15CSE374
INTRODUCTION TO DATA STRUCTURES AND ALGORITHMS

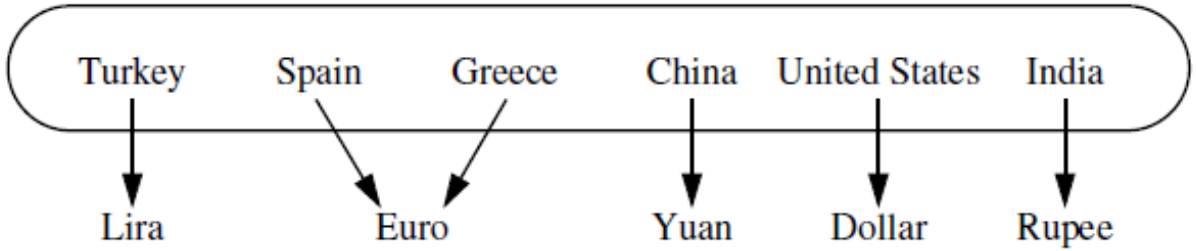
Sarath tv

Last Lecture

- Array representation of B.Tree
- Priority Queue
- Heap
  - Heap property
  - Min heap
  - Max heap.

Maps and Dictionaries

- Key : Values association.
- Associative arrays.
- Name of countries to their associated unit of currency.
- Keys –unique.
- Values not necessarily unique.
- Indices for a map need not be consecutive nor even numeric.



Some applications

- University information system- Student ID – Students record.
- Domain name system – host name – IP address.
- Social media site- username as key –Mapped to user associated information.
- Counting Word Frequencies-we can use words as keys and word counts as values.

MAP ADT

- Define the significant behavior of MAP ADT.
- `M[k]`: Return the value v associated with key k in map M, if one exists; otherwise raise a KeyError.
- `M[k] = v`: Associate value v with key k in map M, replacing the existing value if the map already contains an item with key equal to k.
- `del M[k]`: Remove from map M the item with key equal to k; if M has no such item, then raise a KeyError.
- `len(M)`: Return the number of items in map M.

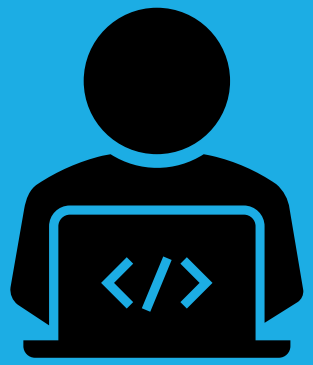
Additional interfaces

- `M.get(k, d=None)`: Return M[k] if key k exists in the map; otherwise return default value d. This provides a form to query M[k] without risk of a KeyError.
- `M.setdefault(k, d)`: If key k exists in the map, simply return M[k]; if key k does not exist, set M[k] = d and return that value.
- `M.pop(k, d=None)`: Remove the item associated with key k from the map and return its associated value v. If key k is not in the map, return default value d (or raise KeyError if parameter d is None).
- `M.popitem()`: Remove an arbitrary key-value pair from the map, and return a (k,v) tuple representing the removed pair. If map is empty, raise a KeyError.
- `M.clear( )`: Remove all key-value pairs from the map.

- `M.keys( ):` Return a set-like view of all keys of M.
- `M.values( ):` Return a set-like view of all values of M.
- `M.items( ):` Return a set-like view of (k,v) tuples for all entries of M.
- `M == M2:` Return True if maps M and M2 have identical key-value
- associations.
- `M != M2:` Return True if maps M and M2 do not have identical keyvalue associations.

Effect of a series of operations on an initially empty map

Operation	Return Value	Map
len(M)	0	{}
M['K'] = 2	–	{'K': 2}
M['B'] = 4	–	{'K': 2, 'B': 4}
M['U'] = 2	–	{'K': 2, 'B': 4, 'U': 2}
M['V'] = 8	–	{'K': 2, 'B': 4, 'U': 2, 'V': 8}
M['K'] = 9	–	{'K': 9, 'B': 4, 'U': 2, 'V': 8}
M['B']	4	{'K': 9, 'B': 4, 'U': 2, 'V': 8}
M['X']	KeyError	{'K': 9, 'B': 4, 'U': 2, 'V': 8}
M.get('F')	None	{'K': 9, 'B': 4, 'U': 2, 'V': 8}
M.get('F', 5)	5	{'K': 9, 'B': 4, 'U': 2, 'V': 8}
M.get('K', 5)	9	{'K': 9, 'B': 4, 'U': 2, 'V': 8}
len(M)	4	{'K': 9, 'B': 4, 'U': 2, 'V': 8}
del M['V']	–	{'K': 9, 'B': 4, 'U': 2}
M.pop('K')	9	{'B': 4, 'U': 2}
M.keys()	'B', 'U'	{'B': 4, 'U': 2}
M.values()	4, 2	{'B': 4, 'U': 2}
M.items()	('B', 4), ('U', 2)	{'B': 4, 'U': 2}
M.setdefault('B', 1)	4	{'B': 4, 'U': 2}
M.setdefault('A', 1)	1	{'A': 1, 'B': 4, 'U': 2}
M.popitem()	('B', 4)	{'A': 1, 'U': 2}



THANK YOU!!!!!