School of Computer Science
University of Guelph

CIS*3490 The Analysis and Design of Algorithms

Winter 2022
Instructor: Fangju Wang

Assignment 2 Guide

1.1 Develop a brute force algorithm based on the definition of *inversion*, which checks every pair of $(A[i], A[j])$ for $i < j$.

1.2 Modify the mergesort algorithm to count the number of inversions in $n \log n$ time.

2.1 Develop a brute force algorithm based on the definition of *convex hull.* The algorithm is to find the the convex hull for a given set of points. Please see Figure 3.5 on page 111. Develop your algorithm to find the nails holding the rubber-band, that is, to find the extreme points.

A brute force algorithm checks every point in the given data set to see if it is an extreme point (nail) of the convex polygon. An extreme point is an ending point of a straight boundary line. If all the points are on one side of a line, the line is a boundary line.

Please read

- "Convex-Hull Problem" on page 109 for the definition and an example of convex hull;
- the definition of extreme point (page 112);
- the equation of a straight line through two points (page 112);
- the method to check if all the points in a set are on one side of a line (page 112).

Then design an algorithm to find the shortest path.

- The equation for calculating the distance between points $s_1 = (x_1, y_1)$ and $s_2 = (x_2, y_2)$ is
$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

2.2 Design a divide-and-conquer algorithm of $\Theta(n \log n)$ based on the idea of quicksort. Please read

- "Convex-Hull Problem" on page 195 for the quick hull algorithm.

Then design an algorithm to find the shortest path.

- You can use the algorithm of 2.1

1.3, 2.3 Make sure your programs can be correctly compiled and executed on the Linux system in SoCS.

Your work should be submitted as a tar file containing files like
`readme, design, main.c, P11.c, P12.c, P21.c, P22.c, makefile`.

Please do not submit the data files.

Any compilation error or warning will result in a mark deduction. There will be some marks allocated for documentation.

Each file should have a comment at the beginning containing your name, id, date, and the assignment name.

The `readme` file should contain the following:

- name, id and assignment number
- a brief description of how to compile and run your programs, including how to enter the names of data files.

The `design` file should include the algorithms you design for 1.1, 1.2, 2.1, and 2.2, efficiency analysis, and comparisons. This can be a text, pdf or scanned file.

In your C files, each function should have a brief comment describing its purpose. Also, any section of code where it is not easily apparent what the code does should have a short comment.

You can use *timespec_get*() to get program running time.