The Linear Search Algorithm
The 2-Dimensional Closest Pair Algorithm
The Unique Element Algorithm
The Heap Sort Algorithm

พีรยา คันธารวงสกุล ศุภกร รักนะ
สิทธินนท์ โชติวิไลวรรณ สิริวัฒน์ โชติเลอศักดิ์

Abstract

รายงานฉบับนี้ประกอบด้วยเนื้อหาของสี่อัลกอริทึมอันได้แก่ Linear Search Algorithm, 2-Dimensional Closest Pair Algorithm, Unique Element Algorithm และ Heap Sort Algorithm ซึ่งรายละเอียดประกอบไปด้วยการกล่าวถึงที่มาของปัญหาเบื้องต้น อัลกอริทึมพื้นฐาน การเพิ่มประสิทธิภาพให้อัลกอริทึม การพิสูจน์ทางคณิตศาสตร์ที่สำคัญ การทดลองเปรียบเทียบและเนื้อหาเพิ่มเติมจากหนังสือหรือบทความงานวิจัยที่น่าสนใจ มิได้กล่าวถึงความรู้พื้นฐานเกี่ยวกับโครงสร้างข้อมูลและการนำไปใช้ในชีวิตประจำวันแต่อย่างใด

สิ่งที่แนบไปด้วยมีงานนำเสนอซึ่งจะมีรายละเอียดบางส่วนของรายงานและโปรแกรมพร้อมคู่มือการใช้งานแนะนำให้อ่านควบคู่กับรายงานเพื่อความเข้าใจที่มากที่สุด[1] ในหัวข้อสุดท้ายเป็นชุดข้อมูลทดสอบสำหรับผู้พัฒนาอัลกอริทึมที่ต้องการเปรียบเทียบประสิทธิภาพ

Instructor

รศ.ดร.ณัฐชา เดชดำรง

[1] https://drive.google.com/drive/folders/1HZ_VxBqOt6_u3bxLjNwX3G3ejVv59SW8?usp=sharing

1. Linear search algorithm

1.1.Search algorithm

หากพิจารณาการเก็บข้อมูลในหน่วยความจำของคอมพิวเตอร์ในกรณีที่สามารถนำข้อมูลนั้นมาใช้ได้โดยเร็วที่สุด บางครั้งต้องเผชิญกับข้อมูลจำนวนมากเกินกว่าที่จะนำไปใช้ได้จริง วิธีการแก้ไขปัญหานี้ทำได้ง่ายโดยลบข้อมูลส่วนที่ไม่ต้องการทิ้งไป ทว่าข้อมูลนั้นก็จะไม่สามารถนำกลับมาใช้ได้อีกต่อ จึงเป็นการดีกว่าที่จะเก็บข้อมูลนั้นไว้พร้อมกับจัดการข้อมูลเพื่อการนำกลับมาใช้ได้เร็วขึ้น การค้นหาข้อมูลจึงเป็นสิ่งสำคัญ

search algorithm สามารถอธิบายได้ว่า การค้นหาข้อมูลที่เก็บไว้โดยใช้ลักษณะเฉพาะบางอย่างเช่น ค่าของข้อมูล, ตำแหน่งของข้อมูล หรือค่าจากการ hash เป็นต้น การหาค่า $f(x)$ จากการป้อนค่า x, หรือการหาคำศัพท์ภาษาอังกฤษจากการป้อนค่าเป็นภาษาไทย ก็ถือเป็นการค้นหาข้อมูลชนิดหนึ่ง

1.2.Linear search algorithm

Linear search หรือ Sequential search เป็นหนึ่งใน search algorithm โดยในปี 1988 *Donald Knuth* ได้ให้คำนิยามของ Linear search ไว้ว่า "**เริ่มค้นหาจากตำแหน่งแรกและดำเนินต่อไปจนเจอค่าที่ถูกต้อง จากนั้นจึงหยุด**"[1] จากคำนิยามนี้ จะเห็นว่าแนวคิดของ Linear search นั้นเห็นได้ชัดและเข้าใจได้ง่ายที่สุด

