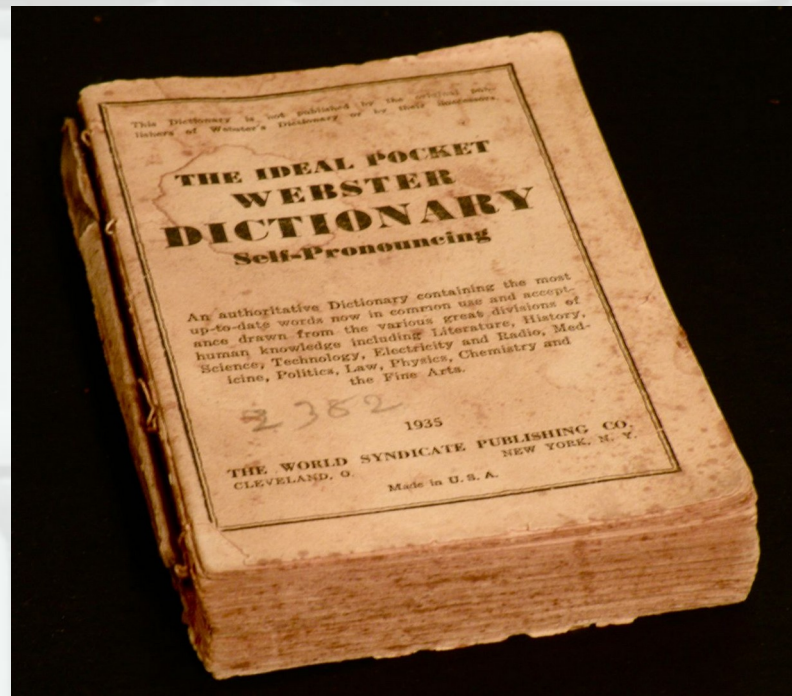


# Речници



# Логическо описание

- Структура с два типа елементи: ключове (с линейна наредба) и стойности
- Операции
  - създаване на празен речник
  - търсене на стойност по ключ
  - добавяне на стойност с даден ключ
  - изтриване на даден ключ
  - обхождане на всички ключове/стойности

# Реализация: списък

- Списък от двойки
- Търсене, изтриване:  $O(n)$ , добавяне:  $O(1)$
- Обхождането е в реда на включването
- Вариант: списък, сортиран по ключове
  - Предимство: спираме търсенето по-рано
  - Сложността не се подобрява!
  - Обхождането е в нарастващ ред

# Реализация: сортиран масив

- Динамичен масив от двойки, сортиран по ключове
- При нужда от повече памет се заделя нов масив и се копира старият
- Алгоритъм за двоично търсене:
  - сравняваме търсения ключ  $Y$  с този в средата на масива  $X$
  - при  $Y < X$  търсим в лявата половина
  - при  $Y > X$  търсим в дясната половина
  - Сложност:  $O(\log n)$
- Включване и изтриване:  $O(n)$

## Реализация: двоично наредено дърво

- Обикновено ДНД
  - търсене, включване, изключване:  $O(n)$
  - обхождане ЛКД — възходящо по ключове
  - средна сложност:  $O(\log n)$
- Балансирано дърво (AVL, червено-черно)
  - търсене, включване, изключване:  $O(\log n)$
- В дърво
  - също  $O(\log n)$ , но предимства при работа с твърд диск

