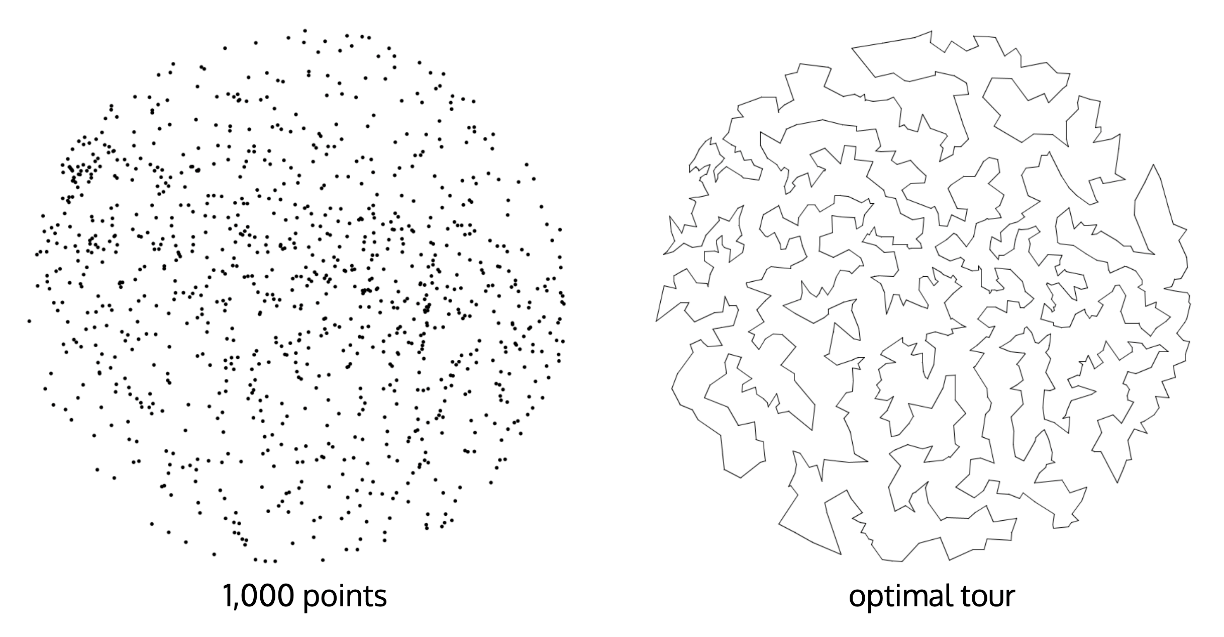
TSP (Extra Credit)



**Task Summary**

You will implement in this exercise two algorithms ([heuristics](https://en.wikipedia.org/wiki/Heuristic_(computer_science))) for finding approximate solutions for the traveling salesman problem. In both algorithms, the tour is built incrementally by adding the points one by one. Your task is to implement the functions that pick where in the tour newly added points should be inserted.

**Step 1.** Copy your implementation of class Tour from exercise 1 to **tour.cpp**.

**Step 2.** Implement the following two friend functions. The description of these functions is provided in the following subsection.

void insert\_closest(const Point2D& point, Tour<Point2D>& tour);

void insert\_smallest(const Point2D& point, Tour<Point2D>& tour);

**Step 3.** Test your program by clicking on the **Run** button. This will print the resulting tour, its length and will ***draw the tour on the screen.***

Do not write a main() function and do not modify the provided one. Use the sample input/output provided in the following subsections to check the correctness of your implementation.

**Closest Neighbor Heuristic** (+10 points)

Implement this heuristic in the following function:

// Inserts the given point after the closest point that is already in the tour.

void insert\_closest(const Point2D& point, Tour<Point2D>& tour);

// Example. Assume the points are: (0, 0), (4, 4), (1, 1)

// Starting with an empty tour, if we add the points one by one:

//

// 1. inserting (0, 0) makes it the beginning (and end) of the tour.

// (tour size = 1)

// 2. inserting (4, 4) makes the tour (0,0)-->(4,4)-->(0,0)

// (tour size = 2)

// 3. inserting (1,1) adds it after (0,0) because

// it is closeser to (0,0) than to (4,4)

// The tour becomes: (0,0)-->(1,1)-->(4,4)-->(0,0)

