Class Map(k,v)<Integer, Integer>

Constructor: assign random int, generate random value

Public int Hashcode(): generates key of 16 bits from above randomly generated key

Abstract class Hash table

Variables:

static capacity

int tablesize (number of elements)

collisionNum

Array for pairs

Instantiates map(k,v)

tableSize()

Isempty

Get(K)

Public int:value Put(k,v)

Measures run time, output toString for time, numOfelements, collisionNum, capacity, probingnumber/bucketitems

Remove(k)

Int hashfunction(k)

Helper function to define input location (collision resolution), implemented in extended abstract class

toString()

getruntime()

resize(): doubles size if half of hashtable is full

Class seperatechaininghashtable()

bucketItems

inner class linkedlinst (for every spot in hashtable)

size == bucketitems

implement collision mechanism

class linear probinghashtable extends hashtable class

numOfProbing

collision mech

class quadraticprobinghashtable extends hashtable class

numOfProbing

collision mech

Main:

Instantiate each type of hashtable with capacity 100

Array of map: Generate 50 random pairs

Put each pair into each hashtable making sure to use same pairs

Note time for

Oli:

* Main
* Abstract map
* Abstract hashtable
* Quadratic
* Theory: 7,8, 3,4

Hubert:

* Chain (linkedlist)
* Linear
* Map
* Theory: 5,6,1,2