CS 173, Spring 2016
Examlet 11, Part A

NETID:

FIRST: | **LAST:**

Discussion: Monday 9 10 11 12 1 2 3 4 5

```
DoIt(a_1, ..., a_n)  \\ input is an array of n integers
        if (n = 1) return a_1
        else
                m = ⌊n/2⌋
                p = DoIt(a_1, ..., a_m)   \\ constant time to extract part of array
                q = DoIt(a_{m+1}, ..., a_n)   \\ constant time to extract part of array
                return max(p,q)
```

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

   Solution:

   $T(1) = c$

   $T(n) = 2T(n/2) + d$

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

   Solution: $\log_2 n$

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

   Solution: $d \cdot 2^k$

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

   Solution: $\Theta(n)$

CS 173, Spring 2016
Examlet 11, Part A

NETID:

FIRST: **LAST:**

Discussion: **Monday** **9** **10** **11** **12** **1** **2** **3** **4** **5**

Act($a_1, \ldots, a_n$) \\ input is a linked list of n integers
    if ($n = 1$) return a_1
    else
        m = $\lfloor \frac{n}{2} \rfloor$
        p = Act($a_1, \ldots, a_m$) \\ O(n) time to split list
        q = Act($a_{m+1}, \ldots, a_n$) \\ O(n) time to split list
        return max(p,q)

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

   Solution:

   $T(1) = c$

   $T(n) = 2T(n/2) + dn + f$

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

   Solution: $\log_2 n$

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

   Solution: There are 2^k nodes, each containing $f + dn/2^k$. So the total is $2^k f + dn$

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

   Solution: $\Theta(n \log n)$

CS 173, Spring 2016
Examlet 11, Part A

NETID:

FIRST: **LAST:**

Discussion: Monday 9 10 11 12 1 2 3 4 5

```
Weave(a_1, ..., a_n)  \\ input is a sorted array of n integers
        if (n = 1) return a_1
        else
                m = ⌊n/2⌋
                if a_m > 0
                        return Weave(a_1, ..., a_m)   \\ constant time to extract part of array
                else
                        return Weave(a_{m+1}, ..., a_n)   \\ constant time to extract part of array
```

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

   Solution:

   $T(1) = c$

   $T(n) = T(n/2) + d$

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

   Solution: $\log_2 n$

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

   Solution: One.

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

   Solution: $\Theta(\log n)$

CS 173, Spring 2016 Examlet 11, Part A

NETID:

FIRST: **LAST:**

Discussion: **Monday** **9** **10** **11** **12** **1** **2** **3** **4** **5**

```
Grind(a_1, ..., a_n: an array of n positive integers, n ≥ 2)
  if (n = 1) return 0
  else if (n = 2) return a_1 + a_2
  else
     p = ⌊n/3⌋
     q = ⌊2n/3⌋
     rv = max(Grind(a_1, ..., a_p), Grind(a_{q+1}, ..., a_n))
     for i=p to q
        rv = max(rv, a_i + a_{i+1})
     return rv
```

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

   Solution:

   $T(2) = c$

   $T(n) = 2T(n/3) + dn + f$

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

   Solution: $\log_3(n)$

   [If n is a power of 3, it will hit the $n = 1$ base case and not the $n = 2$ base case.]

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

   Solution: $\frac{dn}{3^k}2^k + f2^k$

4. (4 points) How many leaves does this recursion tree have? Simplify so that your answer is easy to compare to standard running times. Recall that $\log_b x = \log_a x \log_b a$.

   Solution: $2^{\log_3 n} = n^{log_3 2}$

CS 173, Spring 2016
Examlet 11, Part A

NETID:

FIRST: **LAST:**

Discussion: Monday 9 10 11 12 1 2 3 4 5

```
Pump(a_1, a_2, ... a_n: list of real numbers)
  if (n = 1) then return 0
  else if (n = 2) then return |a_1 − a_2|
  else
    L = Pump(a_2,a_3,...,a_n)
    R = Pump(a_1,a_2,...,a_{n−1})
    Q = |a_1 − a_n|
    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 Pump computes.

   Solution: Pump computes the largest difference between two values in its input list.

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

   Solution: $T(1) = d_1$ $\quad T(2) = d_2$

   $T(n) = 2T(n-1) + cn + p$

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

   Solution: We hit the base case when $n-k=2$, where k is the level. So the tree has height $n-2$.

4. (4 points) How many leaves are in the recursion tree for $T(n)$?

   Solution: 2^{n-2}

CS 173, Spring 2016 Examlet 11, Part A

NETID:

FIRST: | **LAST:**

Discussion: Monday 9 10 11 12 1 2 3 4 5

```
WorkIt(a_1, ..., a_n: a list of n positive integers)
    if (n = 1) return a_1
    else if (n = 2) return max(a_1, a_2)
    else if (a_1 < a_n)
        return WorkIt(a_2, ..., a_n)
    else
        return WorkIt(a_1, ..., a_{n-1})
```

Max takes constant time. Removing the last element of a list takes $O(n)$ time.

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

   Solution: $T(1) = c$

   $T(2) = d$

   $T(n) = T(n-1) + pn$

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

   Solution: We hit the base case when $n - k = 2$, where k is the level. So the tree has height $n - 2$.

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

   Solution: Notice that the tree doesn't branch, so there is only one node at each level. So the total amount of work at level k is $p(n-k)$.

4. (4 points) What is the big-theta running time of WorkIt? Briefly justify and/or show your work?

   Solution:

   $\Theta(n^2)$

   [Much more detail than you needed to give:] Notice that the sum of all the non-leaf nodes is $\sum_{k=1}^{n-3} p(n-k)$. If we move the constant p out of the summation and substitute in the new index value $j = n - k$, we get

   $$p\sum_{j=3}^{n-1} j = p\sum_{j=1}^{n-1} j) - 3 = p\frac{(n-1)n}{2} - 3 = \frac{p}{2}n^2 - \frac{p}{2}n - 3$$

   The dominant term of this is proportional to n^2.