Homework 04 (Due: Friday, April 17, 2020, 11 : 59 : 00pm (Central Time))

CSCE 310

Instructions

This assignment consists of 5 analytical problems and 2 programming problems. Your solutions to the analytical problems must be submitted, as one PDF, via webhandin. While handwritten (then scanned) solutions to the analytical problems are acceptable, you are strongly encouraged to typeset your solutions in LaTeX or a word processor with an equation editor. The legibility of your solutions is of great importance.

Programming Assignment

Your methods will be tested on the `cse.unl.edu` server, using
`gcc version 4.8.5 (SUSE Linux))`. To ensure proper execution, you should test your submission in the webgrader

You will submit
`csce310homework04part01.h`, `csce310homework04part02.h`,
`csce310homework04part01.cpp`, `csce310homework04part02.cpp`
(and, maybe, `csce310homework04part03.h` and `csce310homework04part03.cpp`), along with your PDF, via webhandin.

`tallestTower`

`tallestTower` is a function that, given a vector of n block weights and a vector of n block carrying capacities, will return the greatest number of blocks that can be stacked on top of each other. It can be assumed that a block's carrying capacity will at least be equal to its weight. (The block can support its own weight.)

`tugOfWar`

`tugOfWar` is a function that, given a vector of weights for n individuals, will return the smallest difference in weight possible when dividing those individuals into two (2) teams. Note: Teams do not need an equal number of individuals.

`footRace` (15 Points Extra Credit)

A race is about to be run. Two (2) evenly matched runners take their marks. However, the course is not flat. There are n distinct segments to the course, each slightly uphill or downhill. There are even variations betweeen the two (2) lanes. Each runner must complete the first segment in their assigned lane, but they are free to change lanes (as many times as they want) afterwards. However,

changing lanes comes with a small penalty (different for each transition area between segments). There is no penalty for staying in the same lane from Segment i to Segment $i+1$. The first runner to arrive at the course is given a choice of starting lane. That runner would like to choose the lane that minimizes their time to complete the race.

`footRace` is a function that, given two (2) vectors of n values each (seconds required to complete each of n segments of a track with two lanes), and two (2) vectors of $n-1$ values each (seconds required to change from Lane 1 to Lane 2, or Lane 2 to Lane 1, after completing Segment s), will return the shortest time required to complete the race.

General Guidelines

Sample header, source, and testing files have been provided. You may modify the `.h` and `.cpp` files as needed, but you will only be turning in the four/six files mentioned above. The webgrader will be compiling the code with the command `g++ -o /path/to/executable.out /path/to/source/files/*.cpp` for each part, but I will only be copying `csce310homework04part01.h`, `csce310homework04part02.h`, `csce310homework04part01.cpp`, and `csce310homework04part02.cpp` out of your submission and into separate directories for Part 1 and Part 2.

Written Assignment

Question 1 (10 points)

From *Introduction to Algorithms* ISBN $978-0-262-03293-3$

Draw the recursion tree for the `Merge-Sort` procedure presented below on an array of 16 elements. Explain why memoization is ineffective in speeding up a good divide-and-conquer

algorithm such as `Merge-Sort`.

```
if p < r then
    q ← ⌊(p + r) /2⌋;
    Merge-Sort(A,p,q);
    Merge-Sort(A,q + 1,r);
    Merge(A,p,q,r);
end
```

Algorithm 1: Merge-Sort(A,p,r)

```
n_1 ← q − p + 1;
n_2 ← r − q;
create arrays L[1..n_1 + 1] and R[1..n_2 + 1];
for i ← 1 to n_1 do
    L[i] ← A[p + i − 1];
end
for j ← 1 to n_2 do
    R[j] ← A[q + j];
end
L[n_1 + 1] ← ∞;
R[n_2 + 1] ← ∞;
i ← 1;
j ← 1;
for k ← p to r do
    if L[i] ≤ R[j] then
        A[k] ← L[i];
        i ← i + 1;
    end
    else
        A[k] ← R[j];
        j ← j + 1;
    end
end
```

Algorithm 2: Merge(A,p,q,r)

Question 2 (10 points)

Question 8.1.3 in *The Design and Analysis of Algorithms*

Question 3 (10 points)

From *Algorithm Design and Applications* ISBN: 9781118335918

Binomial coefficients are a family of positive integers that have a number of useful properties and they can be defined in several ways. One way to define them is as an indexed recursive function, $C(n,k)$, where the C stands for "choices" or "combinations". In this case, the definition is as follows:

$$C(n,0) = 1,$$

$$C(n,n) = 1,$$

and, for $0 < k < n$,

$$C(n,k) = C(n-1,k-1) + C(n-1,k)$$

(a) Show that, if we don't use memoization, and n is even, then the running time for computing $C\left(n, \frac{n}{2}\right)$ is at least $2^{\frac{n}{2}}$

(b) Describe a scheme for computing $C(n, k)$ using memoization. Give a big-oh characterization of the number of arithmetic operations needed for computing $C\left(n, \lceil \frac{n}{2} \rceil\right)$ in this case.

Question 4 (10 points)

From *Algorithms* ISBN: 9780073523408

Given two strings $x = x_1, x_2 \ldots x_n$ and $y = y_1, y_2 \ldots y_m$, we wish to find the length of their *longest common substring*, that is, the largest k for which there are indices i and j with $x_i x_{i+1} \ldots x_{i+k-1} = y_j y_{j+1} \ldots y_{j+k-1}$. Show how to do this in time $O(mn)$.

Question 5 (10 points)

From *Algorithms* ISBN: 9780073523408

Counting Heads. Given integers n and k, along with $p_1, \ldots, p_n \in [0, 1]$, you want to determine the probability of obtaining exactly k heads when n biased coins are tossed independently at random, where p_i is the probability that the ith coin comes up heads. Give an $O\left(n^2\right)$ for this task. Assume you can multiply and add two numbers in $[0, 1]$ in $O(1)$ time. A $O(n \log n)$ time algorithm is possible.

Point Allocation

Question	Points
Question 1	10
Question 2	10
Question 3	10
Question 4	10
Question 5	10
`tallestTower`	
Test Cases	1×15
Compilation	10
`tallestTower` Total	25
`tugOfWar`	
Test Cases	1×15
Compilation	10
`tugOfWar` Total	25
Total	100