Honors Computer Science I (COP 3502) Exam #1 Solution
Date: 1/25/2024

1) (a) (6 pts) Determine the following sum in terms of n:

$$\sum_{i=n+1}^{n^2} 2i$$

$$\sum_{i=n+1}^{n^2} 2i = \sum_{i=1}^{n^2} (2i) - \sum_{i=1}^{n} (2i)$$

$$= \frac{2n^2(n^2+1)}{2} - \frac{2n(n+1)}{2}$$

$$= n^2(n^2+1) - n(n+1)$$

$$= n^4 + n^2 - n^2 - n$$

$$= n^4 - n = n(n^3 - 1)$$

$$= n(n-1)(n^2+n+1)$$

Grading: 1 pt split sum, 2 pts formula to nsq, 1 pt formula to n, 2 pts algebra, any of the last three forms is acceptable.

(b) (4 pts) Write a short function that runs in $O(n^2)$ that takes in n as a parameter and returns the result of the summation above by actually adding each of the terms individually. The function prototype is below. (Your grade is based on the direct translation of the summation, not returning the correct answer.)

```
int f(int n) {

    int res = 0;                              // 1 pt
    for (int i=n+1; i<=n*n; i++)              // 2 pts
        res += (2*i);                         // 1 pt
    return res;                               // 0 pts
}
```

Note: take off 1 pt if return is missing, so basically only 2 pts awarded for for loop if return is also there.

2) (15 pts) Find an exact closed form solution for the following recurrence relation. Your answer should be of the form $T(n) = \frac{(an+b)3^n+c}{d}$, where a, b, c and d are all integers.

$$T(n) = T(n-1) + n3^{n-1}$$
$$T(0) = 0$$

Let's iterate the recurrence three times:

$T(n) = T(n-1) + n3^{n-1}$
$T(n) = T(n-2) + (n-1)3^{n-2} + n3^{n-1}$
$T(n) = T(n-3) + (n-2)3^{n-3} + (n-1)3^{n-2} + n3^{n-1}$

After k iterations we have:

$$T(n) = T(n-k) + \sum_{i=n-k+1}^{n} i\,3^{i-1}$$

Since we know T(0), plug in n – k = 0, or k = n:

$$T(n) = T(0) + \sum_{i=1}^{n} i\,3^{i-1} = \sum_{i=1}^{n} i\,3^{i-1}$$

Let S equal the sum. Then we have:

$S = 1(3^0) + 2(3^1) + 3(3^2) + \cdots + n(3^{n-1})$

Multiply through by 3 and subtract:

$3S = 1(3^1) + 2(3^2) + 3(3^3) + \cdots + (n-1)(3^{n-1}) + n(3^n)$

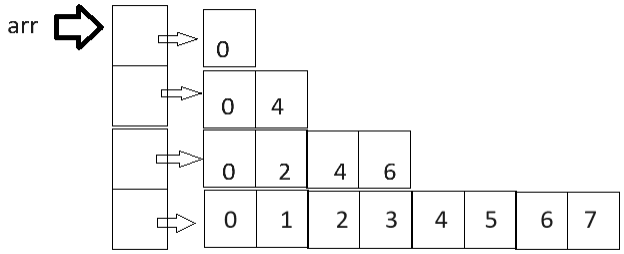
$S - 3S = -2S = 3^0 + 3^1 + \cdots + 3^{n-1} - n(3^n) = \frac{3^n-1}{2} - \frac{2n(3^n)}{2}$, divide by -2

$S = \frac{2n(3^n)-3^n+1}{4} = \frac{(\mathbf{2n-1})\mathbf{3^n+1}}{\mathbf{4}}$

The corresponding integers to fit the form given are a = 2, b = -1, c = 1 and d = 4.

Grading: 1 pt for each iteration, 2 pts for after k iterations,
2 pts for summation form of T(n)
2 pts for mult S by 3
2 pts for subtract
2 pts to plug in formula for geo sum
2 pts finishing up

4) (18 pts) A power array of size n is a 2D array that contains n+1 arrays, where array i is of size 2^i. Each of these n+1 arrays stores values equally spaced out, starting at 0. The common difference of each array is half of the previous array with the last array having a common difference of 1. Here is a picture of the power array of size n = 3



