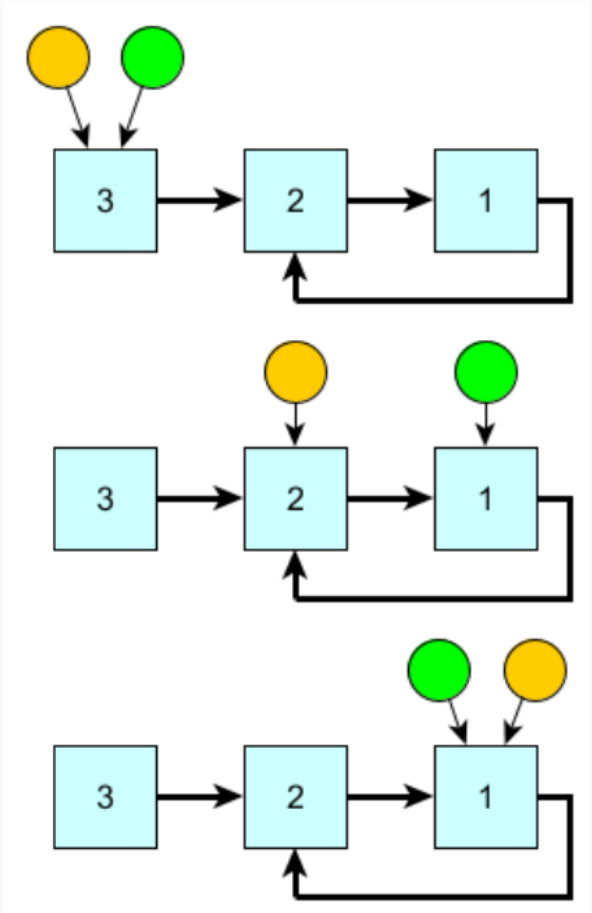
```
struct ListNode {
     int val; ListNode *next;
     ListNode(int x) : val(x), next(NULL) {}
};

int somefunc(ListNode* top);
```

Возможно, в нем есть петля. Возможно он заканчивается нулевым указателем. Как это определить?

АЛГОРИТМ ФЛОЙДА

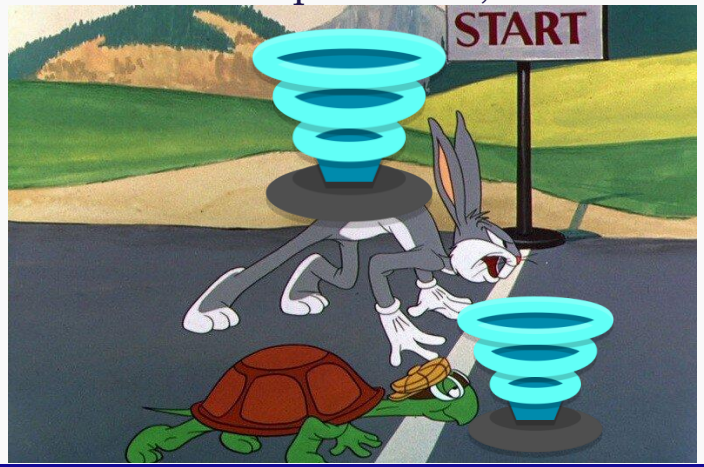
- Начинают два указателя: заяц и черепаха
- Заяц за каждый ход продвигается на два элемента вперед, а черепаха – на один.
- Если они встретились, значит есть петля.

