

# Homework 5

22C:44 Algorithms  
Due Tuesday, April 2, 2002

1. Exercise 9-3.1, page 192.
2. Exercise 9-3.2, page 192.
3. Exercise 9-3.5, page 192.
4. Exercise 33.4-1, page 960.
5. Exercise 11.2-2, page 229
6. Exercise 11.2-3, page 229
7. Exercise 11.4-1, page 244
8. A *local minimum* of an array  $A[i..j]$  is an element  $A[k]$  satisfying  $A[k - 1] \geq A[k]$  and  $A[k] \leq A[k + 1]$ .

Assume that  $i \leq j$ ,  $A[i - 1] \geq A[i]$ , and  $A[j] \leq A[j + 1]$ . These assumptions guarantee that  $A[i..j]$  contains a local minimum.

- a. Construct an efficient recursive algorithm that finds a local minimum in  $A[i..j]$ . The algorithm must have a time bound significantly better than the  $O(n)$  time bound of the “obvious” method for solving this problem.

```
/* Assume A[i-1] >= A[i] and A[j] <= A[j+1]. Return index of local minimum */
int FindLocalMinimum(int A[], int i, int j)
{
    ...
    return(index-of-local-minimum);
}
```

NOTE: in this parts b and c below, express your answers in terms of  $n$ , where  $n$  is defined to be equal to  $j - i + 1$  (the number of elements covered in a call `FindLocalMinimum(A, i, j)`).

- b. Write a recurrence equation for `FindLocalMinimum`.
- c. (3 points) Give a tight bound on the running time of `FindLocalMinimum`.

9.            `/* Given array A, and integers left and right, where left <= right,
              SILLYSORT(A,left,right) sorts the items in A[left..right].
              NOTE: this is not a good way to sort; it is logically sound but
              a bit silly.
              */
              void SILLYSORT(int A[] ; int left, int right);
              int middle;

              if (l != r) {
              middle = (left + right)/2;
              SILLYSORT(A,left,middle);
              SILLYSORT(A,middle+1,right);
              if (A[left] > A[middle+1]) then
              swap(A[left],A[middle+1]);
              SILLYSORT(?,?,?);
              }`

- a. Determine the missing arguments in the third recursive call. HINT: think carefully about what the situation is after the first two recursive calls. Then think carefully about what the situation is right before the third recursive call. Drawing a “picture” of the situation might help.
- b. Write a recurrence relation characterizing the running time of SILLYSORT.
- c. Why is this sort silly? Solving the recurrence of part b is a bit difficult. Give a guess at a tight bound (i.e. guess  $f(n)$  such that  $\text{SILLYSORT}(n)$  is  $\Theta(f(n))$ ).