The Freeze Tag Problem
or
Do Swarms of Robots Dream of Electric Sheep

Isaac Banner
Rachel Silva

The Problem:

Imagine a swarm of robots on the plane, such that all but one robot are ‘asleep’ or in stand-by mode.

To awaken a robot, a currently active robot must go to its location on the plane.

Once a sleeping robot is awakened, it may assist in waking other sleeping robots.

The Objective:

Assuming general position principle, awaken every robot in the swarm in the shortest possible period of time.

The general case of the Freeze Tag Problem is NP-hard.

Our Goal:

We intend to explore and implement a number of sub-optimal algorithms and compare the theoretical running time, actual running time, and solutions with various test inputs.

Clarifications:

Our solutions plan ahead for the robots; they do not solve the problem in real time.

The first awake robot is given by input. We do not choose the first robot to be awake.


ZZZzz


ZZZzz


ZZZzz


ZZZzz

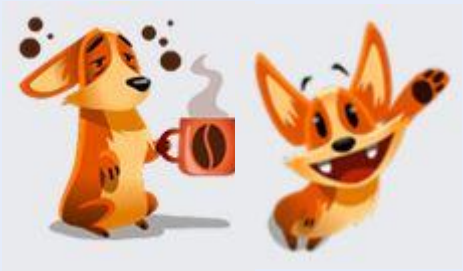

ZZZzz


ZZZzz


ZZZzz

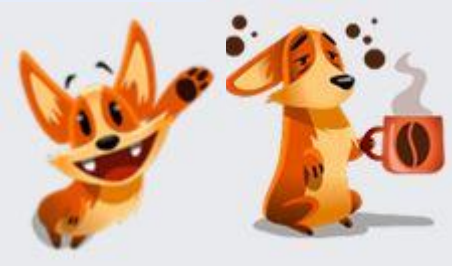

ZZZzz


ZZZzz




ZZZzz


ZZZzz



The Algorithms

Closest Neighbor Greedy Algorithm

Each awake robot is sent to the closest sleeping robot at the time that it is awoken or the time it has reached another robot.

Robots are considered awake when a robot is in transit to wake it, rather than when it is reached, to avoid scheduling conflicts.

Closest Neighbor Greedy Algorithm

Running Time:

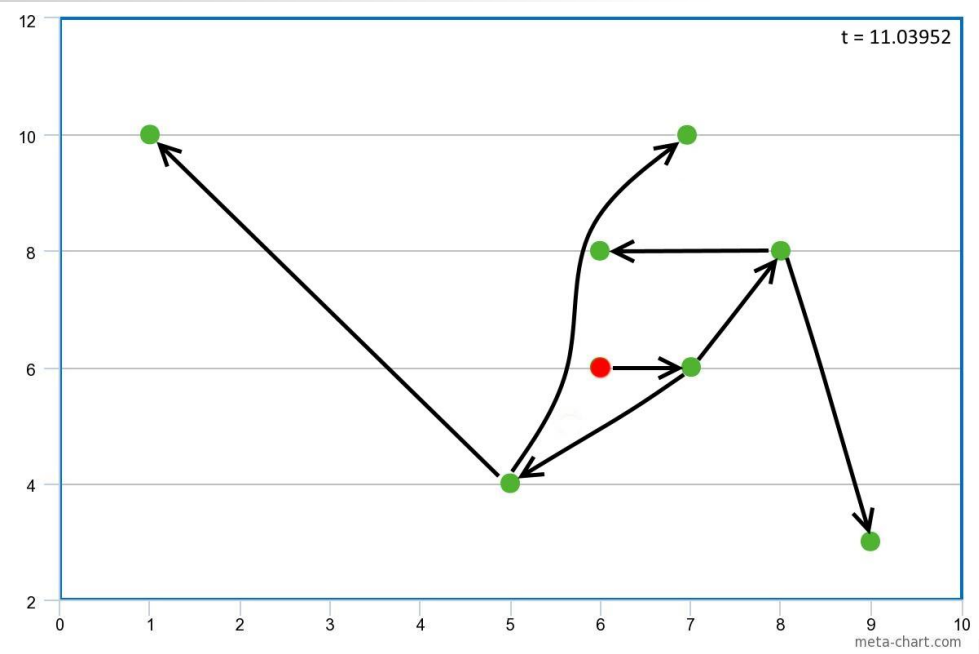
Our current implementation is $O(n^2 \log^2(n))$.