**Smallest Increase Heuristic** (+10 points)

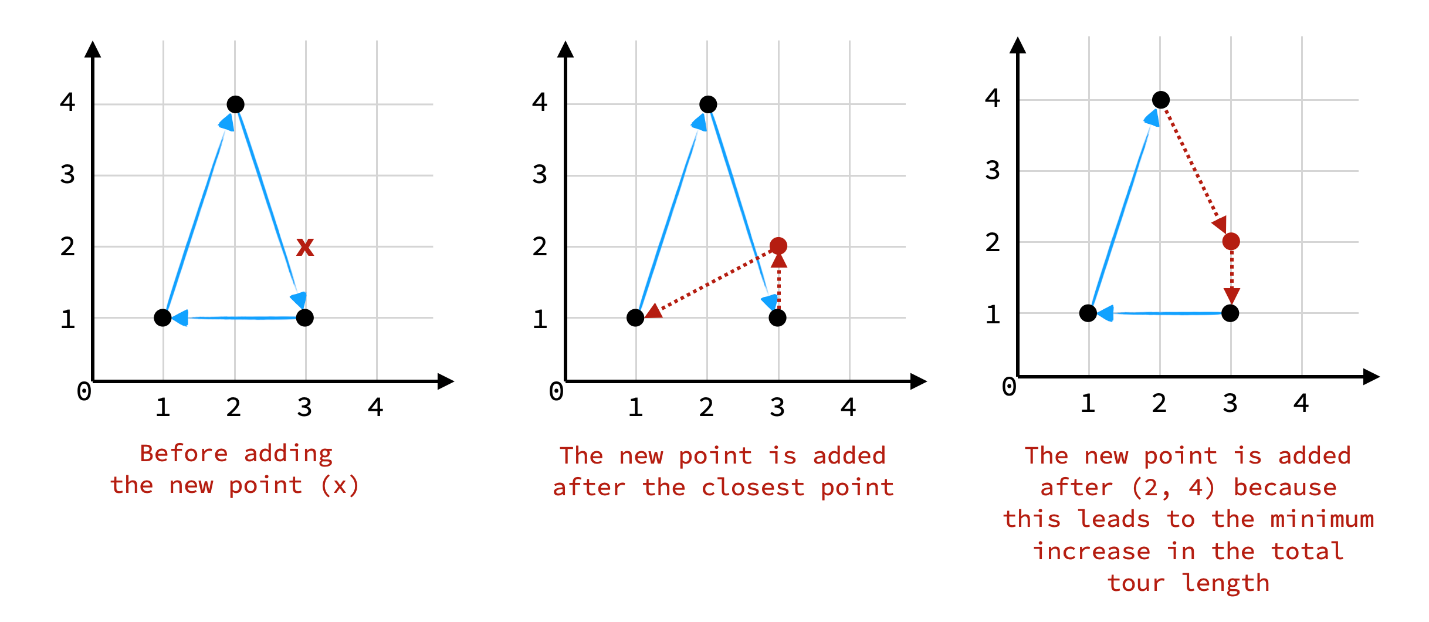
Implement this heuristic in the following function:

// Inserts the given point after the point currently in the tour

// where it results in the least possible increase in the tour length

void insert\_smallest(const Point2D& point, Tour<Point2D>& tour);

The following figure illustrates the difference between the two heuristics:



**Running Time**

* The running time of each function must be �(�)*O*(*n*), where �*n* is the number of nodes currently in the tour.
* In order to implement the **smallest increase** function, you will need to test the effect of inserting the new point after each point in the current tour (i.e. compare the length before the insertion to the length after the insertion). Avoid iterating through the whole tour each time to recompute the tour length.

**Sample Input and Output**

Input FileClosest NeighbourSmallest Increasetsp0.txtLength = 0Length = 0tsp1.txtLength = 0Length = 0tsp2.txt

|  |
| --- |
| A thin line on a white background  Description automatically generated Length = 632.46 |
| A triangle with a white background  Description automatically generated Length = 832.46 | A triangle with a white background  Description automatically generated Length = 832.46 |
| A triangle with a triangle in the middle  Description automatically generated Length = 956.06 | A black triangle on a white background  Description automatically generated Length = 839.83 |
| A black triangle with white lines  Description automatically generated with medium confidence Length = 2595.1 | A white diamond shaped object  Description automatically generated Length = 1872.8 |
| A white triangle with black lines  Description automatically generated Length = 3378.8 | A white corner with a black pencil  Description automatically generated with medium confidence Length = 2545.6 |
| A black and white outline of a triangle  Description automatically generated Length = 1566.1 | A white outline of a number  Description automatically generated Length = 1655.7 |
| A black and white image of lines  Description automatically generated Length = 7389.9 | A white line drawing of a circle  Description automatically generated with medium confidence Length = 4887.2 |
| A circular object with arrows  Description automatically generated Length = 27869 | A circular pattern with many shapes  Description automatically generated with medium confidence Length = 17266 |
| A black and white drawing of a figure  Description automatically generated with medium confidence Length = 6494.0 | A outline of a map  Description automatically generated Length = 4536.8 |
| A close-up of a white background  Description automatically generated Length = 25030 | A white outline of a building  Description automatically generated with medium confidence Length = 14587.2 |
| A map of germany made of black lines  Description automatically generated Length = 93119 | A black and white map of germany  Description automatically generated Length = 55754 |
| A black and white image of a person's face  Description automatically generated Length = 94894 | A black and white image of a person  Description automatically generated Length = 56334 |
| A close-up of a person's face  Description automatically generated Length = 161675 | A black and white image of a person's face  Description automatically generated Length = 95598 |
| A map of the united states  Description automatically generated Length = 77450 | A map of the united states  Description automatically generated Length = 45075 |