Data Structures & Algorithms (COMP2113)

Lecture # 23

Basic Data Structures | Part 07

Dictionaries

Instructor

Dr. Aftab Akram (PhD Computer Science, China)

Lecturer, Division of Science & Technology

University of Education, Lahore

Dictionary

- A Dictionary is a collection of records.
- Most familiar notation for dictionary is a Database.
- The dictionary ADT provides operations for storing records, finding records, and removing records from the collection.
- A database record could simply be a number, or it could be quite complicated, such as a payroll record with many fields of varying types.
- We do not want to describe what we are looking for by detailing and matching the entire contents of the record.
- We typically define what record we want in terms of a key value.
- For example, ID number is the search key

Dictionary

- To implement the search function, we require that keys be comparable.
- At a minimum, we must be able to take two keys and reliably determine whether they are equal or not.
- That is enough to enable a sequential search through a database of records and find one that matches a given key.
- However, we typically would like for the keys to define a total order, which means that we can tell which of two keys is greater than the other.
- Using key types with total orderings gives the database implementor the opportunity to organize a collection of records in a way that makes searching more efficient.

Dictionary

- An example is storing the records in sorted order in an array, which permits a binary search.
- In practice most fields of most records consist of simple data types with natural total orders.
- For example, integers, floats, doubles, and character strings all are totally ordered.

Dictionary - ADT

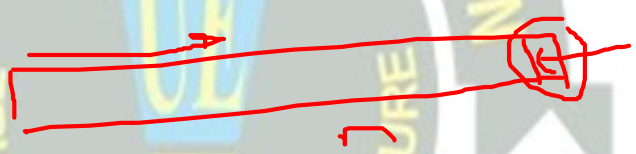
- **`clear`**- Reinitialize dictionary
- **`insert`**–Insert a record
- **`Remove`**–Remove and return a record.
- **`removeAny`**– Remove and return an arbitrary record from dictionary.
- **`find`**– Return a record matching "k“.
- **`size`**– Return the number of records in the dictionary.

Key-Value Pair

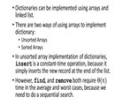
- Key is added as separate field in the dictionary.
- Each entry in the dictionary will contain both a record and its associated key.
- Such entries are known as key-value pair.
- Keys tend to be much smaller than records, so this additional space overhead will not be great.
- The **`insert`** method of the dictionary ADT supports the key-value pair implementation because it takes two parameters, a record and its associated key for that dictionary.

Array-Based implementation of Dictionary

- Dictionaries can be implemented using arrays and linked list.
- There are two ways of using arrays to implement dictionary:
  - Unsorted Arrays
  - Sorted Arrays



- In unsorted array implementation of dictionaries, **`insert`** is a constant-time operation, because it simply inserts the new record at the end of the list.
- However, **`find`**, and **`remove`** both require $\Theta(n)$ time in the average and worst cases, because we need to do a sequential search.



Array-Based implementation of Dictionary

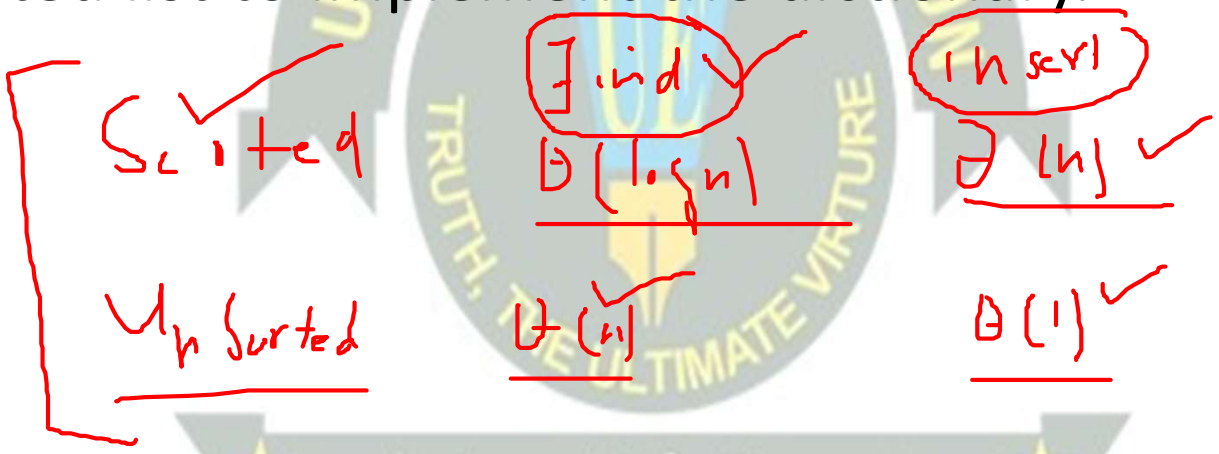
- Method **`remove`** in particular must touch every record in the list, because once the desired record is found, the remaining records must be shifted down in the list to fill the gap.
- Method **`removeAny`** removes the last record from the list, so this is a constant-time operation.
- Another alternative would be to implement the dictionary with a sorted list.
- A sorted list is somewhat different from an unsorted list in that it cannot permit the user to control where elements get inserted.
- The **`insert`** method must be quite different in a sorted list than in an unsorted list.

Array-Based implementation of Dictionary

- The advantage of this approach would be that we might be able to speed up the **`find`** operation by using a binary search.
- The cost for **`find`** in a sorted list is $\Theta(logn)$ for a list of length n.
- The cost of **`insert`** changes from constant time in the unsorted list to $\Theta(n)$ time in the sorted list.
- Whether the sorted list implementation for the dictionary ADT is more or less efficient than the unsorted list implementation depends on the relative number of **`insert`** and **`find`** operations to be performed.

Array-Based implementation of Dictionary

- If many more **`find`** operations than **`insert`** operations are used, then it might be worth using a sorted list to implement the dictionary.



Next Lecture

- In next lecture, we will start our discussion on sorting algorithms.