

Хеширане и хеш таблици

ИТ Кариера



Учителски екип

Обучение за ИТ кариера

<https://it-kariera.mon.bg/e-learning>

Съдържание

- Хеширащи функции
- Хеш таблици
- Управление на колизии в хеш таблици
- Упражнения: хеш таблици



Хеширащи функции

- Хеширащите функции конвертират ключ от произволен тип до стойност от целочислен тип

Иван

Хеш функция

398

Пешо

Хеш функция

511

```
class Person
{
    string firstName;
    string lastName;
    int age;
}
```


Иван
Петров
25

Хеш функция

25950

Хеширащи функции

```
class Person
{
    string firstName;
    string lastName;
    int age;

    public override int GetHashCode()
    {
        
    }
}
```

Хеширащи функции

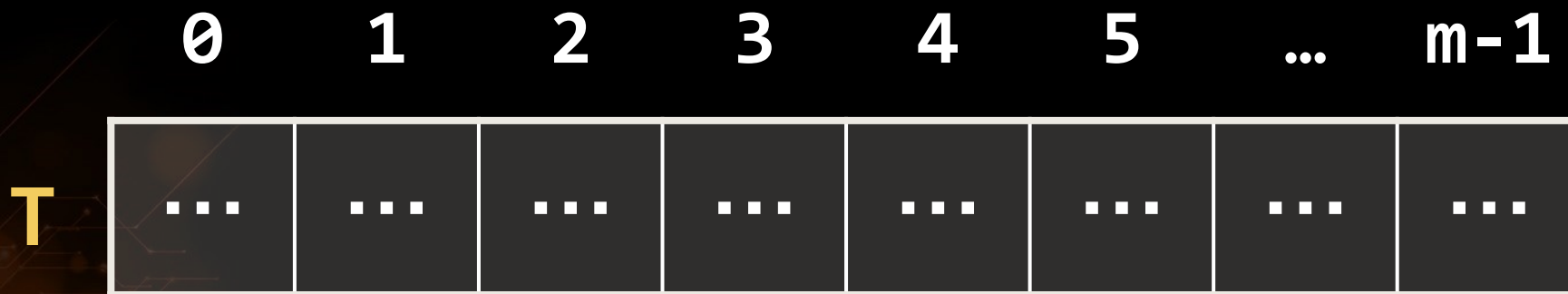
```
class Person
{
    string firstName;
    string lastName;
    int age;

    public override int GetHashCode()
    {
        int firstNameHash = firstName.GetHashCode() * age;
        int lastNameHash = lastName.GetHashCode() * age;

        return firstNameHash + lastNameHash;
    }
}
```


Хеш таблица

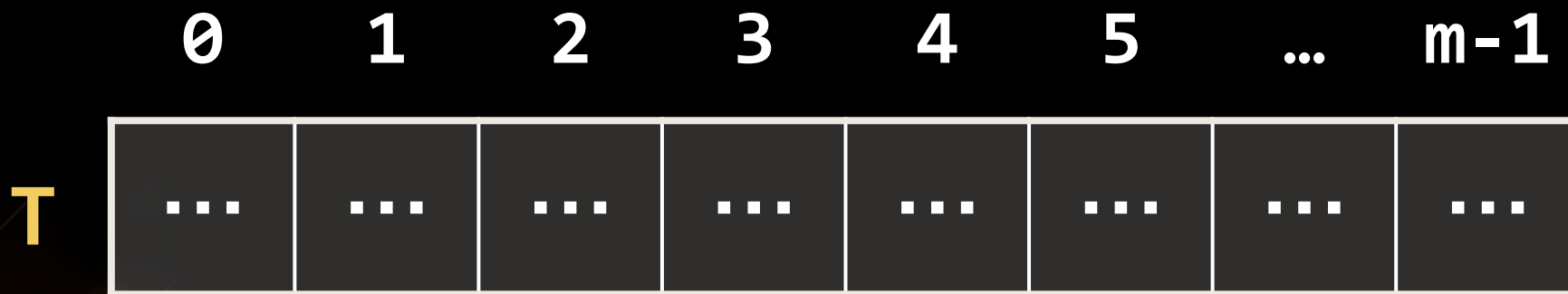
- Хеш таблица е стандартен масив, който съдържа набор от наредени двойки {ключ, стойност}
- Техниката, с която се определя кой ключ на коя позиция в масива да се съхрани се нарича хеширане