1.3.Linear search algorithm ทั่วไป

1.3.1.Pseudocode และ diagram

จากนิยามได้ข้อ 1.2. สามารถอธิบายเพื่อความละเอียดได้ดังนี้

Algorithm *A* (Linear search)

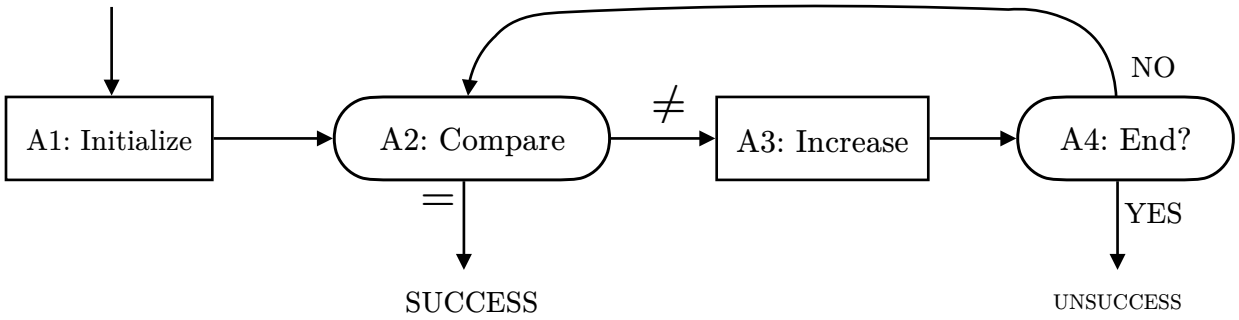
A0: **Input**: A non-empty record R_1, R_2, …, R_N whose respective keys are K_1, K_2, …, K_N search for a given argument K.

A1: [Initialize] Set $i \leftarrow 0$.

A2: [Compare] If $K = K_i$, the algorithm terminate successfully.

A3: [Increment] Increase i by 1.

A4: [End?] If $i \leq N$, go back to **A2**.
Otherwise the algorithm terminate unsuccessfully.



จะเห็นได้ว่าอัลกอริทึมนี้สามารถสิ้นสุดการทำงาน (terminate) ได้สองกรณีคือ successfully (พบค่าที่ต้องการค้นหา) และ unsuccessfully (ไม่พบค่าที่ต้องการค้นหา)

1.3.2.วิเคราะห์อัลกอริทึม

ส่วนนี้จะเป็นการวิเคราะห์อัลกอริทึมในเชิงเวลา ซึ่งขึ้นกับสองปัจจัยคือ

C = การเปรียบเทียบและการให้ค่า;

$S = 1$ (successful *termination*), 0 (unsuccessful *termination*)

ตาราง 1.1: อัลกอริทึม A มีการทำงานสอดคล้องกับปัจจัยดังนี้

Operation	Cost	หมายเหตุ
A1: Initialize K & i	1+1	
A2: Compare	C	
$K = K_i$?	C	
A3: Increase	C - S	ไม่เพิ่ม i ทำให้ $S = 0$
A4: End?	C - S	
Unsuccessful *termination*	1 - S	Success: 1-1 = 0 ไม่เกิด cost
Unit of time = 5C - 2S + 3		

• กรณี successful *termination* จะได้ $C = i$ และ S = 1 หน่วยเวลาเป็น $5i + 1$

• กรณี unsuccessful *termination* จะได้ $C = N$ และ S = 0 หน่วยเวลาเป็น $5N + 3$

สมมุติให้ทุก ๆ ข้อมูลในรายการมีความน่าจะเป็นที่จะพบเท่ากัน ดังนั้นค่าเฉลี่ยของ C ใน กรณี successful *termination* จะเป็น