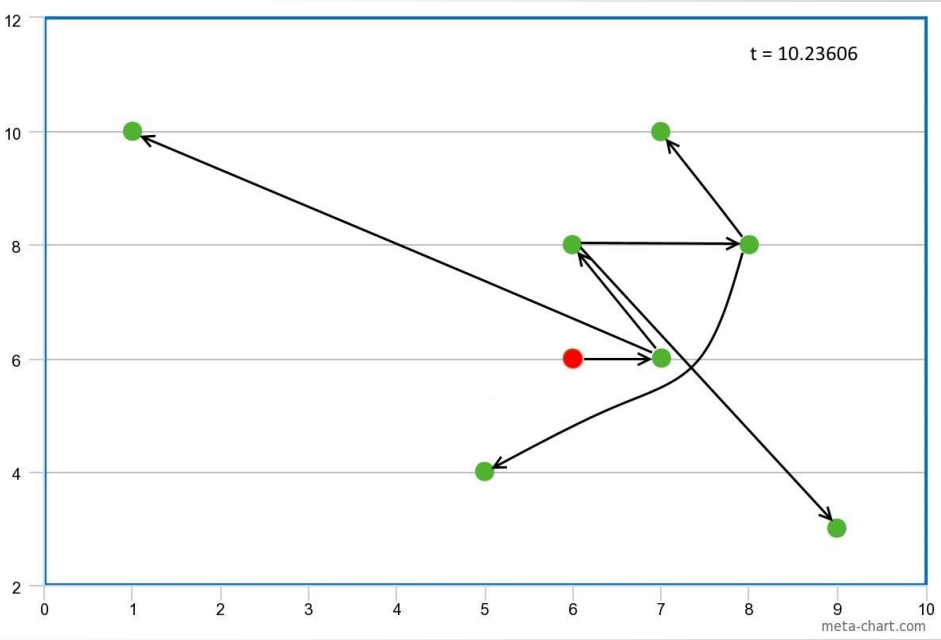
Alternating Distance Greedy Algorithm

Every other awake robot is sent to the furthest sleeping robot.

This algorithm otherwise functions identically to the previous algorithm.