Хеш таблица с
размер **m**

Хеш функции и хеширане

- Хеш таблицата (масива) има m позиции, индексирани от 0 до $m-1$
- Хеш функцията конвертира **ключовете** до **индекси** в масив



$k.$ GetHashCode()

Връща 32 битово
цяло число

Хеширащи функции

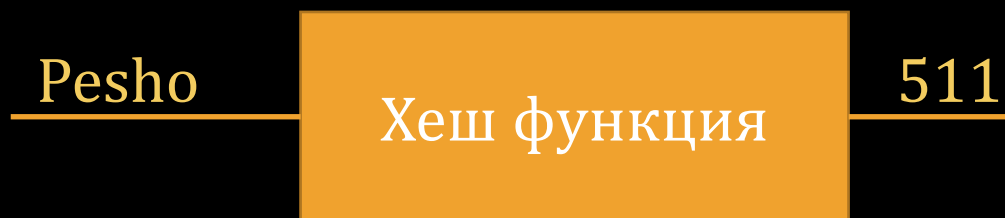
- Перфектно хеширане
 - Перфектно хешираща функция е тази $f(k)$, която прави 1:1 съответствие за всяко уникално k към уникално число в интервала $[0, m-1]$
 - Перфектно хеширащата функция свързва всеки ключ към **уникално** цяло число в рамките на конкретен интервал
- В повечето случаи перфектното хеширане е невъзможно

Хеширащи функции

- Свойства на добрата хешираща функция
 - **Консистентност** - еднакви ключове трябва да произвеждат един и същ хеш
 - **Ефективност** - ефективни при изчисляването на хеш
 - **Равномерност** - хешовете, произведени от хеширащата функция трябва да се равномерно разпределени

Модулна аритметика и хеш таблици

- Имаме масив с размер 16
- Въвеждаме "Pesho"



511 е извън
размера на хеш
таблицата

	0
	1
	2
	3
	4
	5
	6
	7
	...
	15

- Използваме остатъка от делението за да извлечем валидна позиция:

$\text{GetHashCode()} / \text{Array.Length}$

$$511 \% 16 = 15$$

Работа с хеш таблица

stamat

Hash Function % 10

	0
	1
	2
	3
	4
	5
	6
	7
	8
	9

Работа с хеш таблица

mitko

Hash Function % 10

stamat

0

1

2

3

4

5

6

7

8

9

Работа с хеш таблица

ivan

Hash Function % 10

stamat

0

1

2

3

4

5

6

mitko

7

8

9

Работа с хеш таблица

gosho

Hash Function % 10

stamat

0

1

2

3

4

ivan

5

6

mitko

7

8

9

Работа с хеш таблица

maria

Hash Function % 10

stamat

0

1

2

3

4

ivan

5

6

mitko

7

8

gosho

9

Работа с хеш таблица

Колизия

Hash Function % 10

stamat	0
	1
	2
	3
	4
	5
	6
mitko	7
	8
gosho	9

Колизии в хеш таблици

- Колизия настъпва, когато хеш функцията генерира един и същ хеш за различни ключове

$$h(k_1) = h(k_2) \text{ for } k_1 \neq k_2$$

- При нисък брой колизии, бързодействието на хеш таблиците не се афектира

Колизии в хеш таблици

- Стратегии за разрешаване на колизии
 - **Свързване** на елементите в колизия
 - Използване на **други клетки** от таблицата
 - **Cuckoo** хеширане
 - други...