$$\frac{1 + 2 + \cdots + N}{N} = \frac{N + 1}{2}$$

1.3.3.Complexity

Time complexity ได้จาก $\overline{C} \in \Theta(N)$

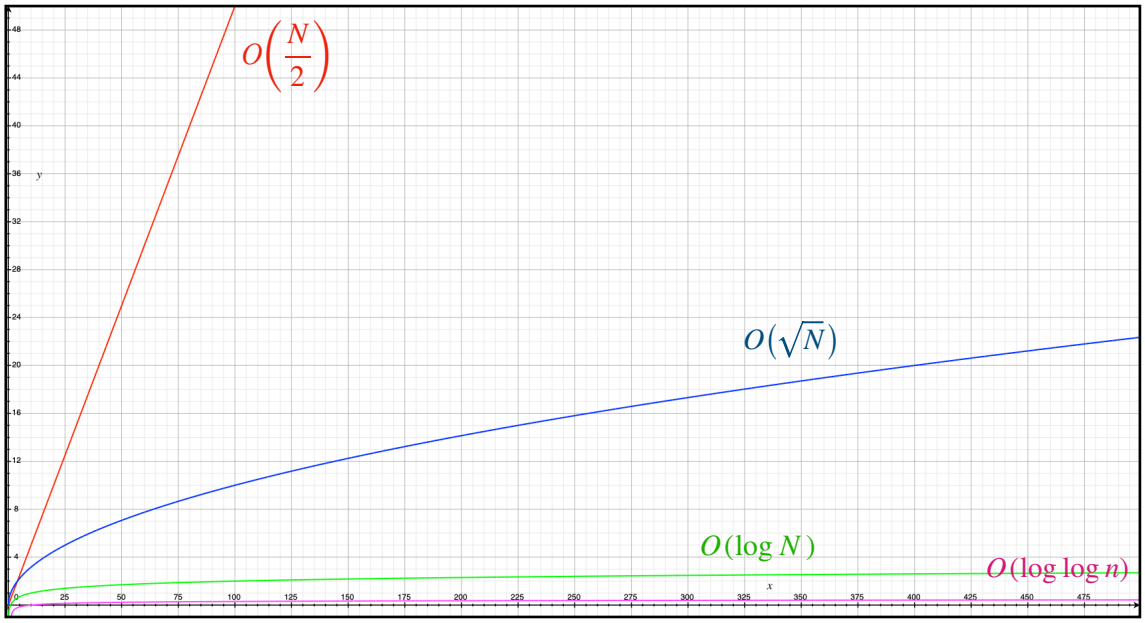
K_1	K_2	...	K_{N-1}	K_N	
Best case: $\Omega(1)$ เมื่อ $K = K_1$				Worst case: $O(N)$ เมื่อ $K = K_N$ หรือ Unsuccessful termination	

1.3.4.เปรียบเทียบกับ search algorithm อื่น

ตารางเปรียบเทียบในโครสร้างข้อมูลที่มีลักษณะเดียวกัน

Search algorithm	Bese-case	Average	Worst-case
Linear search	$O(1)$	$O\left(\frac{N}{2}\right)$	$O(N)$
Binary Search	$O(1)$	$O(\log N)$	$O(\log N)$
Jump search	$O(N)$	$O(\sqrt{N})$	$O(\log N)$
Interpolation search	$O(1)$	$O(\log \log n)$	$O(N)$
Exponential search	$O(1)$	$O(\log \text{index of key})$	$O(\log \text{index of key})$
Fibonacci search	$O(1)$	$O(\log N)$	$O(\log N)$

กราฟ 1.1: เปรียบเทียบ Average complexity กับจำนวนข้อมูล



Linear search มี complexity สูงมากเมื่อเทียบกับ search algorithm อื่น ๆ เมื่อกรณีที่พิจารณาโครงสร้างข้อมูลเดียวกัน (array)

1.4.Linear search algorithm (Quick)

อัลกอริทึมในข้อ 1.3. เป็นอัลกอริทึมที่เห็นได้ทั่วไปและเป็นที่คุ้นเคย แต่สามารถทำงานได้ดีในกรณีที่ข้อมูลมีจำนวนน้อยเท่านั้น เสนอการเพิ่มประสิทธิภาพในเชิงของ unit of time ดังต่อไปนี้[1]

