Свързани списъци, 27.10

# Термини

## Амортизирана сложност

* Колкото повече нараства масива, толкова повече константно добавяне имаме при добавяне в ArrayList.
* Броя на константите операции ще стават все повече от колкото линейните операции.

## Randomized data structure

# Теоретични въпроси

## Въпрос 1

* Възможно ли е паметта, масива просто да му разшири блокчета, вместо да заделяме нов по-голям масив.
* Не.
* Може друг процес да заема паметта, до която се намира масива ни.

## Въпрос 2

* Възможно ли е да правим Binary Search на свързан списък.
* Нито да, нито не.
* Технически можем, но се губи смисъла.

## Въпрос 3

* Според вас кое е по-ефикасно, да направим Shuffle в LinkedList или в Масив(Array)
* Linked List, защото сменяме просто едни сочещи референции, а не местим големи стойности в паметта.

## Въпрос 4

* Възможно ли е да имаме Double Linked List с памет като на Singly Linked List
* Да – XOR Linked List
* Не се ползва в практираката, защото компилатора не може да оптимизира операциите свързани с XOR на адреси.
* Объркваш компилатора.

# Масив

* Константен достъп до елемент на индекс, дължи се защото елементите са един след друг в паметта.
* При linked листа нямаме прахосване на памет, за разлика от динамичен масив.

## Добавяне/изтриване в началото

* O(n)

## Добавяне/изтриване в края

* О(1)

## Добавяне/изтриване в средата

* O(n)

# Свързан списък

* Счита се за линейна структура от данни.
* Може лесно да се направи на нелинейна структура от данни.

## Singly linked list

## Doubly linked list

## Linked list with cycles

## XOR linked list

## Unrolled linked list

* ¾ capacity factor
* Всеки node e масив, ¾ пълен.
* Комбинира предимствата на свързан списък и масив.
* Определяме големината на масива спрямо cache-a.
* Cache lime. Ако cache-a има място за 10 елементи, трябва в масива да имаме място за 10\*x елементи, x > 0, x принадлежи на N.

## Skip list

* Ако се качим на метрото, трябва да спираме на всяка спирка.
* Ще пуснем 3 влака.
* 1вия ще върви нормално
* 2рия ще прескача половината спирки
* 3тия ще прескача другата половина спирки
* На база на входните ни данни да стигнем до мястото, на което да търсим/добавим/трием елемента.