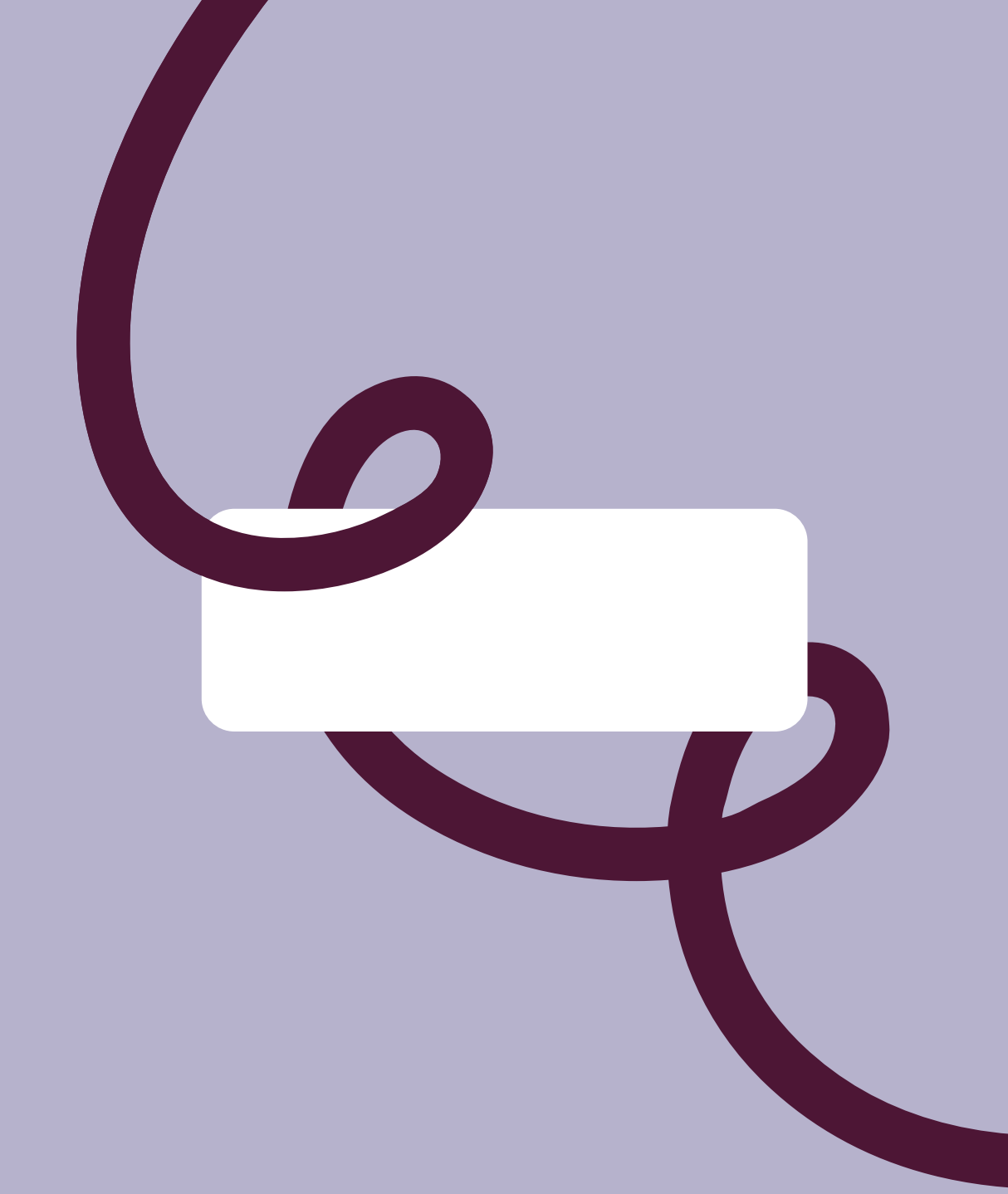


Хеш Таблици

Set - ADT колекция от елементи без повторение
↳ insert(el)
↳ contains(el)
↳ remove(el)

Map/Dictionary/Associate Array - ADT
↳ колекция от key-value pairs (KVP колекция)
↳ "бързо" търсене по ключ
↳ insert(key, value)
↳ contains(key) : bool
↳ get(key) : valueType
↳ remove(key)

- Възможни имплементации
  1) Самобалансиращи се дървета (AVL и RB)
  std::set, std::map
  ordered structure: пазят наредба на елементите.
  ⇝ При итерация редът е строго въведената от comparator-а наредба => обхождаме елементите в сортиран ред
  Benefit: имаме наредба
  Drawback: операциите са $\Theta(\log(n))$

2) **Хеш таблица**: структура от данни за бързо търсене (в основата си тя е масив)
Как да определя на кой индекс да поставя KVP?

