2016 Spring COP 3502 Final Exam Solutions

Date: 4/28/2016

1) (10 pts) Consider implementing a hash table that stores integers, using the linear probing strategy. Assume that the hash table uses the hash function, f, defined below and that size is 23. Show the contents of the table after the following insertions have been made, in the order given: 4362, 999235, 7283624, 8123456, 77, 11111111, 52, 123, 7999999 and 12345675.

```
int f(int n, int size) {
    int res = 0;
    while (n > 0) {
        res = (res + (n%10))%size;
        n = n/10;
    }
    return res;
}
```

index	0	1	2	3	4	5	6	7	8	9	10	11
value							**8123456**	**52**	**11111111**	**7283624**	**123**	**12345675**

index	12	13	14	15	16	17	18	19	20	21	22
value			**999235**	**4362**	**77**	**7999999**					

Note: The key here is to realize that the function just returns the sum of the digits mod size. If you don't realize that, this question takes 20 minutes instead of 3 and it takes away time from the rest of the exam. This was done by design, to reward the students who were able to look at the code and translate what it does into English as opposed to a literal translation line by line.

Grading: 1 pt per number. To get the point, the correct number must be in the correct slot.

2) (10 pts) Consider inserting the following items into a minimum heap in the following order: 18, 8, 16, 4, 3, 12, 17, 2, 22, and 7. Show the state of the heap (drawn as a complete binary tree) after the completion of each insertion. Draw a box around each of your answers.

Solution

1st insert:

18

2nd insert:

8 (perc up 1)

/

18

3rd insert:

8

**/ **

18 16

4th insert:

4 (perc up twice)

**/ **

8 16

/

18

5th insert:

3 (perc up twice)

**/ **

4 16

**/ **

18 8

6th insert:

3

**/ **

4 12 (perc up 1)

/ \ /

18 8 16

7th insert:

3

**/ **

4 12

**/ \ / **

18 8 16 17

8th insert:

2 (perc up 3 times)

**/ **

3 12

**/ \ / **

4 8 16 17

/

18

9th insert:

2

**/ **

3 12

**/ \ / **

4 8 16 17

**/ **

18 22

10th insert:

2

**/ **

3 12

**/ \ / **

4 7 16 17

/ \ /

18 22 8

Grading: 1 pt per tree, give point for a tree if the individual insert is correct based on previous insert.

3) (12 pts) It was mentioned in class that a Bubble Sort could be implemented recursively, since after one iteration of a Bubble Sort on n elements, the maximum value is correctly sorted, so what remains is an array of size n-1. Fill in the function shown below so that it implements a recursive Bubble Sort. You may (and should) call the swap function that is provided below.

```
void swap(int* ptrA, int* ptrB) {
    int temp = *ptrA;
    *ptrA = *ptrB;
    *ptrB = temp;
}

void bubbleSortRec(int* array, int n) {

    if ( n<=1 ) return;                          // 1 pt n==1 ok

    int i;
    for (i = 0 ; i < n-1 ; i++) {                // 1 pt (can Δ)

        if (array[i] > array[i+1])               // 3 pts (can Δ)

            swap(&array[i], &array[i+1]);        // 4 pts (can Δ)
    }

    bubbleSortRec(array, n-1);                   // 3 pts
}
```

4) (8 pts) Complete the function below so that takes in an integer n and returns the number of bits set to 1 in the binary representation of n. In order to get full credit, please use bitwise operators.

```
int bitCount(int n) {

    int i, res = 0;

    for (i=0; (1<<i) <= n ; i++)          // 4 pts (can Δ)

        if ( (n & (1<<i)) != 0)           // 4 pts (can Δ)
            res++;

    return res;
}
```

5) (15 pts) A new company, iTraits, has designed a personality test so that people can identify others like them and unlike them. They use a 20 question survey of yes/no questions to characterize each person in their database. Each person's personality can be stored efficiently in a computer program as an integer, where the i^{th} bit is set to 0 if the answer to question i is no, and the i^{th} bit is set to 1 if the answer to question i is yes. (The questions are numbered 0 to 19.) For example, if a person's answers to questions 0 through 3 were yes and questions 4 through 19 were 0, the value $15 = 2^0 + 2^1 + 2^2 + 2^3$ would be stored for their personality.