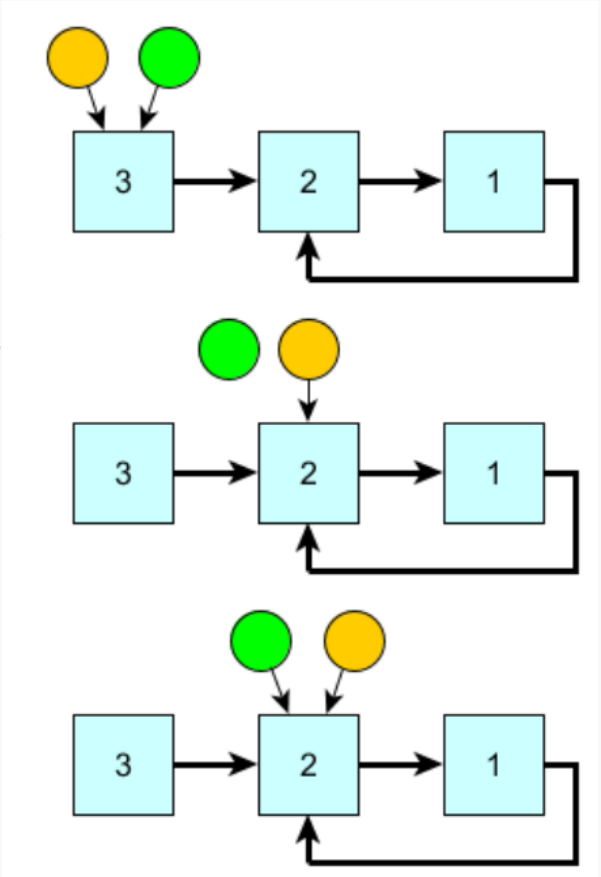




АЛГОРИТМ БРЕНТА

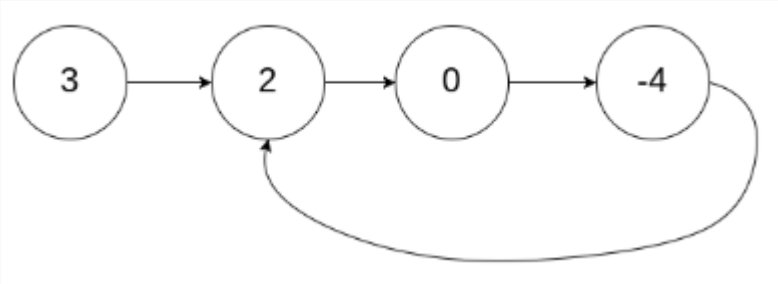
- Начинают два указателя: заяц и черепаха
- Заяц за каждый ход продвигается на единицу и несет с собой телепорт.
- Как только он прошел очередную степень двойки, он телепортирует туда черепаху.
- Если они встретились, значит есть петля.





ЗАДАЧА ТРЕНИРОВОЧНАЯ. ЕСТЬ ЛИ ПЕТЛЯ

Вам известно начало списка A. Напишите функции, реализующие алгоритм Флойда и алгоритм Брента, возвращающие true (есть петля) или false (нет петли). Например:



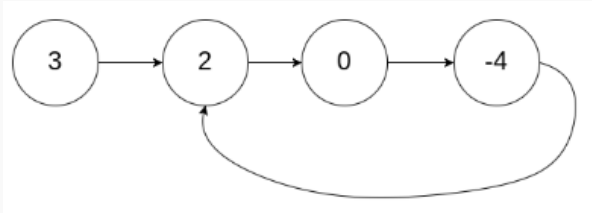
Какая асимптотика у вашего алгоритма?

ЗАДАЧА ТРЕНИРОВОЧНАЯ. ДЛИНА ПЕТЛИ

- Вам известно начало списка A. Вы должны написать функцию, которая определяет длину петли если она есть и возвращает 0, если её нет

```
int loop_len(ListNode *top);
```

- Попробуйте адаптировать один из ранее приведённых алгоритмов для этого
- Разумное решение не будет использовать много дополнительной памяти Например:



Какая асимптотика у вашего алгоритма?

ЗАДАЧА ТРЕНИРОВОЧНАЯ. ПЕРИОД ГЕНЕРАТОРА

- Вам дан генератор $x_{j+1} = f(x_j)$, при этом вы не знаете функцию f
- Генерируемые числа выглядят случайными. Вам нужно написать функцию, которая ищет длину цикла в генераторе, начиная с $x_0 = 0$

```
typedef int (*generator_t)(int);
unsigned cycle_len(generator_t gen) {
// TODO: ваш код здесь }
```

- Например при $f(x) = (x + 2)\%5, x_0 = 0$ генерируется последовательность 0, 2, 4, 1, 3, 0, 2, ... и длина цикла равна 5
- Заметьте, вовсе не факт, что x_0 встретится ещё раз (генератор может быть смещённым). Но для любого $j > i, x_j = x_i$ означает цикл

АЛГОРИТМ RFL – РЕКУРСИВНЫЙ РАЗВОРОТ

- Основная идея в том, что `reverse(x:xs) = reverse(xs):x`

```
ListNode * reverse(ListNode *top) {
  ListNode *xs;
  if (nullptr == top) return nullptr;
  if (nullptr == top->next) return top;
  xs = reverse(top->next);
  top->next->next = top; // единственное тонкое место
  top->next = NULL;
  return xs;
}
```

- Алгоритм сравнительно прост и красив. Увы достаточно длинный список переполнит стек

ЗАДАЧА. ИТЕРАТИВНЫЙ РАЗВОРОТ

- Это ещё одна обычная проблема вида «рекурсия-в-итерацию»
- Ваша задача написать функцию разворота односвязного списка, которая будет работать за $O(1)$ по памяти

```
ListNode *reverse(ListNode *top);
```

