**CSC 3110**

**Algorithm Design and Analysis**

(30 points)

Due: 04/01/2024 11:59 PM﻿﻿﻿

**Note:** **Submit answers in PDF document format. Please read the submission format for appropriate file naming conventions.**

1. Exercises 7.1:
   1. Problem 4 (2 points)   
        
      Yes, distribution counting algorithm is stable. It is stable because A stable algorithm preserves key order even when the keys are equal. The distribution counting algorithm moves from right to left, and puts equal elements into a sorted array that is also built right to left – meaning that elements remain in the same
   2. A close up of a text

      Description automatically generated

Preprocessing the information allows the graph to run faster. The first thing to do is calculate the lowest common ancestor of two vertices using the parent information. The lowest common ancestor is the first ancestor of u that is also an ancestor of v. This involves finding all ancestors of v, then starting with u and going through its ancestors and finding the first shared one. If u is the LCA of v that means u is the ancestor of v. This solution is linear in preprocessing data because each node only needs to be considered once, as LCA can be done concurrently for multiple nodes since they are all part of the same tree. Once the LCA information is processed for each node, it can then be run in constant time to check if the LCA is shared or not.

1. Exercises 7.2:
   1. Problem 2 (part a & b) (2 points)   
      A screenshot of a computer

      Description automatically generated  
      a.)

|  |  |  |  |
| --- | --- | --- | --- |
| A | C | G | T |
| 5 | 2 | 10 | 1 |

* 1. A grid with black lines

     Description automatically generated with medium confidence

Wow look at that excel spreadsheet screenshot, zoom in really far, it was captured on a 4K screen.

* 1. Problem 8 (part a & b) (2 points)  
     A white background with black text

     Description automatically generated  
     A.) Yes, Boyer-Moore with just the bad-symbol table will work because it’s just Horspool’s algorithm.  
     B.) No, Boyer Moore needs the bad-symbol table to work, otherwise the algorithm cannot identify characters that do not match.

1. Exercises 7.3:
   1. Problem 1 (parts a – c) (2 points)   
      A white background with black text

      Description automatically generated  
      a.)Step 1: Apply the hash function h(K) = k mod 11

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 30 | 20 | 56 | 75 | 31 | 19 |
| 8 | 9 | 1 | 9 | 9 | 8 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  | 56 |  |  |  |  |  |  | 30 | 20 |
|  |  |  |  |  |  |  |  | 19 | 75 |
|  |  |  |  |  |  |  |  |  | 31 |

Look at that clustering. Much wow.  
  
b.)Because there is a linked list of 3, the largest search will require traversing the entire linked list, so the largest number of key comparison for search is 3.  
c.)Ugg averages. 1/6 + 2/6 + 2/6 + 1/6 + 2/6 + 3/6 = 1.6666666666667

* 1. Problem 5 (2 points)

A black text on a white background

Description automatically generated  
This is known sometimes as the “Birthday Attack” problem. The answer is 22.5. The implication for hashing is that even when the size of a hash is seemingly massive, there still tends to be clustering from collisions even in small data sizes. Collisions are unavoidable – you just have to live with them and minimize them.

1. Exercises 7.4:
   1. Problem 3 (2 points)   
      A black text on a white background

      Description automatically generated  
      We need a B tree height that is 3 to reduce the number of searches to 3. Which means we need the m value for that B tree with height 3 and nodes = 100,000,000. B tree height is calculating using the following equation log(m/2)((n+1)/4) = log(m/2)(( 100,000,001/4)) = log(m/2)( 25000000.25) = 3

(m/2)^3 = 25000000.25

m/2 = 292.41

m = 584.82 rounded up to 585 because we can’t have a fraction of a child.

* 1. Problem 4 (2 points)  
       
     A diagram of a number

     Description automatically generated  
     A diagram of a diagram

     Description automatically generated

1. Exercises 8.1
   1. Problem 1 (1 point)  
      A black text on a white background

      Description automatically generated  
      Dynamic programming and divide and conquer both involve breaking down the larger problem into smaller subproblems. Dynamic programming however is built around the subproblems overlapping and reusing their results (often through use of a recursive function), divide and conquer involves there being no overlap between subproblems at all.
   2. Problem 5 (2 points)  
      A screenshot of a game

      Description automatically generated  
      If the cell is admissible, the function does not consider it. It is as simple as that. A screenshot of a game

      Description automatically generatedA blue line drawn on a grid

      Description automatically generated

There are at least 14 optimal paths.

1. Exercises 8.2
   1. Problem 4 (part a & b) (2 points)  
      Black text on a white background

      Description automatically generated  
      a.) True. The way that the dynamic programming solution is built for the knapsack problem means that it is based on the maximum of the values to the cells to the left and above it, as a result the values always increase.  
        
      b.)Also True, because the value of the knapsack problem is based on adjacent cells, this means that the knapsack is always increasing in value when descending as well.
   2. Problem 5 (2 points)  
      A close up of black text

      Description automatically generated  
      a.) Divide each of the items value by their weight

b.) Pick the item that has the maximum value per weight

c.) Put as many of that item inside of the knapsack, when no more can be placed, try whichever has the next highest value per weight, see if it fits, repeat until all items are considered

This algorithm will usually generate the optimum value to place in the knapsack, and more importantly it does so incredibly quickly, at a time efficiency of O(i), this linear efficiency is worth the lack of optimality, as in most situations uniformity of good loaded into the knapsack outweighs the benefit of potential additional value.

1. Exercises 8.3
   1. Problem 3 (2 points)  
      

Optimal binary search tree is kinda vague, optimal in terms of search? Here is a formula that will build

The optimal binary search tree is built by placing the highest frequency searches at the highest points of the tree, this guarantees that tree is searched through as quickly as possible.

* 1. Problem 9 (1 point)

A close up of a text

Description automatically generated

1. Exercises 8.4
   1. Problem 3 (2 points)  
      A black text on a white background

      Description automatically generated  
      There is a way to update Warshall’s algorithm by doing the calculations in place instead of writing temporary matrices. This change involves altering the lowest subloop of Warshall’s algorithm to consider if the path through the highest iterative value loop is shorter than the existing point in the graph. If it is, then it is replaced. Because the highest loop passes through every vertex, this means that every additional path is considered, and only the shortest path is kept.
   2. Problem 7 (2 points)

A number grid with black text

Description automatically generated with medium confidence  
Why use a fancy algorithm when you can just draw it by hand and look really hard? Is this fancy and scalable? NO IT IS NOT. Is it the faster than writing down matrix after matrix? YES IT IS. Considering I am more concerned about time efficiency than scalability, this is the best solution.   
A diagram of arrows and numbers

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 2 | 3 | 1 | 4 |
| 6 | 0 | 3 | 2 | 5 |
| 10 | 12 | 0 | 4 | 7 |
| 6 | 8 | 2 | 0 | 3 |
| 3 | 5 | 6 | 4 | 0 |