Closest vs Alternating Distance





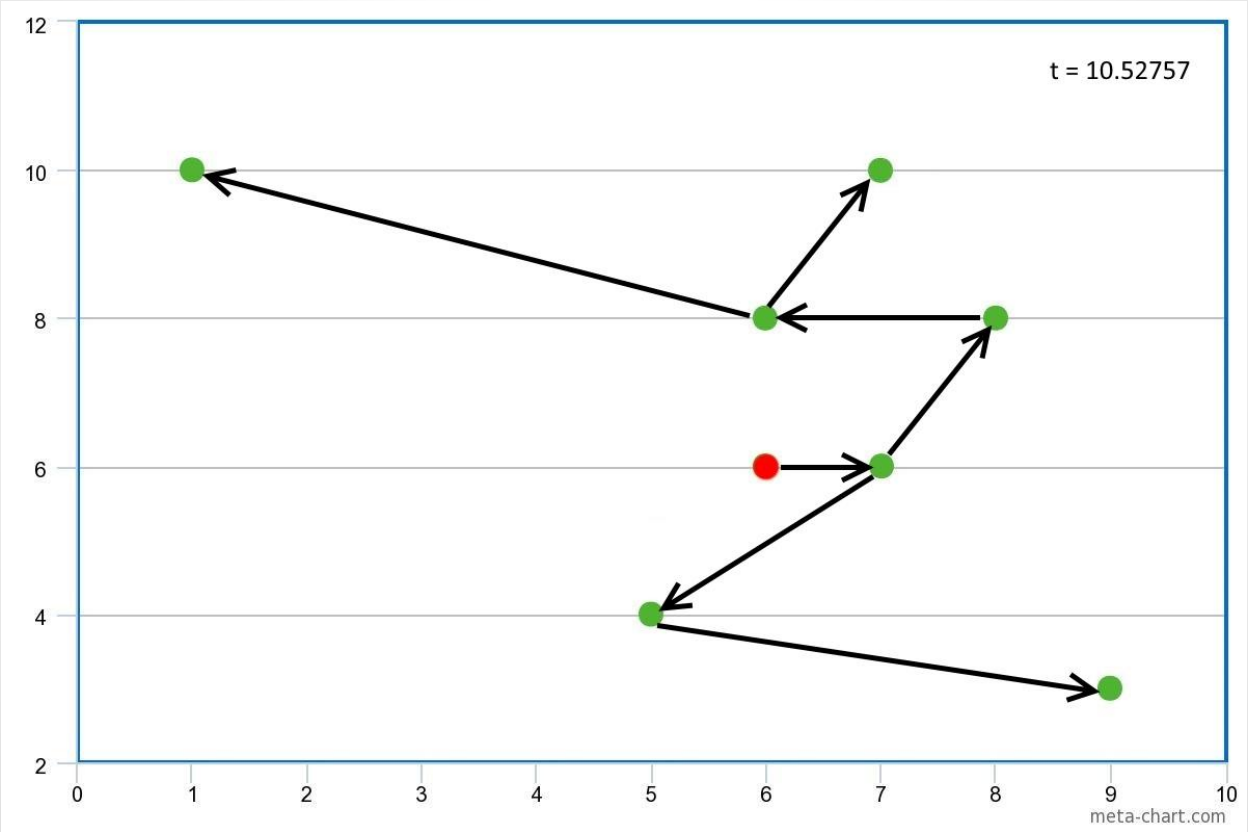
Modified Prim's Min Spanning Tree

Given a set of awake robots, find the shortest distance between an awake robot and a sleeping robot such that each robot has at most 3 edges.

Running Time:

$O(n^2 \log(n))$

Modified Prim's Min Spanning Tree

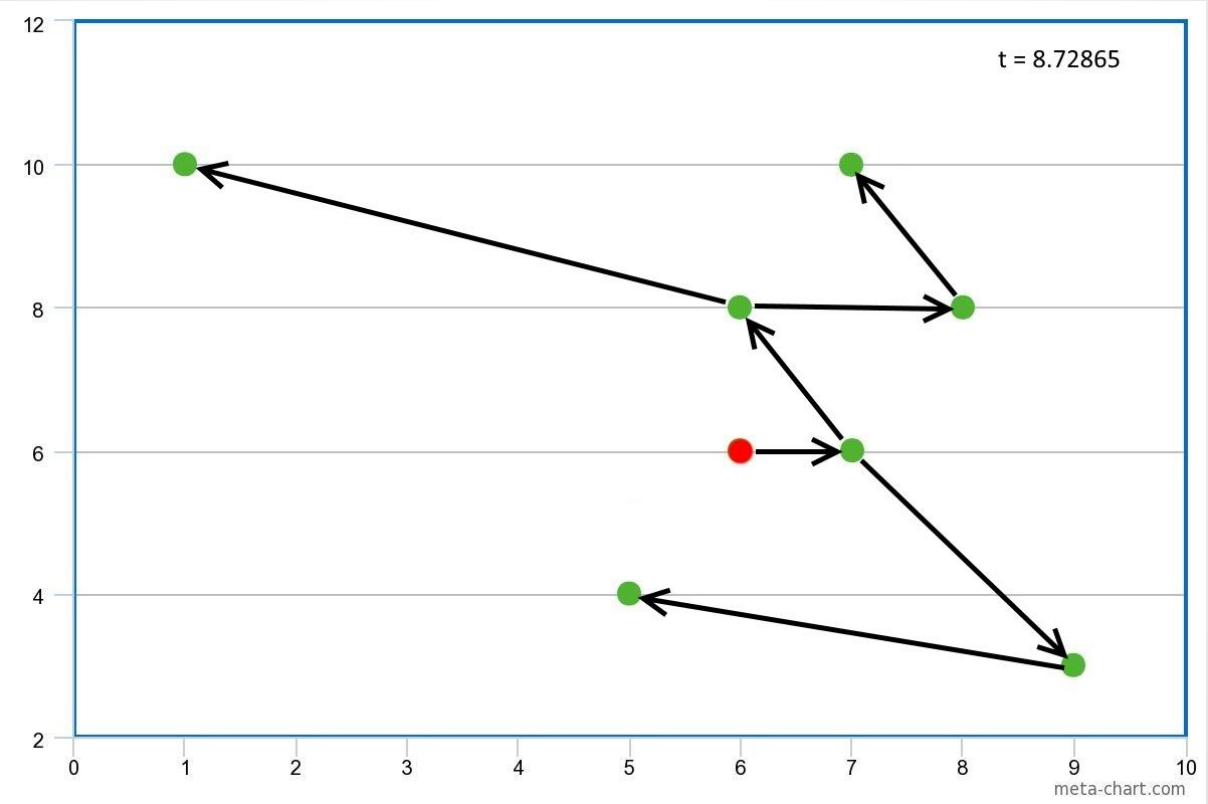


Brute Force

Our Brute Force algorithm worked by generating each ordering of points in lexicographic order. Each lexicographic ordering was treated similarly to a heap in that for any point in the ordering at index k (besides 0), its two child nodes in the traversal were at 2*k and 2*k + 1.

