MAT2440 Practice Exam 2 solutions Halleck Spg 2019 Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Book and notes are prohibited except for a single sheet (back and front) with hand-written formulae/notes. Submit formula sheet with your exam for up to 5 extra points.
* You may write on test page. However, put all your work and answers into the blue book.
* No credit will be given for any answer that is not backed up with work.
* Actual exam will consist of 10 questions similar to the following questions:

1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | 0 | 1 | 2 | 3 | 4 |
| an | 0+2/1=2 | 1+2/2=2 | 8+2/3=26/3 | 27+2/4=55/2 (or 27.5) | 64+2/5 = 64.4 |



1, − 2, 4, −8, 16 (this is a geometric sequence, r = -2)

2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | 0 | 1 | 2 | 3 |
| an | 2 | 2(1)+2=4 | 2(2)+4=8 | 2(3)+8=14 |

3. Let *a* ≠ 0, *b*, and *c* be integers. Show that if *a*|*b* and *a*|*c*, then *a*|(*b* + *c*).

***b* = *ka* and *c* = *ma*, so *b*+*c* = *ka* + *ma* = (*k*+*m*)*a*. \ *a*|(*b* + *c*)**

4.

 

So 3452=(110101111100)2=(11201212)3=(13031)7

1. Convert the following number (ABBA)16 from hexadecimal to octal. Each character gets replaced by 2 characters according to the following chart:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| hex | 0 |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| octal | 0 |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|   | 0 |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

**So** (ABBA)16 = **(12 13 13 12)8**

1. Give Q estimates for each of these functions.
	1. *n* log*(n*2 + 1*)* + *n*2 log *n* **∈ Q(*n*2 log *n*) (consider first the power, the log is secondary)**
	2. *(n* log *n* + 1*)*2 + *(*log *n* + 1*)(n*2 + 1*)* **∈ Q(*n*2(log *n*)2) (powers are same for the 2 leading terms, but then 2nd power of log trumps just log)**
	3. $n^{2^{n}}+n^{n^{2}}$ **∈ Q(**$n^{2^{n}}$**)** **(compare the exponents 2*n* and *n*2, since the base of *n* is the same, exponential always trumps a power)**
2. Let *f*1*(x), f*2*(x)* and *g(x)* be functions from **R** to **R**+. Show that if *f*1*(x)* and *f*2*(x)* are both O*(g(x))*, then *f*1*(x)* + *f*2*(x)* is O*(g(x))*. This requires a formal proof (you need to find a witness pair for *f*1*(x)* + *f*2*(x)*)*.*

**Given witness pairs (k, C) and (ℓ, D), let m=max(k, ℓ), E=C+D, then (m,E) serves as a witness pair: ∀ x > m, *f*1*(x)* + *f*2*(x)* ≤ Cg(x) + Dg(x)=(C+D) g(x) = E g(x) ∴ *f*1*(x)* + *f*2*(x)* is O*(g(x))*.**

1. Suppose that each step counted takes 10 ns (nanoseconds) to complete. If the size of the input goes from *n* to 2*n*, express the increase in processing time as a power, multiplicative factor or additive term (specify which) if the complexity is

**a)** log2 *n:* **log2 2*n =* log2 *n +* log2 2*=*log2 *n +* 1 *so the increase in processing time is an extra 10ns***

**b)** *n:* ***processing time doubles.***

**c)** *n*2 : **(2*n*)2= 4 *n*2 so processing time quadruples.**

**d )** *n*3 *:* **(2*n*)3= 8 *n*3 so processing time increases by a factor of 8.**

**e)** 2*n*: **22*n* = (2*n*)2 so if the size of the input doubles, then the processing time gets squared.**

1. Determine the least number of comparisons (best-case performance)
	1. required to find the maximum of a sequence of *n* integers, using Algorithm 1 of Section 3.1.

**There are n-1 comparisons within loop plus the n for loop variable so 2n-1**

* 1. used to locate an element in a list of *n* terms with a linear search.

**If item is located in the first spot, then only the two comparisons are made within the first pass in the while condition. Another comparison is made outside loop so 3 in total.**

* 1. used to locate an element in a list of *n* terms using a binary search.

