HashPipe

Smoked by Sergiy Isakov

My first attempt was unsuccessful because of running time, though algorithm worked as it should. I.e. it passed all tests except the one in TestsBest. That was because of wrong pipe structure. The link array was only on the root node, so I lost opportunity of logarithmic search time, looking for the floor node consequently for a given height from the root node.

But now I'm going to describe a successful solution which I literally made by lapping holes in the program until I could make it work as I wished. I called nodes as "Nope" (combination of Node and Pipe).

The class Nope has fields: int *hash*, Nope[] *next*, String *key* and Integer *value*.

Class HashPipe.

Fields: Nope *root*, int *size*, int *rootMaxSize*, Nope[] *previousNopes* and Nope *nope*. The root Nope is initialized at height 32 as it was suggested. So I added a variable *rootMaxSize* to keep track of useful height of root Nope(i.e. that has links to other Nopes). An array *previousNopes* is a temporary container of previous Nopes that are needed to update links each time new Nope is added.

Methods:

hash(String key). This method converts hashCode of key String to binary String and then counts trailing zeros using a private method trailingZeros(String str).

put(String key, Integer val). First it creates a new global Nope variable of height calculated by hash method. Then it finds a previous Nope using method floorNope(key). "If statement" checks then if table has this key already and sets new value if it is the case. If not then it updates links for new Nope using pre-calculated array *previousNopes.* At the end *size* variable increments by 1.

floorNope(String key). This method starts scanning links from the top of root Nope which initially assigned to variable *floor*. If the link points to a Nope that is less than the key, then it continues loop but from this pointed Nope(so this Nope is assigned to *floor*). If that Nope is more than the key, then it just goes one level down and so on until it reaches the ground level 0. And obviously at this level the Nope which was used in *floor* variable is an actual floor for the key. Meanwhile this method also creates a temporary array of previous Nopes and height of array is a height of new Nope. So this *previousNopes* array is used in put method to update links.

control(String key, int h). It simply uses *floorNope* method to find the Nope and then after some checks it returns whether NULL or next key at height *h*.

Additionally I wrote method **Iterable<String> keys(String lo, String hi).** It overrides methods of interfaces Iterable and Iterator. I just wrote few lines defining behavior of hasNext() and next() methods.

I also made generic implementation of HashPipe, simply changing String by K and Integer by V, but CodeJudge cannot test it.