

MAPPE E DIZIONARI

DEFINIZIONE DI MAPPA

- ADT che contiene dati come **coppie chiave/valore**
- Ogni chiave deve essere **UNICA**
 - viene utilizzata come identificatore
 - utilizzata per effettuare ricerca/rimozione elementi

DEFINIZIONE DI DIZIONARIO

- Simile ad una mappa
- **NON** si richiede che le chiavi siano uniche
 - possono essere associate a **più valori**
- Si distinguono in:
 - **Dizionari ordinati**
 - chiavi ordinate
 - **Dizionari non ordinati**
 - chiavi non ordinate
- La trattazione si soffermerà su dizionari a chiave unica (= mappe)

INTERFACCIA

```
public interface Dictionary extends Container {  
    void insert(Comparable key, Object value);  
    void remove(Comparable key);  
    Object find(Comparable key);  
}
```

- Definiamo una nuova **eccezione**
 - `class DictionaryItemNotFoundException extends RuntimeException`

IMPLEMENTAZIONE

- Struttura dati: array riempito solo in parte
- Il dizionario contiene oggetti di tipo `Pair`
 - coppie chiave/valore

- È possibile scegliere se:
 - mantenere chiavi **ordinate**
 - ◦ performante nella lettura (si può usare binary search)
 - lasciare chiavi **non ordinati**

PERFORMANCE DIZIONARIO

DIZIONARIO CON ARRAY ORDINATO

- **Ricerca**
 - prestazione $O(\log n)$
 - binary search
- **Rimozione**
 - prestazione $O(n)$
 - bisogna effettuare una ricerca e poi spostare mediamente $\frac{n}{2}$ elementi
- **Inserimento**
 - prestazione $O(n)$
 - o con algoritmo diverso $O(n \log n)$
 - si può usare insertion sort in un array ordinato

DIZIONARIO CON ARRAY NON ORDINATO

- **Ricerca**
 - prestazione $O(n)$
 - ricerca lineare
- **Rimozione**
 - prestazione $O(n)$
 - bisogna effettuare ricerca lineare e poi spostare nella posizione trovata l'ultimo elemento

- **Inserimento**

- prestazione $O(n)$
- Bisogna rimuovere (sovrascrivere) un elemento con la stessa chiave, se c'è, e poi inserire il nuovo elemento nella ultima posizione dell'array (l'ordinamento non interessa)

Dizionario	array ordinato	array non ordinato
ricerca	$O(\lg n)$	$O(n)$
inserimento	$O(n)$	$O(n)$
rimozione	$O(n)$	$O(n)$

[$O(1)$ per chiavi non uniche]

LA CLASSE PAIR

- Contiene elementi formati da **coppie chiave/valore**
- Due **campi di esempio**
 - key (Comparable)
 - value (Object)
- Metodi di accesso e modificatori per questi campi di esempio

IMPLEMENTAZIONE DIZIONARIO NON ORDINATO

```

1 public class ArrayDictionary implements Dictionary {
2     protected Pair[] array;
3     protected int arraySize;
4     protected final int INIT_SIZE = 100;
5
6     public ArrayDictionary() {
7         array = new Pair[100];
8         makeEmpty();
9     }
10
11     public boolean isEmpty() {
12         return arraySize == 0;
13     }
14
15     public void makeEmpty() {
16         arraySize = 0;
17     }
18
19     public String toString() {
20         String s = "";
21         for (int i = 0; i < arraySize; i++) {
22             s = s + array[i] + "\n";
23         }
24         return s;
25     }
26
27     public void insert(Comparable key, Object value) {
28         if (key == null)
29             throw new IllegalArgumentException();
30         try {
31             remove(key);
32         } catch (DictionaryItemNotFoundException e) {
33         }
34
35         if (arraySize == array.length)
36             array = resize(2 * arraySize);
37         array[arraySize++] = new Pair(key, value);
38     }
39
40     protected Pair[] resize(int newLength) // solita tecnica
41     {
42         if (newLength < array.length)
43             throw new IllegalArgumentException();
44         Pair[] newArray = new Pair[newLength];
45         System.arraycopy(array, 0, newArray, 0, array.length);
46         return newArray;
47     }
48
49     public void remove(Comparable key) {
50         array[linearSearch(key)] = array[--arraySize];
51     }
52
53     public Object find(Comparable key) {
54         return array[linearSearch(key)].getValue();
55     }
56
57     private int linearSearch(Comparable key) {
58         for (int i = 0; i < arraySize; i++) {
59             if (array[i].getKey().compareTo(key) == 0) {
60                 return i;
61             }
62         }
63         throw new DictionaryItemNotFoundException();
64     }
65
66     protected class Pair // classe interna ad ArrayDictionary
67     {
68         ...
69     }
70
71 }
72

```

IMPLEMENTAZIONE DIZIONARIO ORDINATO

- L'array deve sempre essere mantenuto ordinato con Insertion Sort
 - ogni volta che si inserisce o toglie un elemento
- La ricerca deve avvenire attraverso **binary search** in quanto l'array è ordinato