Describing separate chaining and linear probing:

In the lab of hash table, there are two reasonable data structure designs, separate chaining and linear probing. Linear probing belongs to Open Addressing. By using linear probing, we treat hash table (h[0,1..m])as circular array. When we are doing searching purpose, let’s say if the initial address we are looking for is i = hash(key) % m. The worst case to search an element will be: i, i + 1, i + 2, … , m-1, 0, 1 , … , i-1 (hash(key) is hash function, m is the size of hash table). Similarly, the way of handling collision is to check if the current address is empty, if yes, insert the element into that bucket. Otherwise increase the address by 1 till you find an empty bucket. You also need to resize the hash table if the last bucket in the hash table is not empty when you are doing insertion.

By using separate chaining, if the size of hash table is m, then the hash table can be defined as a pointer array which is built by m different pointers(hashtable[0…m-1]), each pointer is a bucket. If the elements you are inserting to this hash table share the same bucket, you need to append the element to the back of that bucket, just like adding elements at the end of linked-list. In this case we don’t need to resize the hash table when inserting elements.

Trade-offs between them:

By using linear probing, it is very easy to cause clustering. In this hash table, if the bucket i, i+1, … ,i+k are not empty, then the elements you want to insert into the bucket i, i+1, … , i+k+1 will all be inserting in the bucket i+k+1. However, we don’t need to worry about it anymore by using separate chaining. In real application, using linear probing requests the load factor is relatively small in order to avoid wasting memory. The disadvantage of separate chaining is each pointer needs extra memory, so it will be a better choice to use linear probing if the number of element you want to insert is relatively small, you could increase the size of the hash table so the load factor will be smaller which can reduce collision. However, the load factor of separating chaining could be bigger than one, and it can save a lot of memory if the number of element you want to insert into the hash table is big. Using separate chaining can delete the elements in the hash table easily, but you cannot really delete the elements from linear probing hash table, you can only mark the elements that are deleted. Using separate chaining will also be your better choice if you cannot decide the size of hash table, because separate chaining is using dynamic memory allocation.