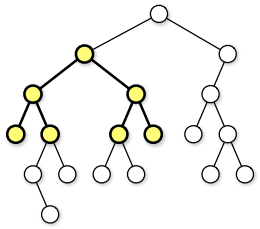
"CS 374" Fall 2014 — Homework 4

Due Tuesday, October 7, 2014 at noon

---

1. For this problem, a *subtree* of a binary tree means any connected subgraph. A binary tree is *complete* if every internal node has two children, and every leaf has exactly the same depth. Describe and analyze a recursive algorithm to compute the *largest complete subtree* of a given binary tree. Your algorithm should return both the root and the depth of this subtree.



The largest complete subtree of this binary tree has depth 2.

2. Consider the following cruel and unusual sorting algorithm.

```
CRUEL(A[1..n]):
  if n > 1
      CRUEL(A[1..n/2])
      CRUEL(A[n/2+1..n])
      UNUSUAL(A[1..n])
```

```
UNUSUAL(A[1..n]):
  if n = 2
      if A[1] > A[2]                          ⟨⟨the only comparison!⟩⟩
          swap A[1] ↔ A[2]
  else
      for i ← 1 to n/4                        ⟨⟨swap 2nd and 3rd quarters⟩⟩
          swap A[i + n/4] ↔ A[i + n/2]
      UNUSUAL(A[1..n/2])                      ⟨⟨recurse on left half⟩⟩
      UNUSUAL(A[n/2+1..n])                    ⟨⟨recurse on right half⟩⟩
      UNUSUAL(A[n/4+1..3n/4])                 ⟨⟨recurse on middle half⟩⟩
```

Notice that the comparisons performed by the algorithm do not depend at all on the values in the input array; such a sorting algorithm is called ***oblivious***. Assume for this problem that the input size n is always a power of 2.

   (a) Prove by induction that CRUEL correctly sorts any input array. *[Hint: Consider an array that contains $n/4$ 1s, $n/4$ 2s, $n/4$ 3s, and $n/4$ 4s. Why is this special case enough?]*

   (b) Prove that CRUEL would *not* correctly sort if we removed the for-loop from UNUSUAL.

   (c) Prove that CRUEL would *not* correctly sort if we swapped the last two lines of UNUSUAL.

   (d) What is the running time of UNUSUAL? Justify your answer.

   (e) What is the running time of CRUEL? Justify your answer.

3. In the early 20th century, a German mathematician developed a variant of the Towers of Hanoi game, which quickly became known in the American literature as "Liberty Towers".[1] In this variant, there is a row of $k \geq 3$ pegs, numbered in order from 1 to k. In a single turn, for any index i, you can move the smallest disk on peg i to either peg $i-1$ or peg $i+1$, subject to the usual restriction that you cannot place a bigger disk on a smaller disk. Your mission is to move a stack of n disks from peg 1 to peg k.

   (a) Describe and analyze a recursive algorithm for the case $k = 3$. ***Exactly*** how many moves does your algorithm perform?

   (b) Describe and analyze a recursive algorithm for the case $k = n+1$ that requires at most $O(n^3)$ moves. To simplify the algorithm, assume that n is a power of 2. *[Hint: Use part (a).]*

   (c) **[Extra credit]** Describe and analyze a recursive algorithm for the case $k = n+1$ that requires at most $O(n^2)$ moves. Do *not* assume that n is a power of 2. *[Hint: Don't use part (a).]*

   (d) **[Extra credit]** Describe and analyze a recursive algorithm for the case $k = \sqrt{n}+1$ that requires at most a polynomial number of moves. To simplify the algorithm, assume that n is a power of 4. What polynomial bound do you get? *[Hint: Use part (a)!]*

   ★(e) **[Extra extra credit]** Describe and analyze a recursive algorithm for arbitrary n and k. How small must k be (as a function of n) so that the number of moves is bounded by a polynomial in n? (This is actually an open research problem, a phrase which here means "Nobody knows the best answer.")

[1] No, not really. During World War I, many German-derived or Germany-related names were changed to more patriotic variants. For example, sauerkraut became "liberty cabbage", hamburgers became "liberty sandwiches", frankfurters became "Liberty sausages" or "hot dogs", German measles became "liberty measles", dachsunds became "liberty pups", German shepherds became "Alsatians", and pinochle (the card game) became "Liberty". For more recent anti-French examples, see "freedom fries", "freedom toast", and "liberty lip lock". Americans are weird.

CS 374 Fall 2014 — Homework 4 Problem 1

Name:	NetID:
Name:	NetID:
Name:	NetID:
Section: 1 2 3	

Describe and analyze a recursive algorithm to compute the *largest complete subtree* of a given binary tree. Your algorithm should return both the root and the depth of this subtree.

CS 374 Fall 2014 — Homework 4 Problem 2

Name:	NetID:
Name:	NetID:
Name:	NetID:
Section: 1 2 3	

(a) Prove by induction that CRUEL correctly sorts any input array.

(b) Prove that CRUEL would *not* correctly sort if we removed the for-loop from UNUSUAL.

(c) Prove that CRUEL would *not* correctly sort if we swapped the last two lines of UNUSUAL.

(d) What is the running time of UNUSUAL? Justify your answer.

(e) What is the running time of CRUEL? Justify your answer.

CS 374 Fall 2014 — Homework 4 Problem 3

Name:	NetID:
Name:	NetID:
Name:	NetID:
Section: 1 2 3	

(a) Describe and analyze a recursive algorithm to solve the Liberty Towers puzzle when $k = 3$. ***Exactly*** how many moves does your algorithm perform?

(b) Describe and analyze a recursive algorithm to solve the Liberty Towers puzzle in $O(n^3)$ moves when $k = n + 1$.

(c) **[Extra credit]** Describe and analyze a recursive algorithm to solve the Liberty Towers puzzle in $O(n^2)$ moves when $k = n + 1$.

(d) **[Extra credit]** Describe and analyze a recursive algorithm to solve the Liberty Towers puzzle in a polynomial number of moves when $k = n + 1$.

(e) **[Extra extra credit]** Describe and analyze a recursive algorithm to solve the Liberty Towers puzzle for arbitrary n and k. How small must k be (as a function of n) so that the number of moves is bounded by a polynomial in n?
