Symbol Tables

Gianpiero Cabodi and Paolo Camurati
Dip. Automatica e Informatica
Politecnico di Torino

Symbol Tables

- Definition
  - A Symbol Table is a data structure with records including a key and allowing operations such as
    - Insertion of a new record
    - Search of a record with a given key
    - Delete, select, order, union
- Sometimes symbol tables are denoted with the term dictionary

Many applications need to insert and search

Applications

Applications	Target, i.e., searching	Key	Return Value
Dictionary	Definition	Word	Definition
Book index	Relevant pages	Word	Page list
DNS	IP address given its URL	URL	IP address
Invers DNS	URL given its IP address	IP address	URL
File system	File on disk	File name	Disk location
Web search	Web page	Keyword	Page list

Implementations

- Linear structures
  - Direct Access Tables
  - Arrays
    - Unordered
    - Ordered
  - Lists
    - Unordered
    - Ordered
- Tree structures
  - Binary Search Trees . (BSTs)
  - Balanced Trees
    - 2-3-4
    - RB-tree
    - B-tree
- Hash Tables

Aready studied
To be done
Not analysed

Complexity: Worst Case

Data Structure	Insert	Search
Direct Access Table	1	1
Unordered Array	1	n
Ordered Array Linear Search	n	n
Ordered Array Binary Search	n	logn
Unordered List	1	n
Ordered List	n	n
BST	n	n
RB-tree	logn	logn
Hashing	1	n

Complexity: Average Case

Data Structure	Insertion	Search	
		Hit	Miss
Direct Access Table	1	1	1
Unordered Array	1	n/2	n
Ordered Array Linear Search	n/2	n/2	n/2
Orderer Array Binary Search	n/2	logn	logn
Unordered List	1	n/2	n
Ordered List	n/2	n/2	n/2
BST	logn	logn	logn
RB-tree	logn	logn	logn
Hashing	1	1	1

Direct Access Tables

- All search algorithms analysed so far are based on comparisons
- Exception
  - Direct access tables
  - Hash tables
- In direct access tables
  - A key $k \in U = \{0, 1, \ldots, card(U)-1\}$
  - Is used as the index of the array st
    - st[0, 1, …, card(U)-1]

The cardinality of U is small
The keys are not necessary integer values

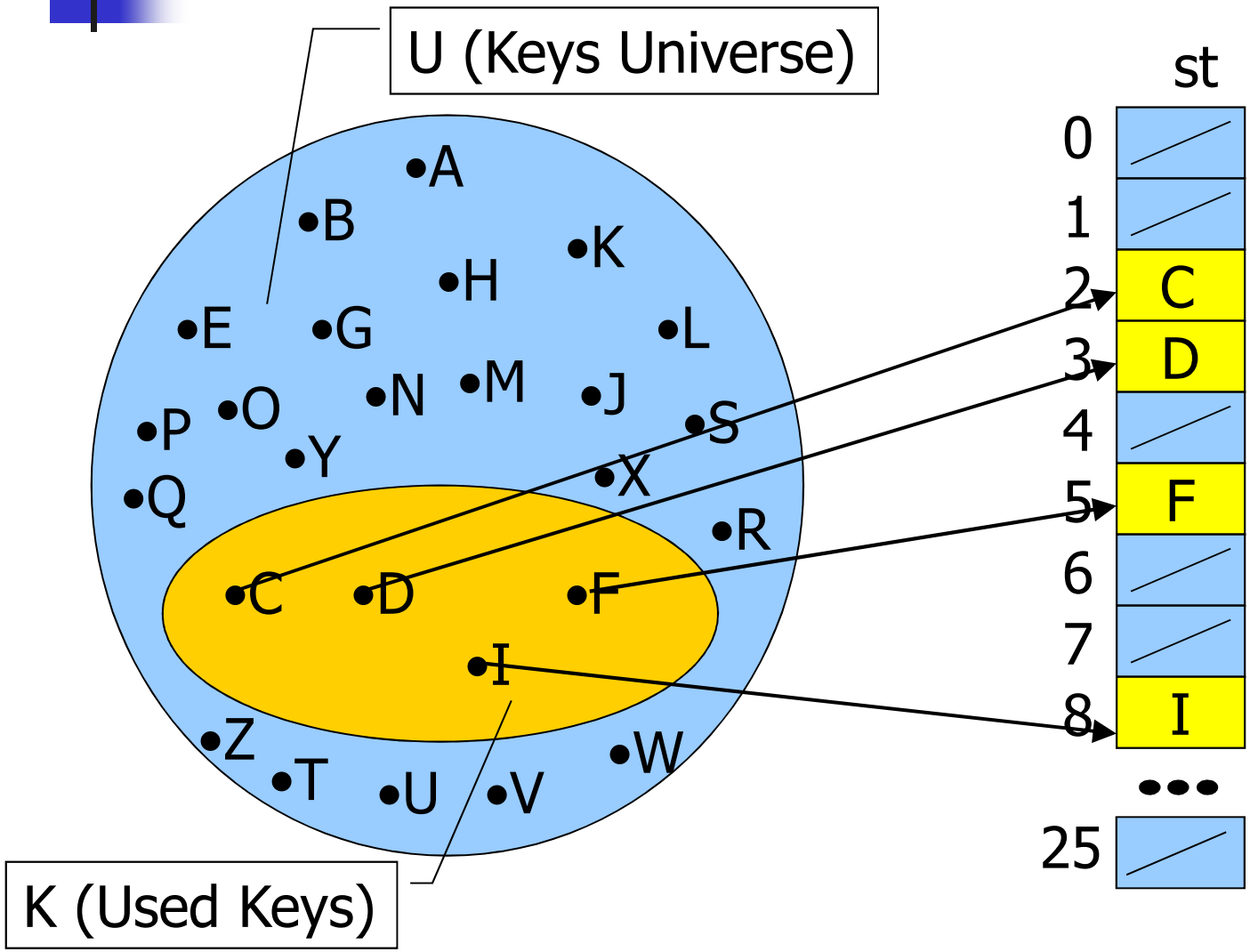
Direct Access Tables

- Given a universe U of keys, i.e., each key $k \in U$
  - maxN is the number of elements in U , i.e., maxN=|U|
- The array st i used to store the keys
- Given a key k
  - Function getindex(k) returns an integer from 0 to maxN-1, acting as an array index
  - If the key k is in the table, its position is st[getindex(k)]
  - If the key k is not in the table st[getindex(k) stores an empty element

Direct Access Tables

- This looks simple enough, but the keys may not be integer values
  - If keys are integers from 0 to maxN-1
    - `getindex(k) = k;`
  - If keys are capital letters in the English alphabet A..Z
    - `getindex(k) = k – ((int) 'A');`
  - For generic keys getindex has to map those keys into integer values in the range [0, maxN-1]

Direct Access Tables



Pros & Cons

- Insert, search, and delete complexity
  - $T(n) = \Theta(1)$
- Init complexity
  - $T(n) = \Theta(\max N)$
- Memory usage
  - $S(n) = \Theta(|U|) = \Theta(\max N)$

Pros & Cons

- Limits of direct access tables
  - For large |U| the array st cannot be allocated, i.e., they can be used only for small maxN
  - If |K| << |U| there is a memory loss
- In those two cases direct access tables have to be extended into hash tables
- Often used to convert keys into integers (and vice-versa) with a cost equal to 1