K_1	K_2	...	K_{N-1}	K_N	K_{N+1}
					Dummy record

1.4.1.Pseudocode

Algorithm B (Quick linear search)

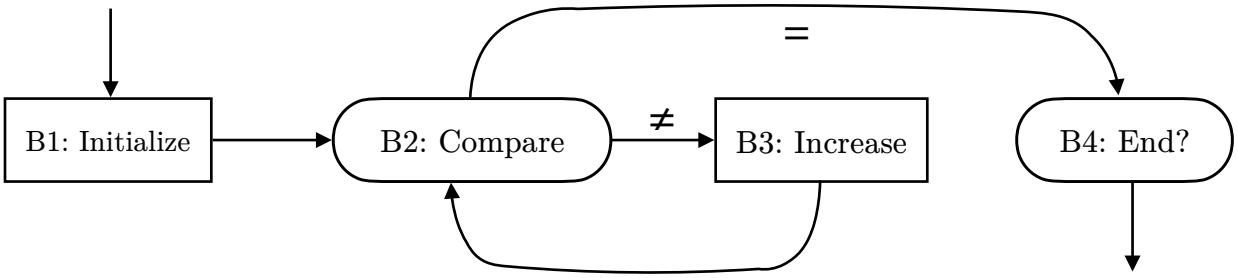
B0: **Input**: Same as algorithm A,*Plus assume a dummy record* R_{N+1} at the end.

B1: [Initialize] Set $i \leftarrow 0$ and set $K_{N+1} \leftarrow K$.

B2: [Compare] If $K = K_i$, go to **B4**

B3: [Increment] Increase i by 1 and return to **B2**

B4: [End?] If $i \leq N$, the algorithm terminates successfully;
otherwise it terminates unsuccessfully ($i = N+1$).



