Exercise 2:

(i) We will apply hashing with chain here.

The basic principle here is to define a key value k for each relation pair (a, b), then use a hash function $hash(k)$ to calculate the hash value of each pair. We will put all the pairs with the same $hash(k)$ into the same entry of hash table. Finally we do the checking operation.

Assume the number of relation pairs is n.

To implement this idea, we will first loop through all the relation pairs, calculating their key values. If we meet a pair of the form (a, a), we simply skip it since it is already symmetric. The key is defined as following: $k = (a * b) \gg 2$. The reason why we right shift the product by 2 is that we want to remove the last 2 binary bits to avoid too much replications. This step takes $O(n)$ complexity.

Second, we will convert the relation pair into a structure $Node\_t$. This structure is composed with the relation pair itself as well as its key value k. This step takes $O(n)$ complexity.

Third, we will calculate the hash value using a hash function $hash(k) = k \ mod \ (\frac{n}{m})$ for each structure, and put the structure into hash table. Here n is the number of relation pairs, m is an integer between 10-30. So the hash table has the length $\frac{n}{m}$, each entry consist a list of all structures with the same hash value. This step takes $O(n)$ complexity.

Finally, we perform the checking operation. Start this by checking whether all the entries have a list with even length. If not, the relation is not symmetric. This step takes $O(\frac{n}{m})$ complexity.

Then from the list in the first entry of the hash table, starting with the first structure $Node\_t$ in that list, loop through that list to find the symmetric pair. If no symmetric pair is found, the relation is not symmetric. If found, simply remove the found structure and the first structure. Then do this step again until we find the relation is not symmetric or the list becomes empty. When we empty the list in the first entry, we move on to the second entry and so on. If we empty the whole hash table, the relation is symmetric.

In this step of checking, we will choose to analyze the average time complexity. That is, all the relation pairs are uniformly distributed into the hash table. We have $\frac{n}{m}$ lists, each list has m structures. In dealing with each list, we need no more than $O(m^2)$ steps, so the overall complexity is $O(\frac{n}{m} * m^2) = O(nm)$.

So for the whole process, the average time complexity is $O(mn)$. Since m is a constant between 10-30, we can get a much better complexity than $O(n^2)$ when n is large.