University of Nevada, Las Vegas Computer Science 477/677 Spring 2024

Assignment 4: Due Saturday March 2 2024 11:59 PM

Upload your homework to Canvas.

Name:_____

You are permitted to work in groups, get help from others, read books, and use the internet. Your answers must be written in a pdf file and uploaded to canvas, by midnight February 16th. Your file must not be unnecessarily long. If you have any questions, or you are having trouble uploading the assignment you may email the grader, Sebrina Wallace, at `wallace4@unlv.nevada.edu`. You may also send me email to ask questions.

1. Write the asymptotic time complexity for each code fragment, using Θ notation.

   (a)
   ```
   for (int i=1; i < n; i++)
       for (int j=i; j > 0; j--)
   ```

   (b)
   ```
   for (int i=1; i < n; i=2*i)
       for (int j=i; j < n; j++)
   ```

   (c)
   ```
   for (int i=1; i < n; i++)
       for (int j=1; j < i; j = j*2)
   ```

   (d)
   ```
   for (int i=1; i < n; i++)
       for (int j=i; j < n; j = j*2)
   ```

   (e)
   ```
   for (int i=2; i < n; i = i*i)
   ```

   (f)
   ```
   for (int i=1; i*i < n; i++)
   ```

2. Give an asymptotic solution to each of these recurrences, using the Bentley-Blostein-Saxe method, otherwise known as the master theorem. Some of them may require substitution.

   (a) $F(n) = 2F(n/2) + n$

   (b) $F(n) = 4F(n/2) + n^3$

   (c) $F(n) = 4F(n/2) + n^2$

   (d) $F(n) = 4F(n/2) + n$

   (e) $T(n) = 7T(n/7) + n$

(f) $T(n) = 9T(n/3) + n^2$

(g) $T(n) = 8T(n/2) + n^3$

(h) $T(n) = T(\sqrt{n}) + 1$ Use substitution: $m = \log n$.

(i) $T(n) = 2T(n-1) + 1$ Use substitution: $n = \log m$, *i.e.* $m = 2^n$.

3. Give an asymptotic solution to each of these recurrences using the Akra-Brazzi method, otherwise known as the generalized master theorem.

   (a) $F(n) = 2F(n/4) + F(n/2) + 1$

   (b) $F(n) = 2F(n/4) + F(n/2) + n$

   (c) $F(n) = 2F(n/4) + F(n/2) + n^2$

   (d) $F(n) = F(3n/5) + F(4n/5) + n^2$

   (e) $F(n) = F(n/3) + 4F(2n/3) + 1$

4. Give an asymptotic solution to each these recurrences, using the anti-derivative method.

   (a) $F(n) = F(n - \log n) + \log n$

   (b) $G(n) = G(n-1) + n^c$ where $c \geq 1$ is a constant.

   (c) $K(n) = K(n - \sqrt{n}) + n$

5. What is the asymptotic complexity of the function `martha(x)` given below, in terms of x? Write a

recurrence and solve. (I mean the actual value of `martha(x)`, not the time to compute it.)

```
float martha(float x)
 {
  assert(x > 0);
  if(x < 1) return 0;
  else return 2*martha(x/2) + x;
 }
```

6. What is the asymptotic time complexity of the above code, which computes `martha(x)`? Assume that each operation takes constant time.

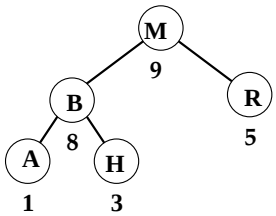
7. What is the asymptotic complexity of the function `george(x)` given below, in terms of x? Write a recurrence and solve. (I mean the actual value of `george(x)`, not the time to compute it.)

```
float george(float x)
 {
  assert(x > 0);
  if(x < 1) return 0;
  else return george(3*x/5) + george(4*x/5) + x*x;
 }
```

8. What is the asymptotic time complexity of the above code, which computes `george(x)`? Assume that each operation takes constant time.

9. Starting with an empty AVL tree, insert the letters Z,S,M,K,Q,N in that order. Show the tree after each insertion and each rotation.

10. Show the steps of deletion of M from the treap given below.



11. The following is C++ code for a function that you are familiar with. I have verified that the code compiles

and runs and is correct. The value of `guess(x,b)` computed by the program is rounded to six significant figures. That value could be positive, negative or zero. The parameter M is not mathematically related to the function, but is only a programming device to contain the recursion, since otherwise the recursive depth could theoretically be infinite.

```
const int N = 30; // limits the depth of the recursion

float guess(float x, float b, int M)
 {
  assert(x > 0.0 and b > 1.0);
  if(x == 1) return 0;
  else if(M <= 0) return 0; // Really?  Shouldn't it be 1?  Does it matter?
  else if(x < 1) return -guess(1/x,b,M);
  else if(x > b) return 1 + guess(x/b,b,M);
  else if(x < b) return guess(x*x,b,M-1)/2;
  else return 1.0;
 }

float guess(float x, float b)
 {
  cout << "Computing guess(" << x << "," << b << ")" << endl;
  return guess(x,b,N);
 }

int main()
 {
  float x;
  cout << "Enter a positive number x:  ";
  cin >> x;
  cout << endl;
  float b;
  cout << "Enter a number larger than 1:  ";
  cin >> b;
  cout << endl;
  cout << guess(x,b) << endl;
  return 1;
 }
```

What does the function guess(x,b) compute? The code is available on the Assignments page, and can be downloaded into a text file and run. Suggested cases to try: `guess(10,3)` and `guess(0.03125,4)`

12. Fill in the blanks.

(a) True or false: Open hashing uses open addressing. _____.

(b) When two data have the same hash value, that is called a _____.

(c) A _____ hash function gives a 1-1 correspondence between the data and the indices of the hash table.

(d) In closed hashing, if a collision occurs, one of the data uses a _____ sequence to search for an unused index.

(e) In open hashing, the data which share a hash value must be stored in a _____. (Choose one of these answers: **search structure**, **priority queue**, **virtual array**, **directed graph**.)

(f) An optimal binary prefix code for a given weighted alphabet can be computed using _____ algorithm.

(g) In an unweighted directed graph, the shortest path between two given vertices can be found by _____-first search. (Choose one of these answers: **Depth**, **Breadth**.)

(h) Binary search tree sort (or simply *tree sort*) is a fast implementation of _____ sort. (Choose of these answers: **selection**, **bubble**, **insertion**, **quick**.)

(i) A _____ order of a directed graph G is an ordering of the vertices of G such that vertex x must be come earlier than vertex y in the ordering if there is an arc from x to y,

(j) The subproblems of a dynamic program must be worked in _____ order.