Session 2 Hash Tables

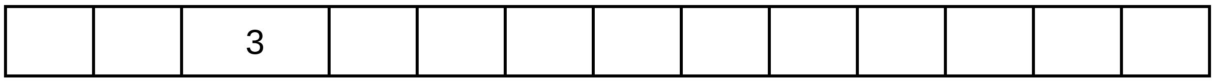
What are hash tables:
In computing, a hash table, also known as hash map, is a data structure that implements an associative array or dictionary. It is an abstract data type that maps keys to values.

Why hash tables:
-String 1 and 2 I want to compare s1 to s2, the time complexity is O(length smallest). Suppose we have a DNA sequence which is around 10^16 so I have a huge operation instead we use a hash table.
-Get and put consume O(1).

Unordered_multiset:
Unordered_multiset is a hashtable. It allows for O(1) insertion and deletion of elements, but it can take longer to find an element. So, it can be a good choice if you need to perform many insertions and deletions, but not many lookups.

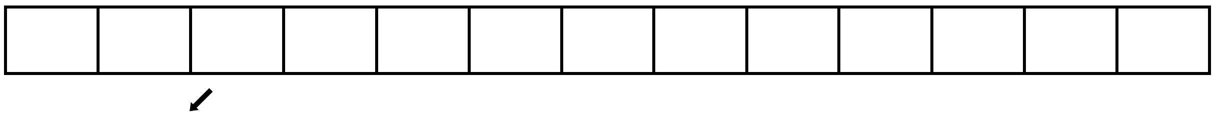
Collision:

put(37,3),m=5 and h(x) return value%m=>2

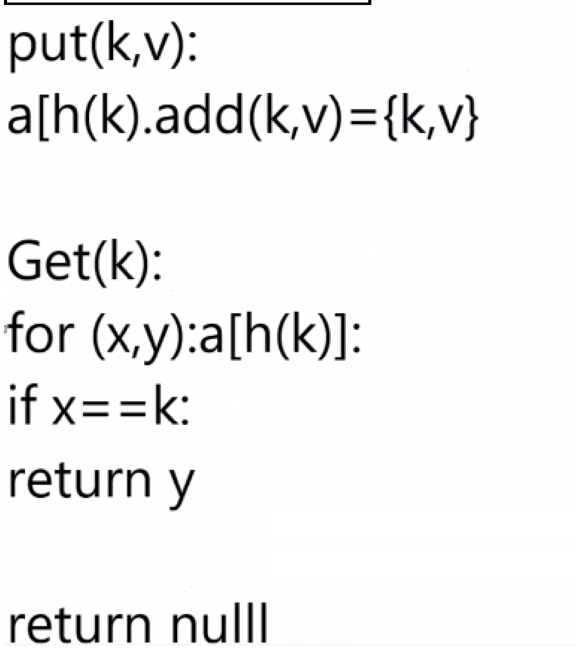


put(52,2),m=5 and h(x) return value%m=>2
We pop into a problem where 2 is already occupied.

Collision is when we have 2 hash functions with the same value to avoid it we use arraylists with pairs where each value is saved with its key.
Put is constant and get is linear but we need to fix the linear time issue.



37,3
52,5



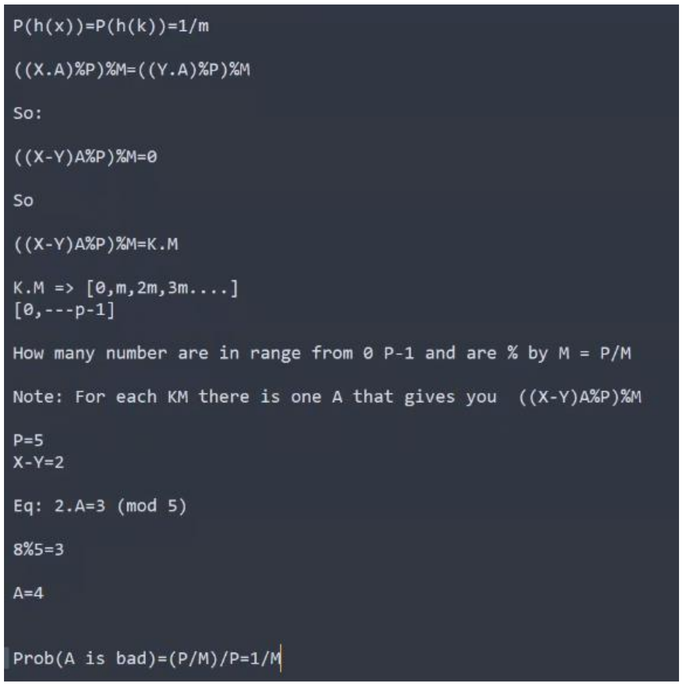
```
put(k,v):
a[h(k).add(k,v)={k,v}

Get(k):
for (x,y):a[h(k)]:
if x==k:
return y

return nulll
```