Колизии – свързване на елементи

Хеш функция

maria

0

1

2

3

4

5

6

7

--	--	--	--	--	--	--	--

Колизии – свързване на елементи

Хеш функция

ivan

0

1

2

3

4

5

6

7

			maria				
--	--	--	-------	--	--	--	--

Колизии – свързване на елементи

Хеш функция

stamat

0

1

2

3

4

5

6

7

			maria		ivan		
--	--	--	-------	--	------	--	--

Колизии – свързване на елементи

Хеш функция

pesho

0

1

2

3

4

5

6

7

stamat

maria

ivan

Колизии – свързване на елементи

Хеш функция

mitko

0

1

2

3

4

5

6

7

stamat

maria

ivan



pesho

Елементите се свързват
в свързан списък

Колизии – свързване на елементи

Хеш функция

joro

0

1

2

3

4

5

6

7

stamat

maria

ivan

mitko

pesho

Колизии – свързване на елементи

Хеш функция

rosi

0

1

2

3

4

5

6

7

stamat

maria

ivan

mitko

pesho

joro

Колизии – свързване на елементи

Хеш функция

alex

0

1

2

3

4

5

6

7

rosi

stamat

maria

ivan

mitko

pesho

joro

Колизии – свързване на елементи

Хеш функция

alex

0

1

2

3

4

5

6

7

rosi

stamat

maria

ivan

mitko

pesho

joro

alex

Колизии – отворена адресация

- **Отворена адресация** е стратегия за разрешаване на колизии, при която конфликтните елементи се съхраняват в друга клетка на хеш таблицата
- **Линейно пробване** - взима се следващия празен слот след позицията на колизията

$$h(\text{key}, i) = h(\text{key}) + i$$

където i е поредния брой на опита: 0, 1, 2, ...

$h(\text{key}) + 1, h(\text{key}) + 2, h(\text{key}) + 3$, и т.н.

Колизии – отворена адресация

- Квадратично пробване - i -тата следваща позиция се определя от квадратна функция (c_1 и c_2 са константи и от тях зависи кои позиции ще бъдат пробвани)