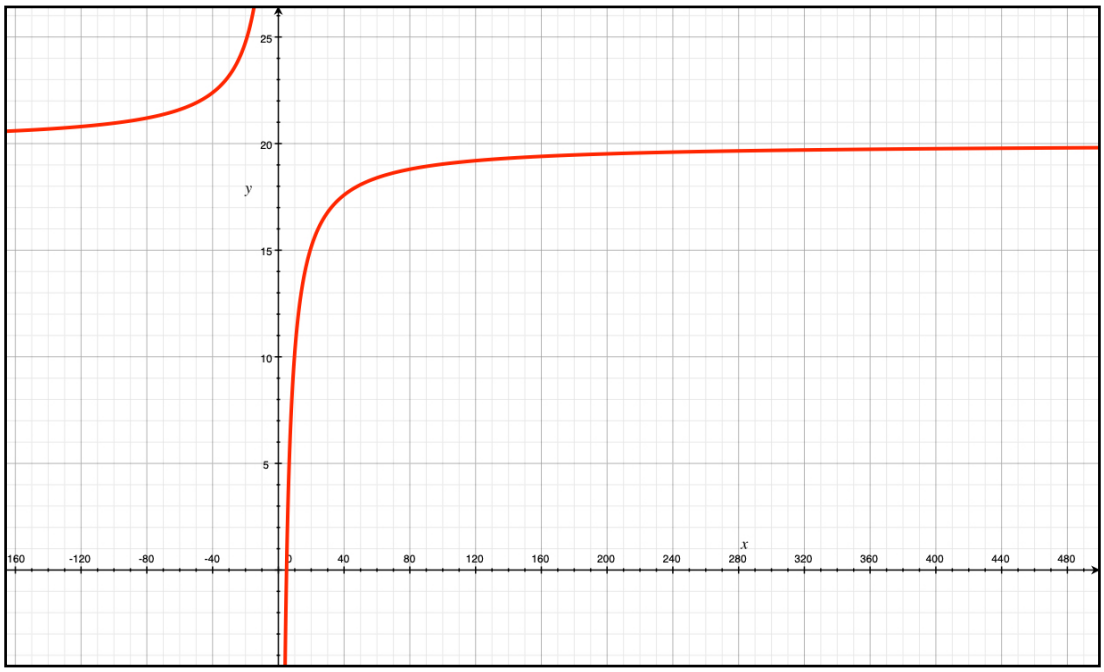
1.4.2.วิเคราะห์อัลกอริทึม

ส่วนนี้จะเป็นการวิเคราะห์อัลกอริทึมในเชิงเวลา ซึ่งขึ้นกับสองปัจจัยคือ C และ S เช่นเดิม ซึ่งอัลกอริทึม B มี unit time running เป็น $4C - 4S + 10$

กรณี successful *termination* ได้หน่วยเวลาเป็น $4i + 6$

เทียบกับอัลกอริทึม A: $(4C + 6) < (5C + 1)$ จะได้ $C > 5$ หมายถึงอัลกอริทึม B จะมี time unit ต่ำกว่า (ใช้เวลาน้อยกว่าในเชิงทฤษฎี) เมื่อค้นเจอข้อมูลที่ต้องการตั้งแต่ 6 ตัวขึ้นไป

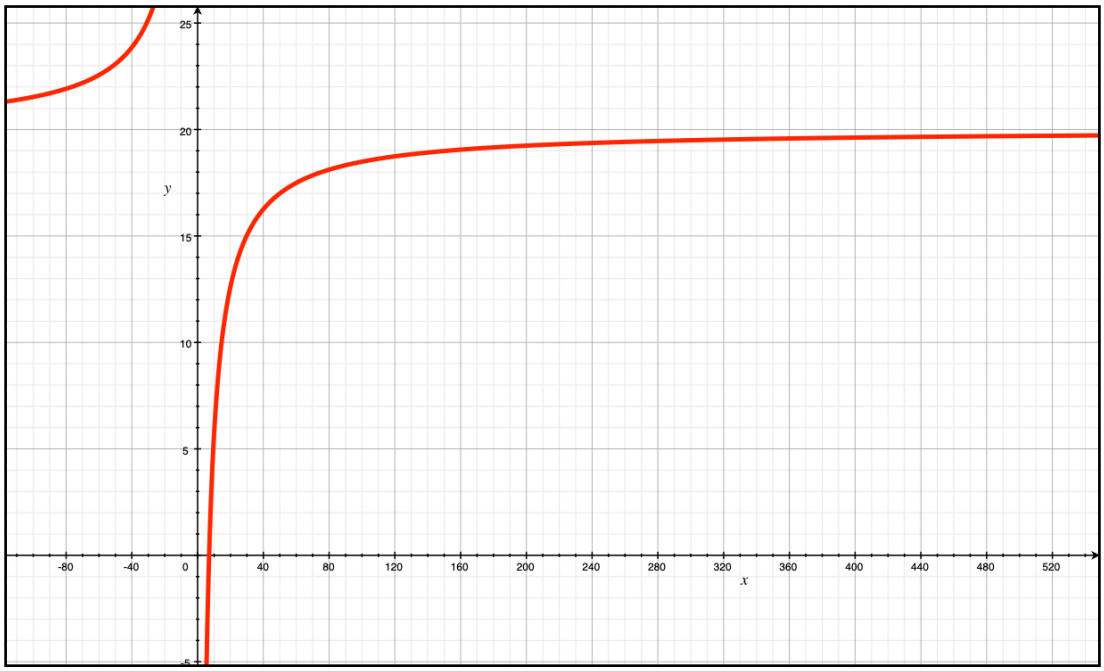
กราฟ 1.2: แสดงเวลาที่ลดลงในหน่วยเปอร์เซนต์เป็นฟังก์ชัน $100(C-5)/(5C-1)\,\%$ เทียบกับตำแหน่งที่ค้นเจอข้อมูล



กรณี unsuccessful *termination* จะได้ $C = N$ และ $\mathrm{S} = 0$ ได้หน่วยเวลาเป็น $4N+10$

