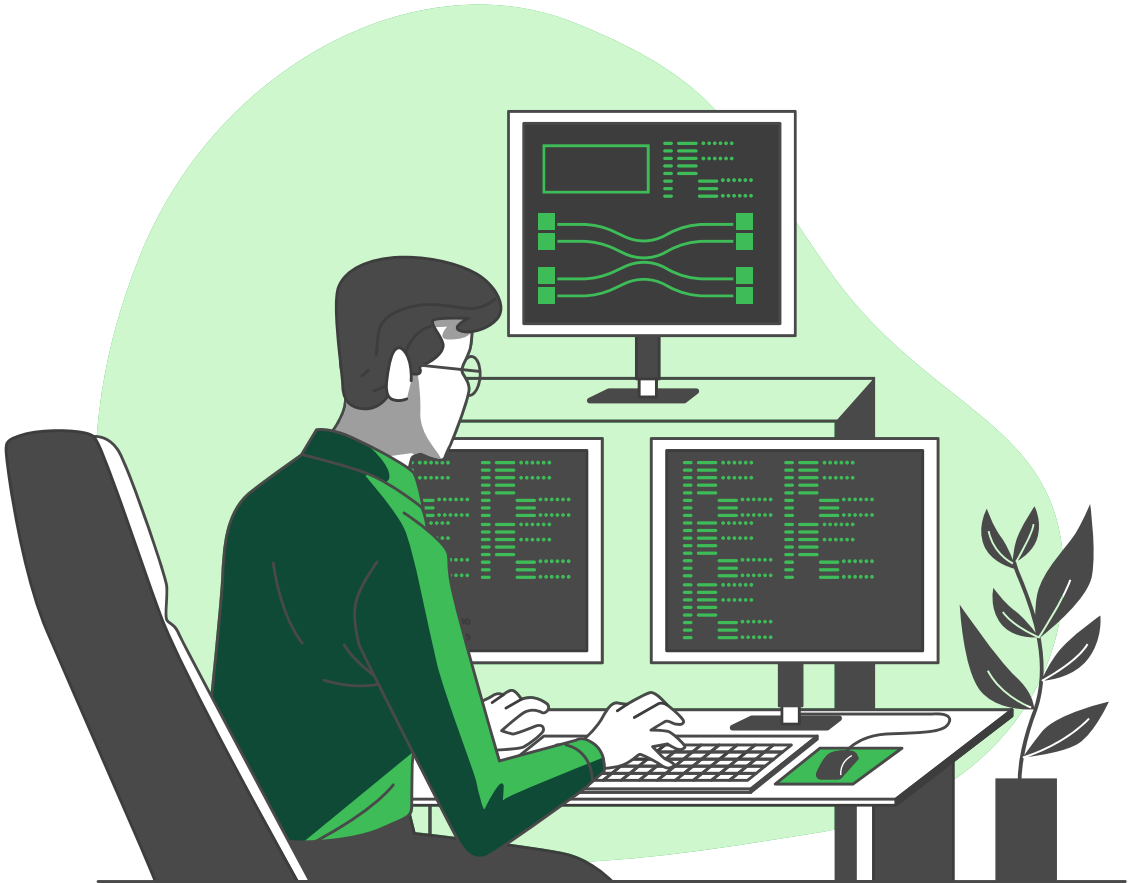


Hashing

CS 251 - Data Structures and Algorithms

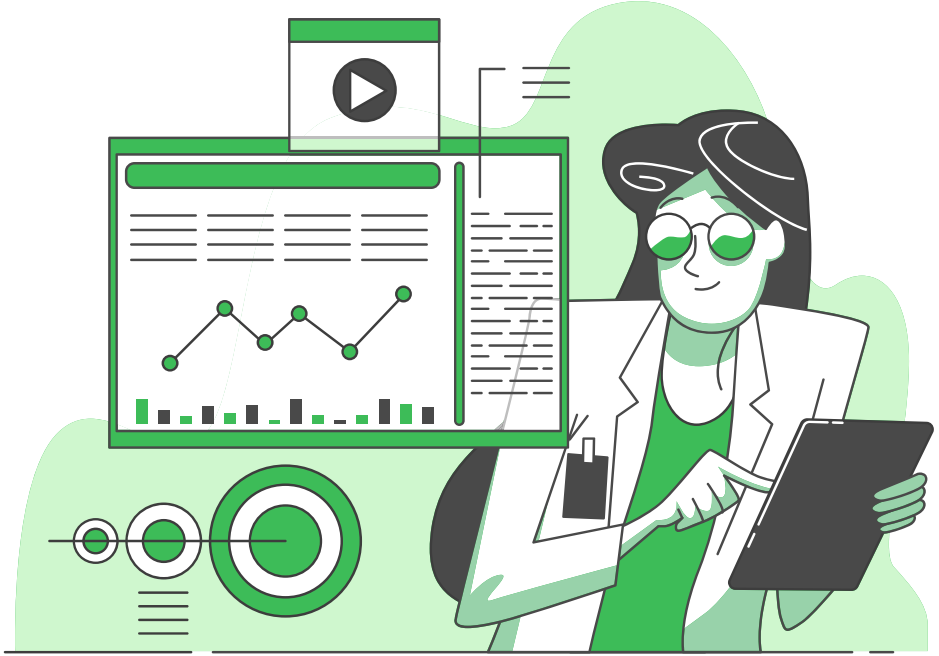
Note:
Slides complement the discussion in class

Table of Contents

01

Hashing

Map any key to integers

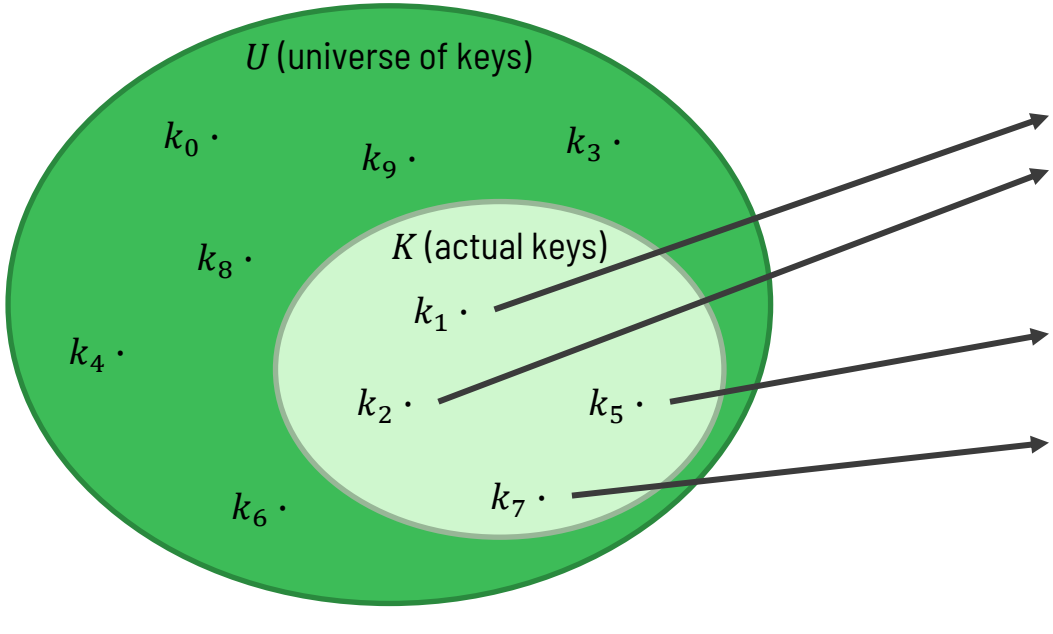


01

Hashing

Map any key to integers

Direct-Address Table



	T
0	-
1	$x_1 = (k_1, v_1)$
2	$x_2 = (k_2, v_2)$
3	-
4	-
5	$x_5 = (k_5, v_5)$
6	-
7	$x_7 = (k_7, v_7)$
8	-
9	-

$m = |U|$

```
Search(T, k):
    return T[h(k)]

Insert(T, x):
    T[h(x.k)] = x

Delete(T, x):
    T[h(x.k)] = null
```

Each data point x has a key $x.k$ and its respective item $x.v$

“to chop into small pieces; make into hash; mince.”