$$h(\text{key}, i) = h(\text{key}) + c_1 * i + c_2 * i^2$$

$$h(\text{key}) + 1^2, h(\text{key}) + 2^2, h(\text{key}) + 3^2, \text{ etc.}$$

- Двойно хеширане - използване на втора хеш функция за колизиите

$$h(\text{key}, i) = h_1(\text{key}) + i * h_2(\text{key})$$

Колизии – линейно пробване

Хеш функция

maria

0

1

2

3

4

5

6

7

--	--	--	--	--	--	--	--

Колизии – линейно пробване

Хеш функция

ivan

0

1

2

3

4

5

6

7

			maria				
--	--	--	-------	--	--	--	--

Колизии – линейно пробване

Хеш функция

stanat

0

1

2

3

4

5

6

7

			maria		ivan		
--	--	--	-------	--	------	--	--

Колизии – линейно пробване

Хеш функция

pesho

0

1

2

3

4

5

6

7

stanat

maria

ivan

Колизии – линейно пробване

Хеш функция

pesho

0

1

2

3

4

5

6

7

stanat

maria

ivan

Колизии – линейно пробване

Хеш функция

pesho							
0	1	2	3	4	5	6	7
		stanat	maria		ivan		

Колизии – линейно пробване

Хеш функция

mitko

0

1

2

3

4

5

6

7

stanat

maria

pesho

ivan

Колизии – линейно пробване

Хеш функция

joro

0	1	2	3	4	5	6	7
		stanat	maria	pesho	ivan		mitko

Колизии – линейно пробване

Хеш функция

						joro		
0	1	2	3	4	5	6	7	
		stanat	maria	pesho	ivan		mitko	

Колизии – линейно пробване

Хеш функция

							joro	
0	1	2	3	4	5	6	7	
		stanat	maria	pesho	ivan		mitko	

Колизии – линейно пробване

Хеш функция

rosi

0

1

2

3

4

5

6

7

stanat

maria

pesho

ivan

joro

mitko

Колизии – линейно пробване

Хеш функция

alex

0	1	2	3	4	5	6	7
rosi		stanat	maria	pesho	ivan	joro	mitko

Колизии – линейно пробване

Хеш функция

0	1	2	3	4	alex	6	7
rosi		stanat	maria	pesho	ivan	joro	mitko

Колизии – линейно пробване

Хеш функция

0	1	2	3	4	5	alex	7
rosi		stanat	maria	pesho	ivan	joro	mitko

Колизии – линейно пробване

Хеш функция

							alex
0	1	2	3	4	5	6	7
rosi		stanat	maria	pesho	ivan	joro	mitko

Колизии – линейно пробване

Хеш функция

alex

0

1

2

3

4

5

6

7

rosi

stanat

maria

pesho

ivan

joro

mitko

Колизии – линейно пробване

Хеш функция

alex

0

1

2

3

4

5

6

7

rosi

stanat

maria

pesho

ivan

joro

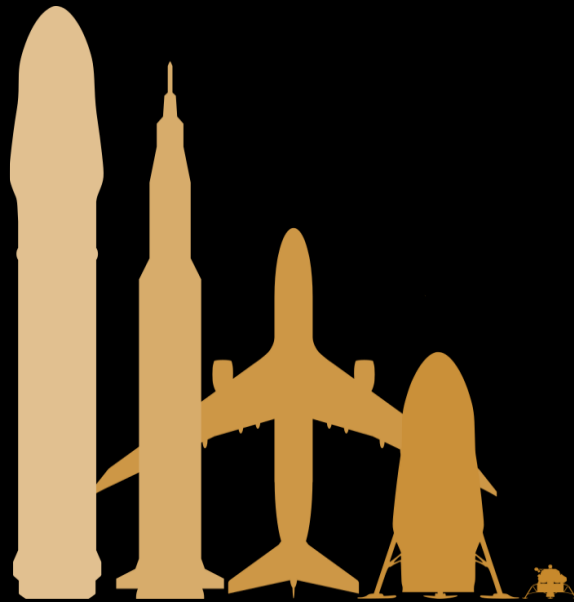
mitko

Колизии – линейно пробване

Хеш функция

0	1	2	3	4	5	6	7
rosi	alex	stanat	maria	pesho	ivan	joro	mitko

Упражнение: сравняване на ключове



Сравняване на ключове
при работа със собствени класове

Сравняване на ключове

`Dictionary<TKey,TValue>` използва:

- `Object.Equals()` – за сравнение на ключове
- `Object.GetHashCode()` – за изчисляване на ключове

`SortedDictionary<TKey,TValue>` използва

- `IComparable<T>` за подредба на ключове

Реализация на Equals() и GetHashCode()

```
public class Point
{
    public int X { get; set; }
    public int Y { get; set; }

    public override bool Equals(Object obj)
    {
        if (!(obj is Point) || (obj == null)) return false;
        Point p = (Point)obj;
        return (X == p.X) && (Y == p.Y);
    }

    public override int GetHashCode()
    {
        return (X << 16 | X >> 16) ^ Y;
    }
}
```

Реализация на IComparable<T>

```
public class Point : IComparable<Point>
{
    public int X { get; set; }
    public int Y { get; set; }

    public int CompareTo(Point otherPoint)
    {
        if (X != otherPoint.X)
        {
            return this.X.CompareTo(otherPoint.X);
        }
        else
        {
            return this.Y.CompareTo(otherPoint.Y);
        }
    }
}
```

Речници

key	value
John Smith	+1-555-8976
Sam Doe	+1-555-5030
Sam Smith	+1-555-4542
John Doe	+1-555-3527

Речници

Дефиниция и функционалност

Речник: Dictionary (MAP)

Абстрактния тип данни “речник” асоциира стойности с уникални ключове

- Тази структура е позната като **карта** или **асоциативен масив**
- Съдържа набор от наредени двойки от тип **{key, value}**

Имплементации

- Хеш таблици, балансирани дървета, списъци, масиви и др.

Dictionary<TKey, TValue>

Основна функционалност:

- **Add(key, value)** – добавя елемент
- **Remove(key)** – премахва елемент
- **this[key] = value** – добавя или подменя елемент
- **this[key]** – извлича елемент
- **Keys** – връща всички ключове (по ред на добавяне)
- **Values** – връща всички стойности (по ред на добавяне)

Dictionary<TKey, TValue>

Основна функционалност:

- **ContainsKey(key)** – проверява дали ключа е в речника
- **ContainsValue(value)** – проверява дали стойността е в речника
- **TryGetValue(key, out value)**
 - Ако намери стойността я записва във параметъра **value** и връща **true**
 - Иначе връща **false**

Упражнение: реализация на хеш таблица



Реализация на хеш таблица
стратегия за колизии - свързване на елементи

Обобщение

- Хеширащи функции
- Хеш таблици
- Управление на колизии в хеш таблици
- Упражнения: хеш таблици



Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"



Министерство
на образованието
и науката



Национална
програма
„Обучение за
ИТ кариера“

- Курсът се разпространява под свободен лиценз **CC-BY-NC-SA**

