Consider open addressing with linear probing and an attempt to see whether a value e is in the set. If e hashes to h, then buckets with indexes h % b.length, (h+1) % b.length, (h+2) % b.length, ... are probed until either e is found or a bucket containing null is found.

e1 e2 e3 e4

0 1 2 3 4 5 6 7

b

Linear probing can result in *clustering*: many values occupy successive buckets, as shown to the right, leading to excessive probes to determine whether a value is in the set. Here, e1 could have hashed to bucket 2, then e2 and e3 to bucket 3, and finally e4 to bucket 2. If e5   
hashes to 2, then five probes are necessary to determine that e5 is not in the set.

Quadratic probing is designed to reduce the amount of clustering that may take place. The idea is simple enough. If a value hashes to h, then quadratic probing looks at the elements with indexes:

h % b.length  
 (h+1^2) % b.length = (h+1) % b.length  
 (h+2^2) % b.length = (h+4) % b.length  
 (h+3^2) % b.length = (h+9) % b.length  
 (h+4^2) % b.length = (h+16) % b.length,  
 ...

The table to the right shows what quadratic probing would produce for inserting values e1, e2, e3, and e4 given above. Clustering has been reduced; there is now a bucket with null between e3 and e4.

e1 e2 e3 e4

0 1 2 3 4 5 6 7

b

**Major problem with quadratic probing**

Quadratic probing does not always work!

Suppose the size of array b is 8, as shown in the two tables to the right above. Suppose value e hashes to 0. Then the buckets probed are:

0^1 % 8 = 0  
 1^2 % 8 = 1  
 2^2 % 8 = 4  
 3^2 % 8 = 1  
 4^2 % 8 = 0  
 5^2 % 8 = 1

Thus, some buckets are probed several times, and it is not clear that all the buckets will be probed.

However, it has been proven that if the table size is a prime, then all the buckets will be probed using quadratic probing. For example, if the table size is 7, we have:

0^2 % 7 = 0  
 1^2 % 7 = 1  
 2^2 % 7 = 4  
 3^2 % 7 = 2  
 4^2 % 7 = 2  
 5^2 % 7 = 4  
 6^2 % 7 = 1  
 7^2 % 7 = 0  
 8^2 % 7 = 1  
 9^2 % 7 = 4