- Какво е хеш (hash)? ⇝ удобен за нас "псевдоним" на някакъв сложен обект. Най-често хешът е число, но в някои случаи може

да бъдат 2 или повече числа, или последователност от битове.
По-общо хеширането е представяне на данни с произволен размер чрез данни с фиксиран размер.
Основното изискване към хеша е да е лесно сравним и компактен.

- Какво е хеширане? - Хешированието е действието, при което алгоритмично даден обект с хеш стойност. Тъй като самото хеширане много зависи от обектите, върху които го прилагаме, а също така и какви хеш стойности ще използваме, това е по-скоро техника, от колкото алгоритъм.

- Какво е хешираща ф-я? : Ф-я, която приема аргумент обект и връща неговия хеш.

- Какво е колизия? На два или повече различни обекта да съпоставим един и същ хеш.

В случая с хеш таблицата хеш ф-ята ще приема ключ и ще връща на кой индекс да се запише стойността.

```
insert("test", 42) { arr[hash("test")] = 42; }
```

- Какви са изискванията към една hash ф-я?

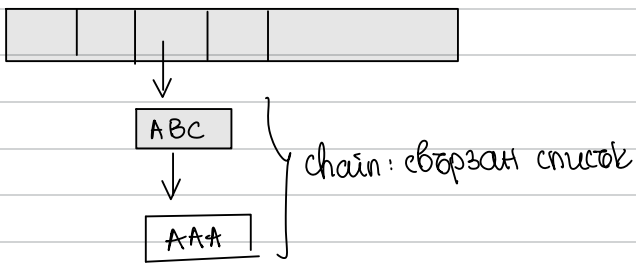
① Детерминистичност (еднакъв вход = еднакъв изход)
за $x = y \Rightarrow hash(x) = hash(y)$
② Бързо смятане на хеш
③ Равномерност: Хешовете трябва да са равномерно разпр.

Хешированието е необратимо преобразуване.

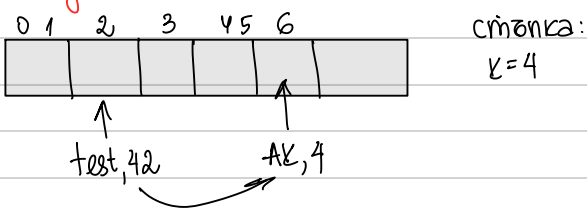
Начини за справяне с колизии.

① SeparateChaining

Нека hash("ABC") = hash("AAA")



② Linear probing

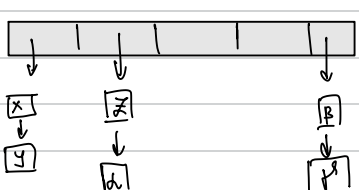


При колизия пробваме следващата клетка със стъпка k
gcd(N,k)=1 (за да пробваме вс. кл.)

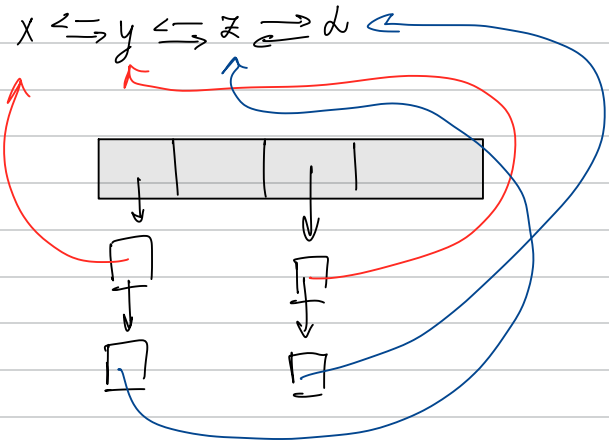
= ще правим Map<String, int> чрез SeparateChaining

– при обхождане ще ги отнегатваме спрямо реда на добавяне

Знаем, че x е преди y / z е преди α / β е преди γ



Как разбираме дали x е преди α или след α.


x
y
z
a