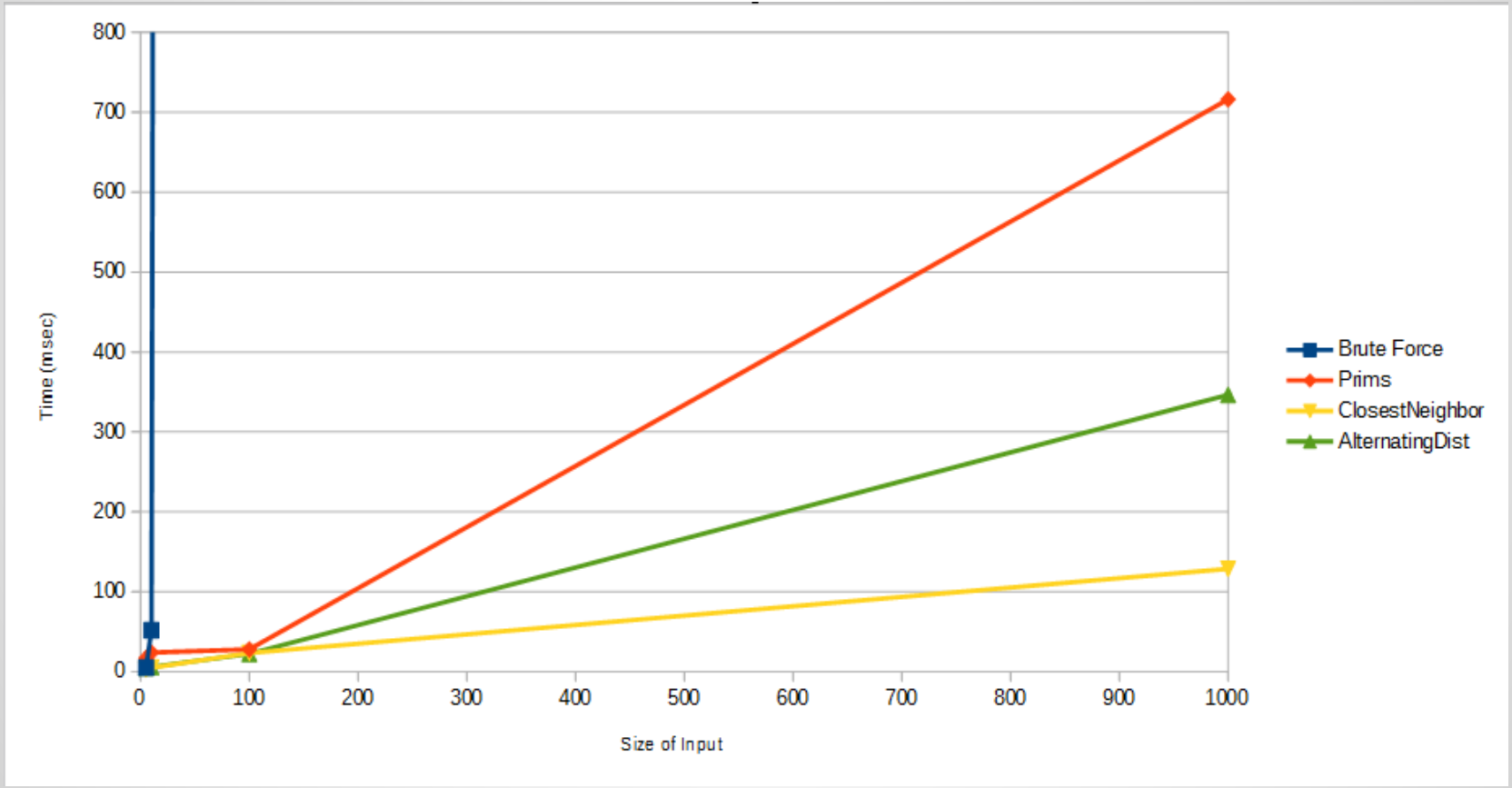
For example, 0-2-3-5-1-4-6-7

Brute Force *



*path was handcrafted - may not be optimal

Practical Running Times



Questions?