We define someone's arch-nemesis as the person with whom they differ on the most number of traits. Complete the function below so that it takes in the value corresponding to a particular individual's personality (personA), the array of values (otherPeople) representing personalities of each other person in a database, and the length of that array (length), and returns the index into otherPeople of the arch-nemesis of personA. If there are multiple arch-nemesi, you may return the index of any one of them. You may call the bitCount function from question 4 and assume that it works properly. In order to get full credit, please use bitwise operators.

```
int getArchNemisis(int personA, int* otherPeople, int length) {

    int i, res = -1, diff = 0;

    for (i=0; i<length; i++) {

        if (bitCount(personA^otherPeople[i]) > diff) {

            diff = bitCount(personA^otherPeople[i]);

            res = i;
        }
    }

    return res;
}
```

Grading: Many ways to write this. Here is criteria -

1 pt for variable declarations and initializations
3 pts for length loop (this is necessary)
4 pts for appropriate xor or alternative operation
1 pt for calling bitcount
3 pts for seeing if this choice is the best so far
2 pts for updating our result if necessary
1 pt for returning

6) (15 pts) Consider a binary search tree of integers where each node stores the number of nodes in its subtree. The struct for such a tree is as follows:

```
struct bintreenode {
    int data;
    int numNodes;
    struct bintreenode* left;
    struct bintreenode* right;
};
```

Write a function that takes in a pointer to a `struct bintreenode root`, and an integer `k`, and returns the k^{th} smallest integer in the tree rooted at root. (Hint: make your function recursive. If you look at how many values are in the left subtree, it will tell you whether or not the k^{th} smallest value is in the left subtree, the right subtree or the root.)

```
// Pre-conditions: root != NULL and 1 <= k <= root->numNodes.
int findKthSmallest(struct bintreenode* root, int k) {


    int leftSide = 0;                    // 3 pts assign leftSide
    if (root->left != NULL)              // -2 if didn't catch
        leftSide = root->left->numNodes;// NULL case.

    if (k <= leftSide)                        // 4 pts this case
        return findKthSmallest(root->left, k);

    else if (k == leftSide+1)                 // 4 pts this case
        return root->data;
    else                                      // 4 pts this case
        return findKthSmallest(root->right, k-leftSide-1);
}
```

7) (12 pts) Using the iteration technique, find a Big-Oh bound for the following recurrence relation, in terms of n:

$$T(n) = 2T\left(\frac{n}{2}\right) + n^2, \text{ for n > 1}$$
$$T(1) = 1$$

Solution

$$T(n) = 2T\left(\frac{n}{2}\right) + n^2$$
$$T(n) = 2[2T\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right)^2] + n^2$$
$$T(n) = 4T\left(\frac{n}{4}\right) + 2 \times \frac{n^2}{4} + n^2$$
$$T(n) = 4T\left(\frac{n}{4}\right) + \frac{3n^2}{2}$$
$$T(n) = 4[2T\left(\frac{n}{8}\right) + (\frac{n}{4})^2] + \frac{3n^2}{2}$$
$$T(n) = 8T\left(\frac{n}{8}\right) + 4 \times \frac{n^2}{16} + \frac{3n^2}{2}$$
$$T(n) = 8T\left(\frac{n}{8}\right) + \frac{7n^2}{4}$$

In general, after k steps, we see that our formula will iterate to:

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + \frac{(2^k - 1)n^2}{2^{k-1}}$$

Plug in $\frac{n}{2^k} = 1$, so let $n = 2^k$ and $k = log_2 n$ to obtain:

$$T(n) = nT(1) + \frac{(n-1)n^2}{n/2}$$
$$T(n) = n + 2n(n-1)$$
$$T(n) = 2n^2 - n = O(n^2)$$

Grading: 2 pts for step 2 of the recurrence (4T(n/4) + 3/2 n^2), 2 pts for step 3 of the recurrence (8T(n/8) + 7/4 n^2), 3 pts for the guess, 3 pts for finding what to plug in for the guess, 2 pts for the final Big-Oh answer after plugging in.

8) (7 pts) An $O(n^2)$ algorithm takes half a second to run on an input of size 10,000. Roughly, how long will it take to run on an input of size 50,000, in seconds?

Solution

Let T(n) be the run time of the algorithm. Then $T(n) = cn^2$. It follows that:

$$\boldsymbol{T(10000) = c(10000)^2 = .5\ sec}$$
$$\boldsymbol{c = .5 \times 10^{-8} sec}$$

$$\boldsymbol{T(50000) = c(50000)^2 = .5 \times \frac{25 \times 10^8}{10^8} = 12.5\ seconds}$$

12.5 seconds

Grading: 3 pts solving for c, 2 pts to substitute c and 50,000. 2 pts to simplify answer in seconds.

9) (7 pts) What is the base 10 value of 5789 equal when converted to base 4?

Solution

4 | 5789
4 | 1447 R 1
4 | 361 R 3
4 | 90 R 1
4 | 22 R 2
4 | 5 R 2
4 | 1 R 1
4 | 0 R 1

1122131_4 (Grading 1 pt per digit going backwards, max 2 pts for any other algorithm)

10) (4 pts) What programming language is named in honor of mathematician Blaise Pascal?

Pascal (4 pts give to all)