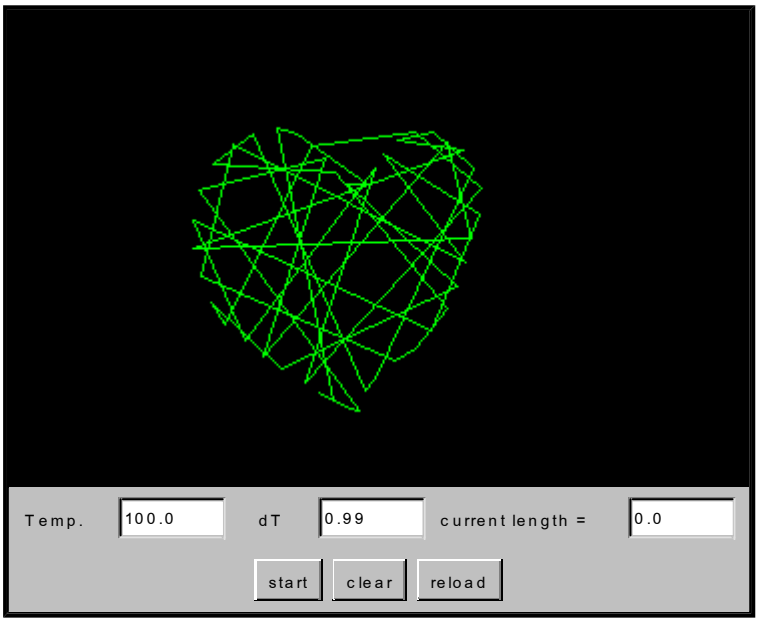
2110427 : การบ้านครั้งที่ 4

จุดประสงค์

- เขียนส่วนโปรแกรมที่แก้ปัญหาการเดินทางของพนักงานขายด้วยขั้นตอนวิธีการจำลองการอบเหนียว (simulated annealing) ผลที่ได้ดังแสดงเป็นตัวอย่างใน applet ข้างล่างนี้



ถ้าต้องการเพิ่มจุดเอง ให้นำเมาส์ไปกดเพิ่มได้ตามใจชอบ กด clear เมื่อต้องการลบจุดออกทั้งหมด กด reload เพื่อนำชุดเริ่มต้นมาใช้อีกครั้ง กด start เมื่อต้องการเริ่มทำงาน

ความเป็นมา

การจำลองการอบเหนียว (simulated annealing ขอเรียกสั้นๆ ว่า SA) เป็นกลวิธีหนึ่งซึ่งได้รับความนิยมมาก ในการแก้ปัญหาการหาค่าเหมาะที่สุดเชิงการจัด (combinatorial optimization problem) ซึ่งเป็นปัญหา NP–hard จะว่าไปแล้ว SA เป็นกลวิธีการค้นผลเฉลยแบบเฉพาะที่ (local search) ซึ่งมีกระบวนการทำงานแบบวนซ้ำ (iterative) เพื่อค้นผลเฉลยในปริภูมิผลเฉลยไปเรื่อยๆ จนกว่าจะได้ผลลัพธ์ที่พอใจ โดยเริ่มจากผลเฉลยเริ่มต้น

ในการบ้านครั้งนี้เราสนใจที่จะนำ SA มาใช้แก้ปัญหาการเดินทางของพนักงานขาย (traveling salesperson problem - ขอเรียกสั้นๆ ว่า TSP) โดยจะเป็น TSP บนจุดต่างๆ ในระนาบสองมิติ สิ่งที่ต้องการหาก็คือเส้นทางสั้นสุดที่ผ่านจุดทุกๆ จุดๆ ละหนึ่งครั้ง แล้วกลับมาที่จุดตั้งต้น TSP เป็นปัญหา NP-hard ในปัจจุบันยังไม่มีใครหาอัลกอริทึมซึ่งให้เส้นทางการเดินทางที่สั้นสุดสำหรับทุกๆ instances ของปัญหาได้ในเวลาแบบพหุนาม อย่างไรก็ตามเราสามารถใช้ SA หาเส้นทางที่สั้นๆ ได้ในเวลาที่ยอมรับได้