- Заметьте, что алгоритм RFL тратит $O(N)$ памяти на стек вызовов

ОБСУЖДЕНИЕ

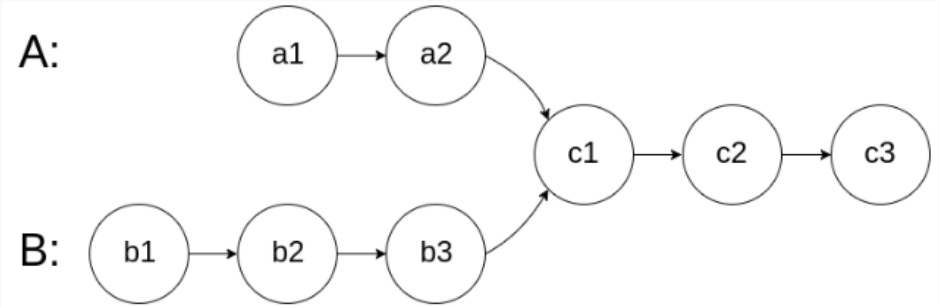
- Что если в вашем входном односвязном списке всё же появится петля?
- Попробуйте прогнать вашу процедуру переворота на зацикленном списке и посмотрите результат
- Интересный вопрос: что же делать?

ОБСУЖДЕНИЕ

- Что если в вашем входном односвязном списке всё же появится петля?
- Попробуйте прогнать вашу процедуру переворота на зацикленном списке и посмотрите результат
- Интересный вопрос: что же делать?
- Плохой вариант: при каждом вызове переворота искать петлю. Это практически удваивает время на переворот
- Хороший вариант несколько менее очевиден. И он называется инкапсуляция

ЗАДАЧА 1

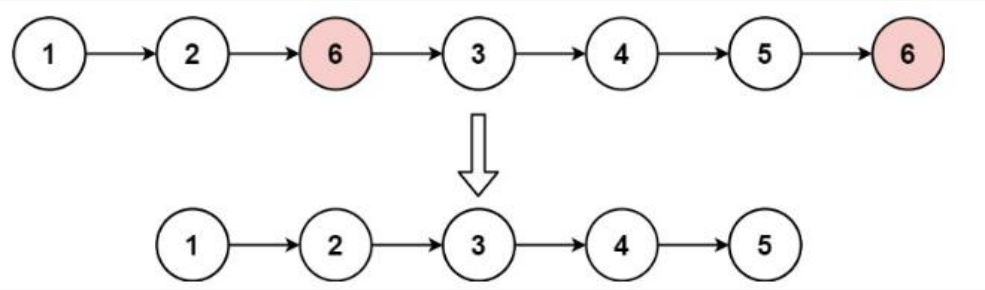
Вам известно начало двух списков A и B. Эти списки, начиная с некоторого элемента имеют общий «хвост». Например:



Напишите функцию, которая принимает два указателя на начало списков A и B и возвращает значение элемента, начиная с которого оба списка объединились (значение элемента c1 в примере). Какая асимптотика у вашего алгоритма?

ЗАДАЧА 2

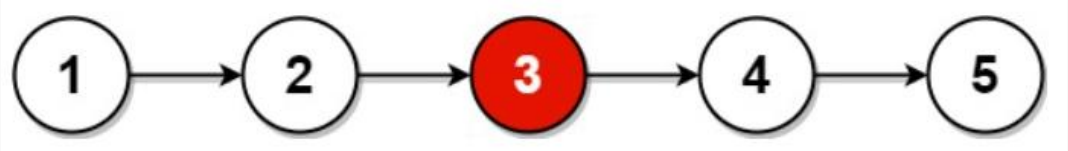
Вам известно начало списка A и дано значение n. Напишите функцию, которая удаляет из списка A этот элемент. Например:



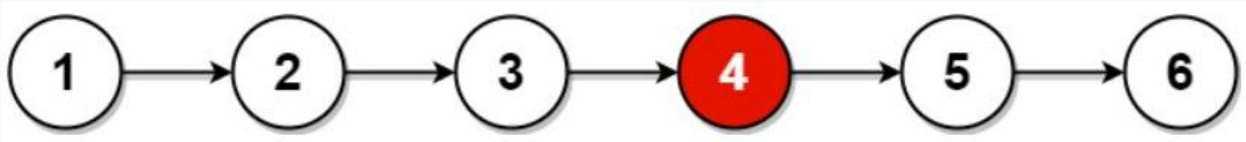
Какая асимптотика у вашего алгоритма?