Write a function that takes in a positive integer n, allocates the correct amount of memory for the array, fills it, and returns a pointer to the array of arrays. Do NOT use the pow function. Use only integers in your calculation, multiplying ints in a loop as needed. (Hint: you should keep track of two counters. One counter initializes at 1, the other at 2^n. After each outer loop iteration, the first counter will double and the second will divide by 2. These two numbers store all you need to fill up a row of the table.

```
int** powerarray(int n) {

    int** res = calloc(n+1, sizeof(int*));          // 3 pts

    int skip = 1;                                   // 3 pts
    for (int i=0; i<n; i++)
        skip *= 2;

    for (int i=0,size=1; i<=n; i++,size*=2) {       // 4 pts

        res[i] = calloc(size, sizeof(int));         // 2 pts
        for (int j=1; j<size; j++)                  // 1 pts
            res[i][j] = res[i][j-1] + skip;         // 3 pts

        skip /=2;                                   // 1 pt
    }

    return res;                                     // 1 pt
}
```

Grading Note: Map points as shown above for completing tasks. Note that most students are likely to set res[i][0] = 0 instead of relying on the loop. So, the j loop is really 1 pt and setting index 0 in the row to 0 is 1 pt.

5) (6 pts) A program that solves a traveling salesman problem on n locations implements an algorithm with a run-time of $O(n2^n)$. For n = 16, the program takes 75 ms to complete. How long, **in seconds,** would we expect the algorithm to take when run on an input with size n = 20?

Let $T(n) = cn2^n$ be the run time of the program for input size c, for some constant c.

$$T(16) = c(16)(2^{16}) = 75ms \rightarrow c = \frac{75\ ms}{16(2^{16})}$$

$$T(20) = \left(\frac{75\ ms}{16(2^{16})}\right) \times 20 \times 2^{20} = (75ms) \times \frac{5}{4} \times 2^4 = 1500ms = \mathbf{1.5\ seconds}$$

1.5 sec

Grading: 1 pt set eqn with const, 1 pt plug in 16, 1 pt solve for c, 1 pt plug in 20, 1 pt get to 1500 ms, 1 to convert to seconds.

6) (6 pts) What is the run-time of the function below in terms of the input variables r and c? Please give your answer in Big-Oh format. Please justify your answer for full credit.

```
int f(int** a, int r, int c) {

    int s = 0;
    for (int i=0; i<r; i++) {
        int x = 0, y = c-1;
        while (x < y) {
            int t = rand()%2;
            int m = (x+y)/2 ;
            if (t == 0)
                x = m+1 ;
            else
                y = m-1 ;
        }
        s += a[i][x] ;
    }
    return s ;
}
```

Outer loop runs r times.

y-x starts at c-1
divides by 2 each time
just like binary search
so while loop runs O(lg c)
times for each value of i.
Multiply to get O(rlg c)

O(rlg c) , Grading 1 pt for outer loop reason, 2 pts for inner loop reason, 3 pts for answer (1 pt r, 1 pt lg c, 1 pt mult if use n no credit at all)

7) (17 pts) The union of two sets is the set of all items that belong in either of the two sets. One way to store a set is in an array in sorted order. Write a function that runs in O(n+m) time that takes in one array of size n and another array of size m and creates a new array of the appropriate size that stores the union of the two input sets and returns a pointer to the newly created array. For example, if the input arrays were [2, 3, 6, 12] and [1, 2, 6, 8, 12, 14] the array created and returned would be [1, 2, 3, 6, 8, 12, 14]. (Hint: At first you won't know the exact size, but you can allocate "plenty of room", then before returning, resize the array to the right size.)

```
int* setunion(int* set1, int n, int* set2, int m) {

    int* res = calloc(n+m, sizeof(int));                    // 3 pts

    int i = 0, j = 0, k = 0;                                // 1 pt
    while (i<n || j<m) {                                    // 1 pt

        if (j==m || (i<n && set1[i]<set2[j]))               // 2 pts
            res[k++] = set1[i++];                           // 1 pts
        else if (i==n || (j<m && set2[j]<set1[i]))          // 2 pts
            res[k++] = set2[j++];                           // 1 pt
        else {                                              // 3 pts
            res[k++] = set1[i++];
            j++;
        }
    }

    res = realloc(res, k*sizeof(int));                      // 2 pts
    return res;                                             // 1 pt

}
```

8) (3 pts) The C programming language was built off of its predecessor, which just happens to be the same as the predecessor to the letter C in the English alphabet. What was the name of the language that the C programming language was built off of?

B (Give to All)