Maximum Length of Pair Chain

You are given `n` pairs of numbers. In every pair, the first number is always smaller than the second number.

Now, we define a pair `(c, d)` can follow another pair `(a, b)` if and only if `b . Chain of pairs can be formed in this fashion.`

Given a set of pairs, find the length longest chain which can be formed. You needn't use up all the given pairs. You can select pairs in any order.

Example 1:

```
Input: [[1,2], [2,3], [3,4]]
Output: 2
Explanation: The longest chain is [1,2] -> [3,4]
```

Note:

1. The number of given pairs will be in the range [1, 1000].

Solution 1

This is equivalent to interval scheduling problem.

```
public int findLongestChain(int[][] pairs) {
    Arrays.sort(pairs, (a,b) -> a[1] - b[1]);
    int sum = 0, n = pairs.length, i = -1;
    while (++i < n) {
        sum++;
        int curEnd = pairs[i][1];
        while (i+1 < n && pairs[i+1][0] <= curEnd) i++;
    }
    return sum;
}
```

written by compton_scatter original link here

Solution 2

Reference: http://www.geeksforgeeks.org/dynamic-programming-set-20-maximum-length-chain-of-pairs/

```
public class Solution {
    public int findLongestChain(int[][] pairs) {
        Arrays.sort(pairs, (a, b) -> (a[0] - b[0]));

        int i, j, max = 0, n = pairs.length;
        int dp[] = new int[n];

        for (i = 0; i < n; i++) dp[i] = 1;

        for (i = 1; i < n; i++)
            for (j = 0; j < i; j++)
                if (pairs[i][0] > pairs[j][1] && dp[i] < dp[j] + 1)
                    dp[i] = dp[j] + 1;

        for (i = 0; i < n; i++) if (max < dp[i]) max = dp[i];

        return max;
    }
}
```

written by shawngao original link here

Solution 3

EDIT : Other solutions are better than this since they sort by pair ends, instead of pair starts. :)

The idea is to sort the pairs on their start.
The rest of the idea is to do the longest increasing subsequence with binary search. Basically first ignore the ones with duplicate start(since you've already picked the best choice). For other pairs, don't take action if something needs to be replaced in the chain with same 'end'. Also, consider them equal (`a[1] == b[0]`) and take no action on them.

```
public int findLongestChain(int[][] pairs) {
    Arrays.sort(pairs, (a,b)->a[0] != b[0] ? a[0] - b[0] : a[1] - b[1]);

    List<int[]> chain = new ArrayList<>();
    chain.add(pairs[0]);
    for(int i = 1; i < pairs.length; i++) {
        int[] pair = pairs[i];
        if(pairs[i][0] == pairs[i-1][0]) continue;
        int idx = Collections.binarySearch(chain, pair,(a,b) -> a[1] == b[0] ? 0 :
a[1] - b[1]);
        if(idx < 0) {
            if(-idx-1 < chain.size()) chain.set(-idx-1, pair);
            else {
                if(chain.get(chain.size()-1)[1] > pair[1]) {
                    chain.set(-idx-1-1, pair);
                }else if(chain.get(chain.size()-1)[1] < pair[0]){
                    chain.add(pair);
                }
            }
        }
    }
    return chain.size();
}
```

written by jaqenhgar original link here