

You are given n pairs of numbers. In every pair, the first number is always smaller than the second number. A pair (c, d) can follow another pair (a, b) if $b < c$. Chain of pairs can be formed in this fashion. Find the longest chain which can be formed from a given set of pairs.

Source: [Amazon Interview | Set 2](#)

For example, if the given pairs are $\{\{5, 24\}, \{39, 60\}, \{15, 28\}, \{27, 40\}, \{50, 90\}\}$, then the longest chain that can be formed is of length 3, and the chain is $\{\{5, 24\}, \{27, 40\}, \{50, 90\}\}$

This problem is a variation of standard [Longest Increasing Subsequence](#) problem. Following is a simple two step process.

- 1) Sort given pairs in increasing order of first (or smaller) element.
- 2) Now run a modified LIS process where we compare the second element of already finalized LIS with the first element of new LIS being constructed.

The following code is a slight modification of method 2 of [this post](#).

```
#include<stdio.h>
#include<stdlib.h>

// Structure for a pair
struct pair
{
    int a;
    int b;
};

// This function assumes that arr[] is sorted in increasing order
// according the first (or smaller) values in pairs.
int maxChainLength( struct pair arr[], int n)
{
    int i, j, max = 0;
    int *mcl = (int*) malloc ( sizeof( int ) * n );

    /* Initialize MCL (max chain length) values for all indexes */
    for ( i = 0; i < n; i++ )
        mcl[i] = 1;

    /* Compute optimized chain length values in bottom up manner */
    for ( i = 1; i < n; i++ )
        for ( j = 0; j < i; j++ )
            if ( arr[i].a > arr[j].b && mcl[i] < mcl[j] + 1)
                mcl[i] = mcl[j] + 1;

    // mcl[i] now stores the maximum chain length ending with pair i

    /* Pick maximum of all MCL values */
    for ( i = 0; i < n; i++ )
        if ( max < mcl[i] )
            max = mcl[i];

    /* Free memory to avoid memory leak */
    free( mcl );

    return max;
}
```

```
/* Driver program to test above function */
int main()
{
    struct pair arr[] = { {5, 24}, {15, 25},
                          {27, 40}, {50, 60} };
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Length of maximum size chain is %d\n",
           maxChainLength( arr, n ));
}
```

```
    return 0;  
}
```

Output:

Length of maximum size chain is 3

Time Complexity: $O(n^2)$ where n is the number of pairs.

The given problem is also a variation of [Activity Selection problem](#) and can be solved in $(n \log n)$ time. To solve it as a activity selection problem, consider the first element of a pair as start time in activity selection problem, and the second element of pair as end time. Thanks to Palash for suggesting this approach.