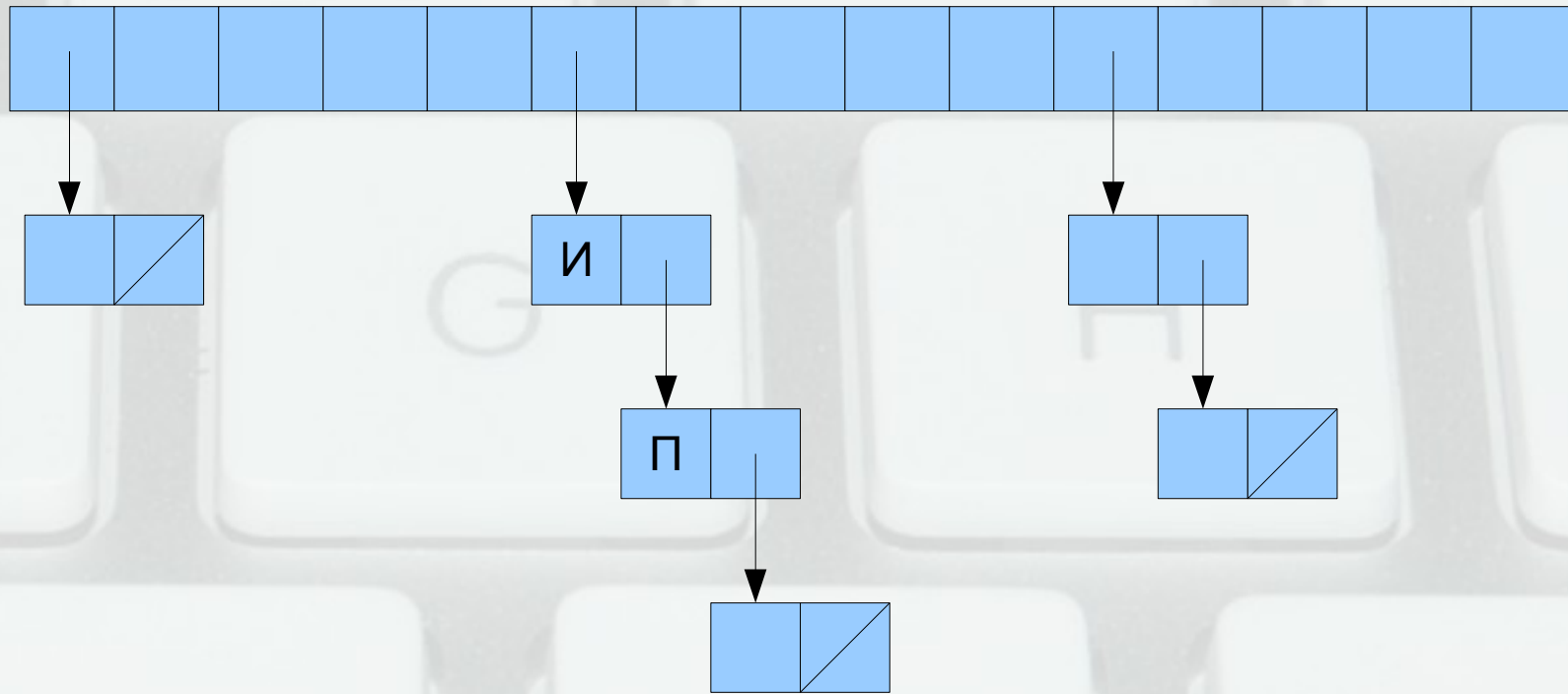
# Хеш таблици

- Масив от  $n$  двойки ключ/стойност
- Хеш функция  $h : \text{Key} \rightarrow [0; n)$ 
  - По ключ пресмята позиция в масива
  - Колизия:  $h(k_1) = h(k_2)$  при  $k_1 \neq k_2$
  - Идея: функция, която дава възможно най-добро “разбъркване” (hash)
  - Намаляване на вероятността от колизия
  - Колизиите не могат да бъдат предодвратени изцяло!

# Разрешаване на колизии

- Разрешаване чрез пряко свързване
- Разрешаване чрез отворено адресиране

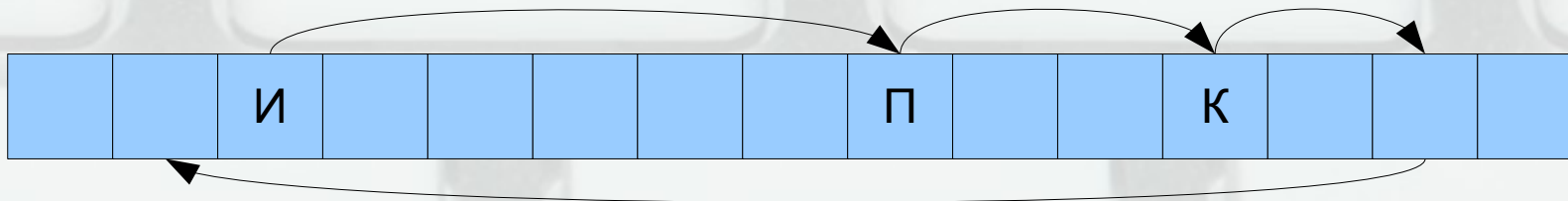
# Пряко свързване





# Отворено адресиране

- Използва се вторична хеш функция
- $g : [0; n) \rightarrow [0; n)$
- за пресмятане на следващата възможна позиция в масива



# Сложност на хеш таблиците

- Търсене, включване, изтриване
  - най-лоша сложност:  $O(n)$
  - средна сложност:  $O(1)$  !!
- Причина: амортизирана сложност

# Разширение на хеш таблици

- Със запълването на хеш таблицата, всички операции се забавят
- Стратегии за разширение:
  - прехеширане
  - инкрементално разширение

# Криптографски хеш функции

- еднопосочни функции
  - $h$  е лесна за смятане
  - $h^{-1}$  е практически трудна за пресмятане
- ефект на лавината
  - ако  $a \approx b$ , то  $h(a) \neq h(b)$
- намирането на колизии е практически невъзможно