

Worth: 8%

Due: By 6pm on Monday 2 November

For each question, please write up detailed answers carefully. Make sure that you use notation and terminology correctly, and that you explain and justify what you are doing. Marks **will** be deducted for incorrect or ambiguous use of notation and terminology, and for making incorrect, unjustified, ambiguous, or vague claims in your solutions.

- [18] 1. Consider the following “ternary” search algorithm.

```

TERNARYSEARCH( $x, A, b, e$ ):
  if  $e == b$ :
    if  $x \leq A[b]$ :      return  $b$ 
    else:                return  $e + 1$ 
  else if  $e == b + 1$ :
    if  $x \leq A[b]$ :      return  $b$ 
    else if  $x \leq A[e]$ : return  $e$ 
    else:                return  $e + 1$ 
  else:
     $c = (2 * b + e - 2) / 3$  # integer division
     $d = (b + 2 * e - 1) / 3$  # integer division
    if  $x \leq A[c]$ :      return TERNARYSEARCH( $x, A, b, c$ )
    else if  $x \leq A[d]$ : return TERNARYSEARCH( $x, A, c + 1, d$ )
    else:                return TERNARYSEARCH( $x, A, d + 1, e$ )

```

Since $\log_3 n < \log_2 n$ for all $n > 1$, it would seem that this algorithm should be more efficient than binary search. However, the difference between $\log_3 n$ and $\log_2 n$ is only a constant factor.

In order to compare both algorithms, we need to analyze their running times more precisely.

For your reference, here is the binary search algorithm for this question.

```

BINARYSEARCH( $x, A, b, e$ ):
  if  $e == b$ :
    if  $x \leq A[b]$ :      return  $b$ 
    else:                return  $e + 1$ 
  else:
     $m = (b + e) / 2$     # integer division
    if  $x \leq A[m]$ :     return BINARYSEARCH( $x, A, b, m$ )
    else:                return BINARYSEARCH( $x, A, m + 1, e$ )

```

- [6] (a) Define a “basic operation” to be any comparison operator (“==”, “ \leq ”) or assignment operator (“=”)—all other operations and keywords are ignored.
- Let $T(n)$ denote the worst-case number of basic operations performed by TERNARYSEARCH on any input of size n , and let $B(n)$ denote the worst-case number of basic operations performed by BINARYSEARCH on any input of size n .
- Write detailed and exact recurrence relations for $T(n)$ and $B(n)$. Justify that your recurrences are correct, from the algorithms.
- [12] (b) Determine which algorithm performs more basic operations in the worst-case, for any input of size $n \geq n_0$ (**for some constant n_0 to be determined as part of your solution**), and write a detailed proof of your claim. Show all of your work.

- [16] 2. Suppose that you plan to buy a house and will need a mortgage of A dollars at a monthly interest rate I and a monthly payment P . Let A_n denote the amount you have left to pay after n months (in particular, $A_0 = A$). Assume that at the end of each month, you are first charged interest on all the money you owed during the month and then your payment is subtracted.
- [3] (a) Write a recurrence relation for A_n , the amount owing after n months, in terms of A_{n-1} , I , and P . Also include appropriate base case(s). Justify your recurrence briefly.
- [5] (b) Solve your recurrence relation so that you get a closed form formula for A_n in terms of A , n , I , and P . Show your work.
- [8] (c) Write a detailed proof that your formula in (b) is correct.

- [46] 3. In this question we will examine two algorithms that return the two points closest to each other, given a list of points in the plane. The first algorithm is recursive and the second algorithm is iterative. Assume that $\text{DISTANCE}(p_1, p_2)$ returns the Euclidean distance between any two points p_1 and p_2 and that p_∞ is a sentinel value with the property that $\text{DISTANCE}(p_\infty, p_\infty) = \infty$.

- [23] (a) Write a detailed proof of correctness for the following algorithm.

```

FIND_CLOSEST_PAIR_REC( $A, e$ ):
    # Precondition:  $A$  is a non-empty list of 2D points and  $0 \leq e < \text{len}(A)$ .
    # Postcondition: Returns a pair of points which are the two closest points in  $A[0 \dots e]$ .
    if  $e < 1$ : return ( $p_\infty, p_\infty$ )
    ( $p, q$ ) = FIND_CLOSEST_PAIR_REC( $A, e - 1$ )
     $min$  = DISTANCE( $p, q$ )
    for  $i = 0, \dots, e - 1$ :
        if DISTANCE( $A[e], A[i]$ ) <  $min$ :
             $min$  = DISTANCE( $A[e], A[i]$ )
             $p = A[e]$ 
             $q = A[i]$ 
    return ( $p, q$ )

```

- [23] (b) Write a detailed proof of correctness for the following algorithm.

```

FIND_CLOSEST_PAIR_ITER( $A$ ):
    # Precondition:  $A$  is a non-empty list of 2D points and  $\text{len}(A) > 1$ .
    # Postcondition: Returns a pair of points which are the two closest points in  $A$ .
     $min$  =  $\infty$ 
     $p = -1$ 
     $q = -1$ 
    for  $i = 0, \dots, \text{len}(A) - 1$ :
        for  $j = i + 1, \dots, \text{len}(A) - 1$ :
            if DISTANCE( $A[i], A[j]$ ) <  $min$ :
                 $min$  = DISTANCE( $A[i], A[j]$ )
                 $p = i$ 
                 $q = j$ 
    return ( $A[p], A[q]$ )

```