Dividing an array takes constant time.

1. (5 points) Let $T(n)$ be the running time of Jump. Give a recursive definition of $T(n)$.

2. (3 points) What is the height of the recursion tree for $T(n)$, assuming $n$ is a power of 3?

3. (3 points) What is amount of work (aka sum of the values in the nodes) at level $k$ of this tree?

4. (4 points) What is the big-Theta running time of Jump? Briefly justify your answer.
01 Swing(k,n) \ inputs are positive integers
02 if (n = 1) return k
03 else if (n = 2) return \(k^2\)
04 else
05 \ \ \ \ \ \ \ \ \ \ \ \ \ \ half = \lfloor n/2 \rfloor
06 \ \ \ \ \ \ \ \ \ \ answer = Swing(k,half)
07 \ \ \ \ \ \ \ \ \ \ answer = answer*answer
08 \ \ \ \ \ \ \ \ \ \ if (n is odd)
09 \ \ \ \ \ \ \ \ \ \ answer = answer*k
10 \ \ \ \ \ \ \ \ \ \ return answer

1. (5 points) Suppose \(T(n)\) is the running time of Swing. Give a recursive definition of \(T(n)\).

2. (4 points) What is the height of the recursion tree for \(T(n)\)? (Assume that \(n\) is a power of 2.)

3. (3 points) How many leaves are in the recursion tree for \(T(n)\)?

4. (3 points) What is the big-Theta running time of Swing?
01 Waltz\(a_1, a_2, \ldots, a_n\): list of real numbers
02 \quad \text{if } (n = 1) \text{ then return } 0
03 \quad \text{else if } (n = 2) \text{ then return } |a_1 - a_2|
04 \quad \text{else}
05 \quad \quad \text{L} = \text{Waltz}(a_2, a_3, \ldots, a_n)
06 \quad \quad \text{R} = \text{Waltz}(a_1, a_2, \ldots, a_{n-1})
07 \quad \quad \text{Q} = |a_1 - a_n|
08 \quad \text{return } \max(L, R, Q)

Removing the first element of a list takes constant time; removing the last element takes \(O(n)\) time.

1. (3 points) Give a succinct English description of what Waltz computes.

2. (4 points) Suppose \(T(n)\) is the running time of Waltz. Give a recursive definition of \(T(n)\).

3. (4 points) What is the height of the recursion tree for \(T(n)\)?

4. (4 points) How many leaves are in the recursion tree for \(T(n)\)?
01 Grind(\(a_1, \ldots, a_n\)) \(\backslash \) input is a sorted array of \(n\) integers
02 \quad \text{if } (n = 1) \text{ return } a_1
03 \quad \text{else}
04 \quad \quad m = \lfloor \frac{n}{2} \rfloor
05 \quad \quad \text{if } a_m > 0
06 \quad \quad \quad \text{return } \text{Grind}(a_1, \ldots, a_m) \quad \backslash \text{ constant time to extract part of array}
07 \quad \quad \text{else}
08 \quad \quad \quad \text{return } \text{Grind}(a_{m+1}, \ldots, a_n) \quad \backslash \text{ constant time to extract part of array}

1. (5 points) Suppose that \(T(n)\) is the running time of Grind on an input array of length \(n\) and assume that \(n\) is a power of 2. Give a recursive definition of \(T(n)\).

2. (4 points) What is the height of the recursion tree for \(T(n)\)?

3. (3 points) How many leaves does this tree have?

4. (3 points) What is the big-Theta running time of Grind?
1. (5 points) Suppose that $T(n)$ is the running time of Weave on an input array of length $n$. Give a recursive definition of $T(n)$.

2. (4 points) What is the height of the recursion tree for $T(n)$, assuming $n$ is a power of 2?

3. (3 points) What is the amount of work (aka sum of the values in the nodes) at level $k$ of this tree?

4. (3 points) How many leaves are in the recursion tree for $T(n)$? (Simplify your answer.)
1. (5 points) Suppose that $T(n)$ is the running time of Act on an input array of length $n$. Give a recursive definition of $T(n)$. Assume that dividing an array in half takes constant time.

2. (3 points) What is the height of the recursion tree for $T(n)$, assuming $n$ is a power of 2?

3. (3 points) What is the amount of work (aka sum of the values in the nodes) at level $k$ of this tree?

4. (4 points) What is the big-Theta running time of Act. Briefly justify your answer. Recall that $\sum_{k=0}^{n} a^k = \frac{a^{n+1} - 1}{a - 1}$. 

```java
01 Act(a_1, \ldots, a_n; b_1, \ldots, b_n) \quad \text{input is 2 arrays of } n \text{ integers, } n \text{ is a power of 2}
02 \quad \text{if } (n = 1)
03 \quad \quad \text{return } a_1 b_1
04 \quad \text{else}
05 \quad \quad p = \frac{n}{2}
06 \quad \quad \text{rv} = \text{Act}(a_1, \ldots, a_p, b_1, \ldots, b_p)
07 \quad \quad \text{rv} = \text{rv} + \text{Act}(a_1, \ldots, a_p, b_{p+1}, \ldots, b_n)
08 \quad \quad \text{rv} = \text{rv} + \text{Act}(a_{p+1}, \ldots, a_n, b_{p+1}, \ldots, b_n)
09 \quad \quad \text{rv} = \text{rv} + \text{Act}(a_{p+1}, \ldots, a_n, b_1, \ldots, b_p)
10 \quad \quad \text{return } \text{rv}
```
01 Dig (a_1, \ldots, a_n: array of integers)
02 \quad \text{if (} n = 1 \text{)}
03 \quad \text{if (} a_1 > 8 \text{)} \text{ return true}
04 \quad \text{else return false}
05 \quad \text{else if (} \text{Dig}(a_1, \ldots, a_{n-1}) \text{ is true and } \text{Dig}(a_2, \ldots, a_n) \text{ is true})
06 \quad \text{return true}
07 \quad \text{else return false}

1. (3 points) If Dig returns true, what must be true of the values in the input array?

2. (5 points) Give a recursive definition for \( T(n) \), the running time of Dig on an input of length \( n \), assuming it takes constant time to set up the recursive calls in line 05.

3. (3 points) What is the height of the recursion tree for \( T(n) \)?

4. (4 points) What is the big-theta running time of Dig?
01 Swim(a_1, \ldots, a_n) \quad \text{input is a sorted list of } n \text{ integers}
02 \quad \text{if } (n = 1) \ \text{return } a_1
03 \quad \text{else}
04 \quad \quad m = \left\lfloor \frac{n}{2} \right\rfloor
05 \quad \quad \text{if } a_m > 0
06 \quad \quad \quad \text{return Swim}(a_1, \ldots, a_m) \quad \text{O(n) time to extract half of list}
07 \quad \quad \text{else}
08 \quad \quad \quad \text{return Swim}(a_m+1, \ldots, a_n) \quad \text{O(n) time to extract half of list}

1. (5 points) Suppose that \( T(n) \) is the running time of Swim on an input list of length \( n \) and assume that \( n \) is a power of 2. Give a recursive definition of \( T(n) \).

2. (4 points) What is the height of the recursion tree for \( T(n) \)?

3. (3 points) What value is in each node at level \( k \) of this tree?

4. (3 points) What is the big-Theta running time of Swim?