Leetcode May Challenge DAY: 23

1. Python

```
class Solution(object):
    def intervalIntersection(self, A, B):
        """
        :type A: List[List[int]]
        :type B: List[List[int]]
        :rtype: List[List[int]]
        """
        la = len(A)
        lb = len(B)
        pa = 0
        pb = 0
        ret = []
        while pa<la and pb<lb:
            cr = [None,None]
            cr[0] = max(A[pa][0], B[pb][0])
            if A[pa][1]<=B[pb][1]:
                cr[1] = A[pa][1]
                pa+=1
            else:
                cr[1] = B[pb][1]
                pb+=1
            if cr[0]<=cr[1]:
                ret.append(cr)
        return ret
```

2. C++

```
class Solution {

public:

   vector<vector<int>> intervalIntersection(vector<vector<int>>& A,
vector<vector<int>>& B) {

      vector<vector<int>>ans;

      int i=0,n=A.size();

      int j=0,m=B.size();

      while(i<n && j<m)

      {

         int l = max(A[i][0],B[j][0]);

         int r = min(A[i][1],B[j][1]);

         if(l<=r)

         {

            ans.push_back({l,r});

         }

         if(A[i][1] <= B[j][1]) i++;

         else j++;

      }

      return ans;

   }

};
```

3. JAVA

```
class Solution {
    public int[][] intervalIntersection(int[][] A, int[][] B) {

        List<int[]> result = new ArrayList<>();

        int i = 0;
        int j = 0;

        while (i < A.length && j < B.length) {
            // choose the interval with the larger end point
            if (A[i][1] > B[j][1]) {
                j = intervalIntersection(B, A[i], j, result);
                i++;
            } else {
                i = intervalIntersection(A, B[j], i, result);
                j++;
            }
        }

        return result.toArray(new int[0][]);

    }
```