Homework 5

ACM1501　　U201514716　　罗海旻

Exercise 1. In this exercise we explore strengthening the contracts on in-place sorting functions.

1. Write a function is_permutation which checks that one segment of an array is a permutation of another.

2. Extend the specifications of sorting and partitioning to include the permutation property.

3. Discuss any specific difficulties or problems that arise. Assess the outcome.

1.解：

```
bool is_permutation(int[] A, int[] B,int n,int l1, int r1, int l2, int r2)
//@requires 0<=l1 && l1<=r1 && r1 < n
//@requires 0<=l2 && l2<=r2 && r2 < n
//@esures \result == true || \result == false
{
    if (r1 - l1 != r2 - l2) return false;
    bool[] C = alloc_array(bool, r2 - l2 + 1);
    for (int i = 0; i < r2 - l2 + 1; i++) C[i] = true;
    for (int i = 0; i < r1 - l1 + 1; i++) {
        for (int j = 0 ; j < r2 - l2 + 1; j++) {
            if (A[l1+i] == B[l2+j] && C[j]) {
                C[j] = false;
                break;
            }
        }
    }
    for (int i = 0; i < r2 - l2 + 1; i++)
        if (C[i]) return false;
    return true;
}
```

2.解：

无法做到这一点，由于是 in-place sort，无法获取排序之前的原始数组(除非排序之前进行数组的拷贝)。

3.

Exercise 2. Prove that the precondition for sort together with the contract for partition implies the post-condition. During this reasoning you may also assume that the contract holds for recursive calls.

解：

1. Upper - lower <= 1 时，函数返回，此时仅有一个或不含元素，显然有序
2. Upper - lower > 1 时，假设划分后的两个子数组排序结果正确。由

partition 后置约束条件可知，pivot 左侧元素小于 pivot 值，右侧元素大于 pivot 值，则整个数组排序正确。

综上所述，qsort 的后置元约束满足，即得到正确排序的数组。

Exercise 3. Our implementation of partitioning did not pick a random pivot, but took the middle element. Construct an array with seven elements on which our algorithm will exhibit its worst-case behavior, that is, on each step, one of the partitions is empty.

解：[6,4,2,1,3,5,7]

Exercise 4. An alternative way of implementing the partition function is to use extra memory for temporary storage. Develop such an implementation of

```
1 int partition(int[] A, int lo, int pi, int hi)
2 //@requires 0 <= lo && lo <= pi;
3 //@requires pi < hi && hi <= \length(A);
4 //@ensures lo <= \result && \result < hi;
5 //@ensures ge_seg(A[\result], A, lo, \result);
6 //@ensures le_seg(A[\result], A, \result+1, hi);
```

解：代码如下：

```
int partition(int[] A, int lower, int pivot_index, int upper)
//@requires 0 <= lower && lower <= pivot_index;
//@requires pivot_index < upper && upper <= \length(A);
//@ensures lower <= \result && \result < upper;
//@ensures ge_seg(A[\result], A, lower, \result);
//@ensures le_seg(A[\result], A, \result+1, upper);
{
  int pivot = A[pivot_index];
  swap(A, pivot_index, upper-1);

  int left = lower;
  int right = upper-1;
  int[] t = alloc_array(int, upper - lower);
  for (int i = lower; i < upper ; i++) {
    if (i == pivot_index);
    else if (A[i] <= pivot) {
      t[left - lower] = A[i];
      left++;
    } else {
      t[right - lower] = A[i];
      right--;
    }
  }
  t[left - lower] = A[pivot_index];
```

```
    for (int i = lower ; i < upper ; i++) A[i] = tmp[i-lower];
    return left;
}
```