**There are élog2 nù passes in loop which has 2 comparisons (including for loop variable) plus 1 to exit loop and 1 outside loop. So 2(élog nù+1) in total**

1. Find the following sums:





1.

 If the set is uncountable, use a proof by contradiction.

**Countable: create an array with 4 infinite rows, those congruent to 1 mod 3 (pos and neg), then those congruent to 2 mod 3 (pos and neg)**

 **1, 4, 7, …**

 **-2, -5, -7, …**

 **2, 5, 7, …**

**-1, -4, -7, …**

**Visit elements in each column sequentially. This provides bijection with the natural #’s**



 **Uncountable: proof by contradiction, assume that such numbers can be listed:**

**Create a number *r* different from all numbers in the list by visiting each diagonal entry dii. If it is 0, then choose 1, if it is 1, then choose 0. Assign this digit to be the ith entry di of *r* = 0.d1d2d3…. *r* is a decimal which contains only 0’s and 1’s and is different by design from every number in our list, which is a contradiction. Hence, it is impossible to list out all such numbers and hence they form an uncountable set.**



 **Countable: Proof is very similar to part a). Create an array with 2 (infinite) rows.**

 (1,1), (1,2), (1,3), …

 (2,1), (2,2), (2,3), …

**Visit elements in each column sequentially. This provides bijection with the natural #’s**

12.

**procedure** *sum*(*a*1, *a*2, …., *a*n: integers)

 *sum* := *a*1

 **for** *i* := 2 to *n*

 sum := sum + *ai*

 return *sum* {*sum* is the sum of elements in the sequence}

13.

**procedure** *smallestDiff*(*a*1, *a*2, …., *a*n: integers) {we are assuming n>1)

 *diff* := *a*2 − *a*1

 **for** *i* := 3 to *n*

 new := *a*n − *a*n−1

 if new < diff, diff := new

 return *diff* {*diff* is the smallest difference between consecutive elements in the sequence}

14.

Hint: last line should be: Return {max, min}

where max is the largest and min is the smallest of the inputted sequence of integers.

**procedure** *maxmin*(*a*1, *a*2, …., *a*n: integers)

 *max* := *a*1, *min* := *a*1

 **for** *i* := 2 to *n*

 if *max* < *ai* then *max* := *ai*

 if *min* > *ai* then *min* := *ai*

 return (*max, min*) {(*max, min*)is ordered pair of largest, smallest elements}

1. Use bubble sort with input 3, 1, 5, 7, 4 showing lists obtained at each step

**See answer on right. Entries that are being compared are boxed and in bold. The # of comparisons (potential swaps) decreases for each pass (4, 3, 2, 1, 0) of the loop. The entries that no longer undergo comparisons are in red.**

1. Use insertion sort with input 3, 1, 5, 7, 4, showing as separate steps the comparisons, rotations and insertions.



1. Use the greedy algorithm to make change for 77, 43 and 24 cents
	1. using an 18 cent coin, dimes, nickels and pennies

**77/18 = 4 R 5 so 4 18¢ and 1 5¢**

**43/18 = 2 R 7 so 2 18¢, 1 5¢ and 2 1¢**

**24/18 = 1 R 6 so 1 18¢, 1 5¢ and 1 1¢**

* 1. using an 18 cent coin, dimes and pennies

**77/18 = 4 R 5 so 4 18¢ and 5 1¢ or 9 coins total**

**43/18 = 2 R 7 so 2 18¢, and 7 1¢ or 9 coins total**

**24/18 = 1 R 6 so 1 18¢ and 6 1¢ or 7 coins total**

* 1. The greedy algorithm does not always give optimal solutions

(optimal meaning that its output has the fewest possible coins).

Find the coin configuration(s) (either a or b) and example input(s) (77, 43 and 24) which has a more optimal solution.

**In b, if we take out 3 18¢ from 77 ¢, we get 23 ¢ left which 2 dimes & 3 pennies or 8 coins**

**If we take out no 18 ¢ from 43 ¢, then we get 4 dimes and 3 pennies or 7 coins**

**If we take out no 18 ¢ from 24 ¢, then we get 2 dimes and 4 pennies or 6 coins**

**In all 3 of these examples, the greedy algorithm failed to give us the optimal solution.**