การทำงานของ SA นั้นล้อเลียนอุณหพลศาสตร์ของกระบวนการอบเหนียว (annealing) ซึ่งเป็นขั้นตอนการลดอุณหภูมิระหว่างการหลอมโลหะเพื่อให้ได้โลหะที่อยู่ในสภาวะที่เหมาะที่สุด (เพื่อให้ได้โลหะที่เหนียว ไม่เปราะ) ในกรณีที่เรานำ SA มาแก้ TSP เราเปรียบเส้นทางการเดินทางที่หาได้ปัจจุบันเสมือนกับเป็นสถานะปัจจุบันของโครงสร้างโลหะในระบบ เปรียบความยาวของเส้นทางเสมือนพลังงานของโครงสร้างของโลหะ และเปรียบการทำงานแบบวนซ้ำเสมือนกับการค่อยๆ ลดอุณหภูมิลงเรื่อยๆ ระหว่างการอบเหนียว ในวงวนการทำงานนั้นจะประกอบด้วยการสร้างเส้นทางใหม่โดยการปรับเปลี่ยนเส้นทางปัจจุบัน จากนั้นตัดสินใจว่าจะยอมรับเส้นทางใหม่ที่หาได้หรือไม่ ซึ่งเปรียบเสมือนกับการเปลี่ยนแปลงโครงสร้างของโลหะระหว่างการอบเหนียว

ประเด็นสำคัญก็อยู่ตรงที่เกณฑ์การยอมรับเส้นทางใหม่นี้ว่าจะตัดสินใจอย่างไร ตรงนี้เองที่อาศัยความรู้จากกระบวนการทางฟิสิกส์ระหว่างการอบเหนียว ซึ่งจะยอมรับโครงสร้างใหม่เสมอถ้าดีกว่า แต่ถ้าเลวลง ก็อาจจะยอมรับภายใต้ความน่าจะเป็นซึ่งมีค่าแปรตามอุณหภูมิขณะนั้นในรูปแบบ $e^{-dE/kt}$ (dE คือพลังงานที่เปลี่ยนแปลง k คือค่าคงตัวของโบลต์ซมันน์ – Boltzmann และ t คืออุณหภูมิปัจจุบันของการอบเหนียว) สำหรับกรณี TSP ก็หมายความเราจะยอมรับเส้นทางใหม่ที่หาได้เสมอถ้าเส้นทางใหม่นั้นมีความยาวสั้นกว่า แต่ถ้ายาวกว่าเราจะยอมรับภายใต้ความน่าจะเป็นในลักษณะเดียวกัน ตรงนี้ตีความได้ว่าเรายอมรับเส้นทางที่

ยาวกว่าได้ง่ายในระยะแรกของการวนซ้ำ (เมื่ออุณหภูมิสูง) แต่จะยอมรับเส้นทางที่ยาวกว่ายากขึ้นเรื่อยๆ เมื่ออุณหภูมิลดลง และยอมรับเส้นทางที่ยาวกว่าน้อยได้ง่ายกว่าที่ยาวกว่ามาก การวนซ้ำจะสิ้นสุดเมื่อถึงอุณหภูมิเย็น หรือถึงสภาพที่ไม่ได้ผลเฉลยที่ดีกว่าเป็นเวลาเกินที่ตั้งเป็นเกณฑ์ไว้ เขียนโครงหลักของการทำงานได้ดังนี้

```
TspSA( p ) {
  tour = initialTour( p )
  temp = high_temperature
  while not terminate( tour ) {
    newTour = nextTour(tour)
    dE = length(newTour,p)-length(tour,p)
    if ( dE < 0 || random(0,1) < e^(-dE/kt) )
       tour = newTour
    temp = temp * tempFactor
  }
}
```

