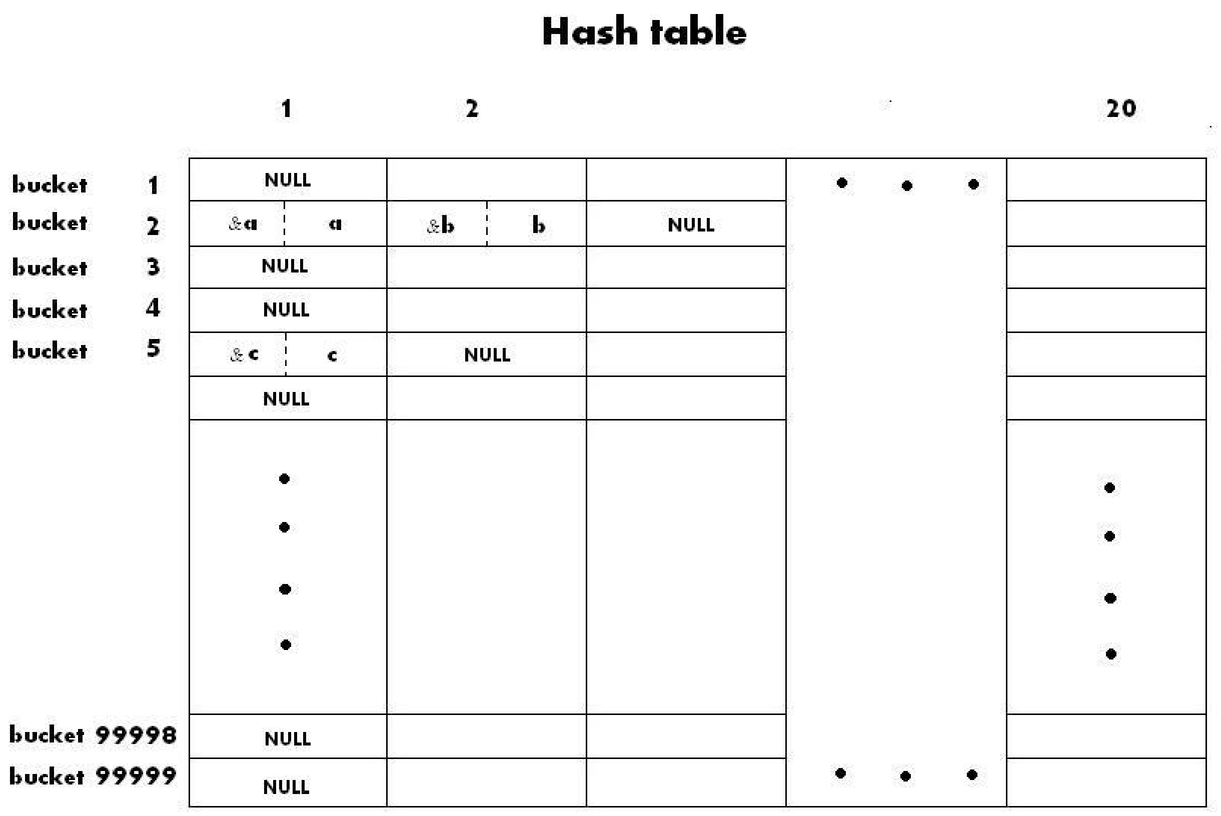
Following your question, here is the description of our implementation of maman 12.
Hash table contains 99999 buckets. Each bucket is an array of 20 entries that where mapped to that bucket by the hash function. The hash function is a mapping of the pointer that points to the allocated region to some HT bucket:

Lets see an example:
Int * a, *b, *c;
a= malloc(&a, byteNum);
assignPointer(&b.&a);
c = malloc(&c, byteNum);

Since a and b are pointing to the same region their values are identical, so the hashing function maps them to the same bucket. But is might happen that c will be also mapped to that bucket (even thought this is only a possibility that depends on the size of HT and the mapping function).

Entries of the HT must contain the address of the pointer that points to the allocated region. Since the library moves the allocated regions we need to store somewhere the pointer to pointer. And the storage for the meta data is our HT.


Hash table

	1	2			20
bucket 1	NULL			● ● ●	
bucket 2	&a ¦ a	&b ¦ b	NULL		
bucket 3	NULL				
bucket 4	NULL				
bucket 5	&c ¦ c	NULL			
	NULL				
	● ● ● ●				● ● ● ●
bucket 99998	NULL				
bucket 99999	NULL			● ● ●	

Every time the memcompaction function is about to move some allocated region, we have to traverse the bucket that corresponds to that region and to update each pointer that were pointing to that region.

Registering/unregistering allocated memory regions is done by updating hash table.