Practice Questions on Recursive Algorithms

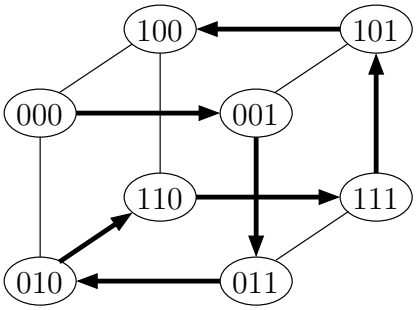
These questions (and the ones listed under readings) are good preparation for Quiz #2. They are not to be handed in. Some hints follow the questions, but don't look at them unless you are stuck after working on the problem for a while.

1. Let $n \geq 1$. Suppose we have a collection of $N = 2^n$ computers. They are connected into a hypercube network as follows. Each computer is labelled by a binary string of length n. No two computers have the same label. There is a wire connecting two computers if and only if the two labels of the computers differ in exactly one bit. For example, if $n = 8$ there would be a direct connection between the computers labelled 01001001 and 01011001 because the two labels differ only in the value of their fourth bits. When $n = 3$, the network forms a cube as shown in the diagram below.

Suppose we want the computer with label $000\ldots0$ to send some information to all other computers, but we only want to transmit messages across $N - 1$ of the edges. Willemina figured out how to do this and wrote a recursive algorithm to print out the order in which the information should be passed around the network.

```
1: LIST(n) : list of strings
2:     if n = 1 then return ⟨0, 1⟩
3:     else
4:         L ← LIST(n − 1)
5:         let x_1, x_2, ..., x_k be the elements of L (in order)
6:         return ⟨0x_1, 0x_2, ..., 0x_{k−1}, 0x_k, 1x_k, 1x_{k−1}, ..., 1x_2, 1x_1⟩
```

For example, LIST(3) outputs $\langle 000, 001, 011, 010, 110, 111, 101, 100\rangle$. Thus, according to Willemina's scheme, the message would be sent around the network as shown by the arrows below.



Prove that for all $n \geq 1$, the output of LIST(n) is correct. In other words, prove that it has the following two properties.

- Every binary string of length n is printed exactly once.
- For every pair of consecutive labels in the list, there is a wire between the two computers with those labels.

2. Give a careful proof of correctness for the following recursive algorithm.

```
1: MULTIPLY(int y, int z) : int
2:     precondition: y ≥ 0 and z ≥ 0
3:     if z = 0 then return 0
4:     else
5:         x ← MULTIPLY(y + y, ⌊z/2⌋)
6:         if z is odd then x ← x + y
7:         return x
8:     postcondition: value returned is equal to y · z
```

3. Consider the following recursive algorithm. The array A represents a binary tree that stores n keys: for $1 < i \leq n$, the parent of $A[i]$ is $A[\lfloor i/2 \rfloor]$.

```
1: IH(A[1..n], i) : boolean
2:     precondition: 1 ≤ i ≤ n
3:     if 2i ≤ n and (A[2i] > A[i] or not IH(A, 2i)) then return false
4:     else if 2i + 1 ≤ n and (A[2i + 1] > A[i] or not IH(A, 2i + 1)) then return false
5:     else return true
```

(a) Give a good, simple description of what this algorithm does by completing the following post-condition:
IH$(A[1..n], i)$ returns true if and only if ...

(b) State the claim that you would prove by induction to prove the correctness of this algorithm. How are you measuring the size of the input?

(c) Prove the claim in (b).

(d) Give a good bound on the worst-case running time of IH$(A[1..n], 1)$ using Θ notation. Show your answer is correct.

4. Suppose $A[1..n]$ is an array of integers sorted so that $A[1] < A[2] < \cdots < A[n]$. Design a recursive binary search algorithm that returns an i such that $A[i] = i$ if such an i exists (or returns 0 if no such i exists).

5. Given an array $A[1..n]$ of distinct numbers, design an algorithm to reorder A so that $A[1] < A[2] > A[3] < A[4] > A[5] < \cdots$. The asymptotic worst-case time should be as good as possible.

6. In class, we saw how to find the closest pair among a set of points in the plane. Finding the pair that is farthest apart is harder. But it is easy to find an *approximate* solution: given points $P_1, \ldots, P_n$ in the plane find a pair P_i, P_j so that for all k, ℓ, $dist(P_k, P_\ell) \leq 2 \cdot dist(P_i, P_j)$. Your solution should run in $O(n)$ time.

Hints

1. Use (regular) induction on n. Be explicit about how you use the induction hypothesis in the induction step.

To show that a label is printed exactly once, you must argue that it is printed and that it is not printed twice.

The argument about the wires in the induction step can be broken into 3 cases: a pair of consecutive labels in the first half of the output, in the second half of the output or straddling the two halves.

2. Use induction on z. Break the induction step into cases according to the test on line 6.

3. Review what you learned about heaps from EECS2101. They are also covered in Chapter 6 of the textbook.

For (b), the claim is the same as always: For all sizes s, if you call the algorithm on an input of size s that satisfies the preconditions, it terminates and satisfies the postconditions.

For (d), it is better to think in terms of the tree rather than the array. Let $T(k)$ be the worst-case running time when the algorithm is called with an index that is the root of a subtree containing k nodes. Get a recurrence for $T(k)$ and then use the recurrence to prove your bound by induction. (The Master Theorem doesn't really apply here.)

4. The test you use to recurse on one half of the array or the other should be sufficient to ensure that the answer you are looking for is *not* in the part of the array that you are discarding.

An alternative way of looking at this problem is to think of searching for the value 0 in the array $B[1..n]$ where $B[i] = A[i] - i$; the array B is sorted in non-decreasing order.

5. This can be done in $O(n)$ time. Your answer can be *extremely* short if you use one of the algorithms covered in class.

6. You can choose $i = 1$. Then how would you pick j? To show your solution is correct, you can use the triangle inequality about 3 points P, Q, R: $dist(P, R) \leq dist(P, Q) + dist(Q, R)$. Your solution should even work if points are in three-dimensional (or d-dimensional) space.