ЗАДАЧА 3

Вам известно начало списка A. Напишите функцию, которая возвращает значение, находящееся в середине списка. Например:



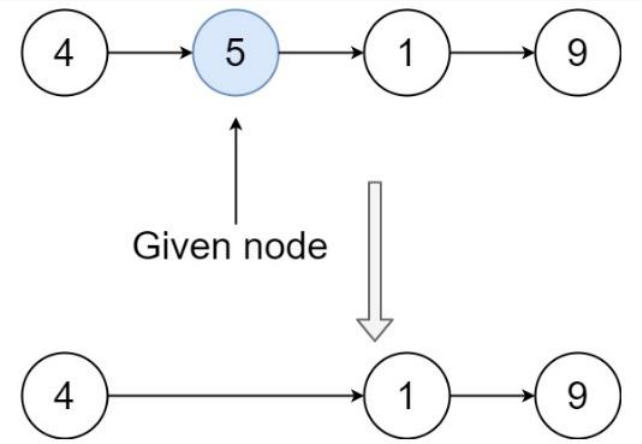
Если в середине два элемента, то функция должна вернуть второй. Например:

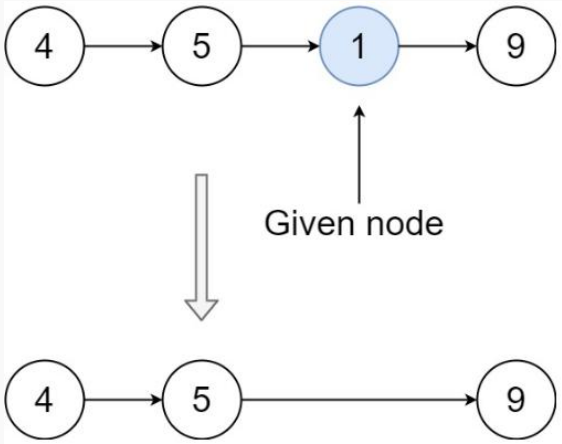


Какая асимптотика у вашего алгоритма?

ЗАДАЧА 4

Вам известен указатель на некоторый элемент списка A. Напишите функцию, которая удалит этот элемент из списка. Например:

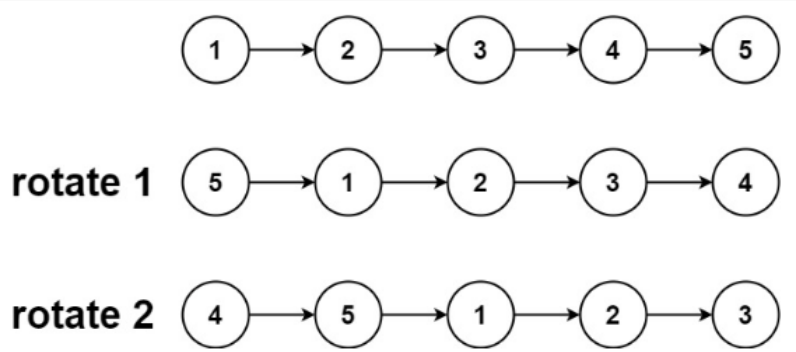




Какая асимптотика у вашего алгоритма?

ЗАДАЧА 5

Вам известно начало списка A и некоторое целое число k. Напишите функцию, которая сдвигает циклически элементы списка на k позиций вправо, если $k > 0$ и влево, если $k < 0$. Например, для $k = 2$:



Какая асимптотика у вашего алгоритма?

ДОМАШНЕЕ ЗАДАНИЕ

Доделать задачи 1 – 5.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Бьерн Страуструп, Язык программирования C++/ ред. А. Боборыкин. – 4-е изд. - Москва: Издательство БИНОМ, 2023. – 1213 с.
2. Дональд Кнут Искусство программирования, том 1. Основные алгоритмы = The Art of Computer Programming, vol.1. Fundamental Algorithms. — 3-е изд. — М .: «Вильямс» , 2006. – 720 с.
3. Дональд Кнут Искусство программирования, том 2. Получисленные алгоритмы = The Art of Computer Programming, vol.2. Seminumerical Algorithms. — 3-е изд. — М .: «Вильямс» , 2007. — 832 с.
4. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. М.: МЦМНО, 1999. – 960 с.
5. Скотт Мейерс, Эффективное использование C++. 55 верных способов улучшить структуру и код ваших программ / ред. Д.А. Мовчан – 3-е изд. – Москва: ДМК Пресс, 2017. – 300 с.
6. Robert Sedgewick Algorithms, 4th edition, 2011