สิ่งที่ต้องทำ

เขียนโปรแกรมใน class Hw4.java ให้สมบูรณ์ public methods ใน class นี้ที่ต้องเขียนคือตัว constructor และ methods initialTour และ oneAnnealingStep ผมเขียนโปรแกรมหลักในการทำวงวนของ SA ไว้แล้วข้างล่างนี้ (อยู่ใน class อื่น)

```
      ...

53:   Hw4 hw4 = new Hw4( distanceTable );
54:   tour = hw4.initialTour();
55:   while { temp > 0.001 && inProgress() ) {
56:     tour = hw4.oneAnnealingStep( tour, temp );
57:
58:     showResult( tour );
59:
60:     try {
61:       Thread.sleep(100L);
62:     } catch(InterruptedException e) { }
63:
64:     temp = temp * tempFactor;
65:   }
66:
      ...
```

ก่อนจะเข้าวงวนของ SA ผมสร้าง object ของ class Hw4 ก่อน (บรรทัดที่ 53) โดยส่ง distanceTable ซึ่งเป็น array สองมิติขนาด n x n โดยที่ n คือจำนวนจุดในระนาบ distanceTable[i][j] จะเก็บระยะทางระหว่างจุด i กับ j บนระนาบ จากนั้นเรียก initialTour (บรรทัดที่ 54) เพื่อขอเส้นทางแรก

จากนั้นเข้าสู่วงวนของกระบวนการ SA ในหนึ่งรอบจะเรียก oneAnnealingStep (บรรทัดที่ 56) หนึ่งครั้ง โดยส่งเส้นทางการเดินทางปัจจุบัน และค่าของอุณหภูมิไปให้ เมื่อ method นี้ทำงานเสร็จ จะต้องคืนเส้นทางการเดินทางใหม่กลับคืนมา จากนั้นก็แสดงผลลัพธ์บนจอภาพ (บรรทัดที่ 58) หยุดการทำงานสักครู่ (บรรทัดที่ 60 ถึง 62) ลดอุณหภูมิลงนิดหน่อย (บรรทัดที่ 64) จากนั้นกลับสู่ต้นวงวนเพื่อตรวจสอบว่าควรทำงานต่อหรือไม่ (บรรทัดที่ 55) โดยจะหยุดการทำงานเมื่ออุณหภูมิเย็นพอ หรือไม่ก็เมื่อผู้ใช้กดปุ่ม stop

หมายเหตุ : เส้นทางการเดินทาง (หรือ tour) ที่คืนมานั้นเป็น array ของ int ที่เก็บลำดับของหมายเลขจุดของ tour

แฟ้ม Hw4.java ที่นักเรียนต้องเขียนนั้นมีโครงให้ดังนี้

```
/**
 * Title: 2110427 (2544) : HW4
 * Description: Traveling Salesperson Problem + Simulated Annealing
 * Copyright:   Copyright (c) 2001
 * Company:
 * @author
 * @version 1.0
 */

public class Hw4 {

    public Hw4( double[][] distanceTable ) {

        ...
    }
    public int[] initialTour() {

        ...

        return tour;
    }

    public int[] oneAnnealingStep( int[] tour, double temperature ) {

        ...

        return tour;
    }
}
```

Download และการทดสอบ

- unzip แฟ้ม 427hw4.zip ลงใน directory ที่เก็บเฉพาะการบ้านนี้ จะพบแฟ้มต่างๆ ดังนี้
    - Hw4.java (แฟ้มที่ต้องเขียน methods ต่างๆ ของการบ้านครั้งนี้)
    - hw4.jar (แฟ้มเก็บ class สำคัญอื่นๆ ของ applet)
    - map.txt (แฟ้มเก็บพิกัดจุด มีไว้แสดงเป็นตัวอย่างตอนเริ่มทำงาน)
    - hw4.htm (แฟ้มที่กำลังอ่านอยู่นี้)
    - hw4.pdf (เอกสารที่กำลังอ่านอยู่นี้แบบ pdf)
- แก้ไขแฟ้ม Hw4.java ด้วย editor อะไรก็ได้ (เช่น notepad, edit plus, ...)
- compile ด้วย javac -classpath .;hw4.jar Hw4.java
- สั่งทำงานด้วย appletviewer hw4.htm
  (หรือจะเปิด hw3.htm ด้วย web browser ก็ได้)

กำหนดส่ง

- วันที่ 3 กันยายน 2544