最长上升子序列算法

题目描述：一个序列有N个数：A[1],A[2],...,A[N]，求出最长上升子序列的长度（LIS：longest increasing subsequence)。例如，对于序列(1, 7, 3, 5, 9, 4, 8)，有它的一些上升子序列，如(1, 7), (3, 5, 9) , (3, 4, 8)等等。这些子序列中最长的长度是4，比如子序列(1, 3, 5, 9) ，(1, 3, 5, 8)和(1, 3, 4, 8).

最优解结构特征

对于处于第i个的数，它的最长序列一定是前一个小于它的数的最长序列+1或者原来的最长序列。

定义最优解

$$dp[i] = max(dp[i], dp[j] + 1);$$

其中，j必须满足，i的前一个小于a[i]的数。

计算最优解

```
vector<int> longest_increasing(vector<int> &a, vector<int> &dp){
    for (int i = 0; i < a.size(); i++) {
        for (int j = 0; j < i; j++) {
            if (a[i] > a[j]) {
                a[i] = max(a[i], a[j] + 1);
            }
        }
    }
    return dp;
}
```

构造最优解

对于最长序列数组，先找到最大值的位置i以及最长序列lengthi，然后再寻找前一个小于它的数并且子序列长度为l-1的数，递归直到lengthi = 1；

```
vector<int> find_longest(vector<int> &a, vector<int> &dp){
    vector<int> subsequence;
    int max = 0;
    int max_index = 0;
    int flag, flag_index;
    for (int i = 0; i < a.size() ; i++) {
        if (dp[i] > max) {
```

```
            max = dp[i];
            max_index = i;
        }
    }
    flag = max;
    flag_index = max_index;
    subsequence.push_back(max_index);
    while (flag != 0) {  // no subsequence, than break
        for (int i = flag_index - 1; i >= 0; i--) {
            if (a[i] == flag - 1 && a[i] > a[flag_index]) {
                flag = flag - 1;
                flag_index = i;
                subsequence.push_back(i);
            }
        }
    }
    return subsequence;
}
```