—Hash (verb) Definition

Image from: https://www.cutco.com/learn/how-to-mince-fresh-herbs/

Mapping Any Key to an Integer Key

Working with **integer** keys? Make them non-negative (in case they aren't).

Working with **floating point** keys? Use XOR of the two halves of the binary representation (aka. folding).

Working with **Strings**? Use both integer representation of each character and their respective locations (e.g., rolling hashing).

Working with **compound keys**? Mix them based on primitive data types.

Example: String hashCode()

hashCode

```
public int hashCode()
```

Returns a hash code for this string. The hash code for a `String` object is computed as

```
s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1]
```

using `int` arithmetic, where `s[i]` is the *i*th character of the string, `n` is the length of the string, and `^` indicates exponentiation. (The hash value of the empty string is zero.)

Overrides:

`hashCode` in class `Object`

Returns:

a hash code value for this object.

See Also:

`Object.equals(java.lang.Object)`, `System.identityHashCode(java.lang.Object)`

https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#hashCode()

Example: String hashCode()

```java
public class Hashing
{
   public static void main(String[] args)
   {
      String[] words = {"LIVE", "EVIL", "VILE", "LEVI", "VEIL"};

      for (String w : words)
      {
         System.out.println("H(" + w + ") = " + w.hashCode());
      }
   }
}
```

H(“LIVE”) = 2337004
H(“EVIL”) = 2140564
H(“VILE”) = 2634604
H(“LEVI”) = 2333164
H(“VEIL”) = 2630674

Polynomial Rolling Hash

$$H(S,a) = s_0 a^{m-1} + s_1 a^{m-2} + \cdots + c_{s-1} a^0 = \sum_{i=0}^{m-1} s_i a^{m-i-1}$$

where a is a constant, $S = s_0 s_1 \ldots s_{m-1}$ is a string of length m.

Consider the string $S = \text{"abcd"}$, $a = 26$, and consecutive substrings of length 3:

$$H(\text{"abc"}, 26) = 1 \times 26^2 + 2 \times 26^1 + 3 \times 26^0$$
$$H(\text{"bcd"}, 26) = 2 \times 26^2 + 3 \times 26^1 + 4 \times 26^0$$

$$H(\text{"bcd"}, 26) = (H(\text{"abc"}, 26) - H(\text{"a"}, 26) \times 26^2) \times 26 + H(\text{"d"}, 26)$$
$$= \left((1 \times 26^2 + 2 \times 26^1 + 3 \times 26^0) - (1 \times 26^0) \times 26^2\right) \times 26 + (4 \times 26^0)$$
$$= 2 \times 26^2 + 3 \times 26^1 + 4 \times 26^0$$

Rolling Hash Algorithm

```
algorithm H(S:string, a:ℤ)
   let n be the length of S
   sum ← 0
   p ← n – 1
   for i from 0 to n - 1 do
      sum ← sum + intval(S[i]) * a^p
      p ← p - 1
   end for
   return sum
end algorithm
```

```
algorithm rollinghash(S:string, m:ℤ⁺, a:ℤ)
   hashvalue ← H(S[0, m - 1], a)
   let n be the length of S
   for i from 1 to n - m do
      hashvalue = ((hashvalue – intval(S[i-1])*a^(m-1)) * a) – intval(S[i+m-1])
   end for
end algorithm
```

Note: S[i] corresponds to the numerical value of the character at index i.

$h(\text{last slide}) = \text{End}$

Do you have any questions?