CSP2348-5243 Data Structures

Solutions to Tutorial 03

Exercise 1:

code	##	cost

int ascending(int[] array, boolean unique) {
  if (array==null) return -1;
  int n=array.length;
  for (i=1; i<n; i++){
  if (array[i-1]>array[i]) return i;
  if (unique && array[i-1]==array[i]) return i;
  }
  return n;
}
	01
	02
	03
	04
	05
	06
	07
	08
	09
		O(1)
		O(1)
		O(1)
		O(1)×#e
		O(1)×#e
		O(1)×#e
		-
		O(1)
		-

loop-control=1,2,3,…,n-1 #e = number of executions ≈ O(n) maximum cost = O(n)		

Exercise 2:

code	#	cost

int searchLinear(int[] array, int item) {
  if (array==null) return -1;
  for (int i=0; i<array.length; i++)
  if (array[i]>=item)
  return i;
  return array.length;
}
	01
	02
	03
	04
	05
	06
		O(1)
		O(1)×#e
		O(1)×#e
		O(1)×#e
		O(1)×#e
		O(1)

loop-control = 0,1, 2, 3,…, n-1 #e = number of executions ≈ O(n) maximum cost = O(n)		

Exercise 3:

code	#	cost
public void merge(int[] a1, int l1, int r1, int[] a2, int l2, int r2, int[] a3, int l3) {	01	O(1)
//merge existing a1[l1,...,r1] and existing a2[l2,...,r2]	02	-
//into existing a3[l3], where a1 and a2 are sorted	03	-
int i = l1, j = l2, k = l3;	04	O(1)
while (i <= r1 && j <= r2) {	05	O(1)x#e1
if (a1[i] <= a2[j]) {	06	O(1)x#e1
a3[k++] = a1[i++];	07	O(1)x#e1
} else {	08	-
a3[k++] = a2[j++];	09	O(1)x#e1
}	10	-
}	11	-
while (i <= r1) {	12	O(1)x#e2
a3[k++] = a1[i++];	13	O(1)x#e2
}	14	-
while (j <= r2) {	15	O(1)x#e3
a3[k++] = a2[j++];	16	O(1)x#e3
}	17	-
}	18	-

1st while loop: Loop-control1 = 1, 2, … n_1+n_2
#e1 = number of execution ≈ $O(n_1+n_2)$)

(Note: 2nd & 3rd while loops would execute one of them only)

2nd while loop: Loop-control2 = …,n_1
#e2 = number of execution ≈ $O(n_1)$

3rd while loop: Loop-control2 = …,n_2
#e3 = number of execution ≈ $O(n_2)$

Maximum cost = $O(n_1+ n_2) + O(n_1) + O(n_2) = O(n_1+ n_2)$

Exercise 4 ~ 7:

See separate codes in separate documents