We can notice that insertion (put) would require O(1), concerning the get:
Summation of get= summation of h(x)=summation of h(y)=probability of h(x)=1/m

<u>Randomization:</u>
h(k)=((k.A)%P)%M
P is a prime number, A is a random number between 0 and P-1
-Prime numbers modulo lead to a unique number that's why we prefer using them in a hash function.
-We choose randomly because it's extremely hard for 2 hash functions to give the same answer.



```
P(h(x))=P(h(k))=1/m

((X.A)%P)%M=((Y.A)%P)%M

So:

((X-Y)A%P)%M=0

So

((X-Y)A%P)%M=K.M

K.M => [0,m,2m,3m....]
[0,---p-1]

How many number are in range from 0 P-1 and are % by M = P/M

Note: For each KM there is one A that gives you  ((X-Y)A%P)%M

P=5
X-Y=2

Eq: 2.A=3 (mod 5)

8%5=3

A=4


Prob(A is bad)=(P/M)/P=1/M
```

<u>Security:</u>
I can do a security check if the size of a certain array in the arraylist got really big. I can stop and go take a new hash function and taking it randomly makes it hard to figure out which function is operating. When taking a new hash function we have to repeat the program from the beginning. So even if someone got into the system it is approximately impossible for them to figure out which function is being used.

<u>Open Addressing:</u>
It is a method that uses arrays, while trying to insert if a cell is not empty we move to the neighbour cell until we find an empty spot.

put(37,3)=> h(2)

		3										

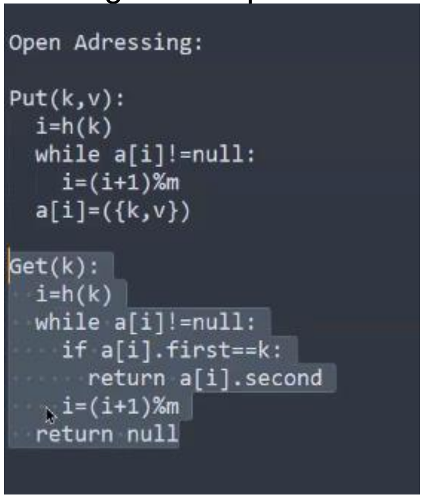
put(52,5)=> h(2) go to 2 it's already occupied go to next cell

		3	5									

So if we are looking for (7,2)==>h(2) we start from 3 and move on if we reach a blank cell means that our value is not there so we stop and return

Problem we might face is having an array full, in such cases we can double the size of the array.

We might also use a method of jumping not by moving sequentially but this would grow the problem of caching discussed below.



```
Open Adressing:

Put(k,v):
  i=h(k)
  while a[i]!=null:
    i=(i+1)%m
  a[i]=({k,v})

Get(k):
  i=h(k)
  while a[i]!=null:
    if a[i].first==k:
      return a[i].second
    i=(i+1)%m
  return null
```

Why are arrays better than list:
-Listes require more memory consumption
-If we want to insert the nodes, in list nodes are shuffled and it’s bad for time due to caching so we jump from 1 to 100 then 100 to 1000 so this makes caching harder whereas an array doesn't have random access to memory so it makes caching better.
Arrays are easier to propagate on memory and time wise.

Notes:
-Most things on a PC are made up of arrays so map and set are made up of arrays, above an array there is an algorithm.
-Java uses binary search trees for hash tables.
-Circular Array if I am at the end of the array in order not to go out of bounds we use %size of array (discussed in OS).
-During interviews never stop talking when you get stuck, ask questions, ask for hints, try to explain your thoughts but never sit in silence.
-To erase a single element from a multiset either erase at a specific index or erase with a counter of 1 so that not all equal elements to that value are erased.