เทียบกับอัลกอริทึม A: $(4N+10) < (5N+3)$ จะได้ $\mathrm{N} > 7$ หมายถึงอัลกอริทึม B จะมี time unit ต่ำกว่า (เร็วกว่าในเชิงทฤษฎี) เมื่อมีข้อมูลตั้งแต่ 8 ตัวขึ้นไป

กราฟ 1.3: แสดงเวลาที่ลดลงในหน่วยเปอร์เซนต์เป็นฟังก์ชัน $100(N-7)/(5N+3)\,\%$ เทียบกับจำนวนข้อมูล



อัลกอริทึม B สามารถลดเวลาลงได้มากถึง 20% จากการลู่เข้าของฟังก์ชันที่ $+\infty$

1.5.จะเกิดอะไรขึ้นถ้าค้นหาข้อมูลในชุดข้อมูลที่เรียงลำดับแล้ว?

เมื่อถึงตำแหน่งหนึ่งที่ค่าของตัวค้นเริ่มมากกว่าค่าในชุดข้อมูลการค้นจะสิ้นสุดลงไม่ว่าจะค้นเจอหรือไม่ก็ตาม ซึ่งเงื่อนไขนี้ทำให้การทำงานใช้เวลาน้อยลงเป็นอย่างมาก

1.5.1.Pseudocode และ diagram

Algorithm C (Presort Linear search)

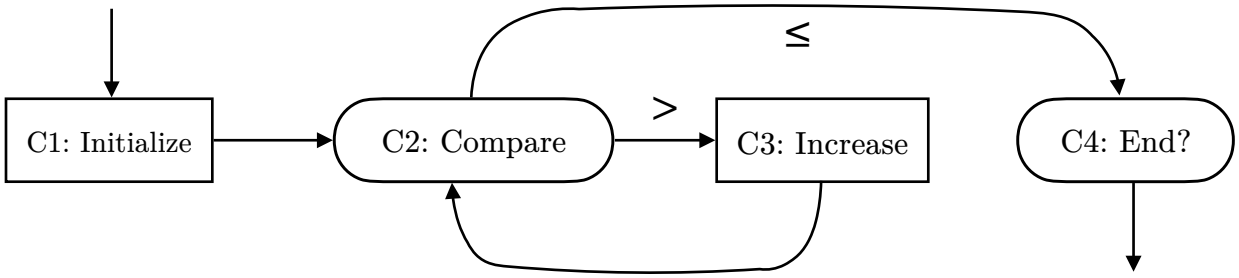
C0: **Input**: A non-empty record R_1, R_2, …, R_N, whose respective keys are K_1, K_2, …, K_N, search for a given argument K and dummy record R_{N+1} with $K_{N+1} = \infty$

C1: [Initialize] Set $i \leftarrow 0$.

C2: [Compare] If $K \leq K_i$, go to **C4**.

C3: [Increment] Increase i by 1 and return to **C2**.

C4: [End?] If $K = K_i$, the algorithm terminates successfully.
Otherwise the algorithm terminate unsuccessfully.



จุดน่าสนใจ ถ้า key ทั้งหมดมีโอกาสเท่ากัน อัลกอริทึมนี้จะใช้เวลาเฉลี่ยเท่ากับอัลกอริทึม B ในกรณี successful termination แต่ใน กรณี unsuccessful termination จะใช้เวลาน้อยกว่าถึง 2 เท่า

1.6.กรณีที่โอกาสในการค้นเจอข้อมูลแต่ละตัวไม่เท่ากัน[1]

ในหัวข้อที่ผ่านมาเป็นกรณีที่โอกาสในการค้นเจอข้อมูลแต่ละตัวเท่ากัน แต่ในหัวข้อนี้กำหนดให้โอกาสสำหรับการค้นเจอ K_j คือ p_j สำหรับ $p_1 + p_2 + \ldots + p_N = 1$ โดยปกติเวลาที่ใช้สำหรับ successful termination จะขึ้นกับค่า C ซึ่งขณะนี้จะมีค่าเฉลี่ยเป็น

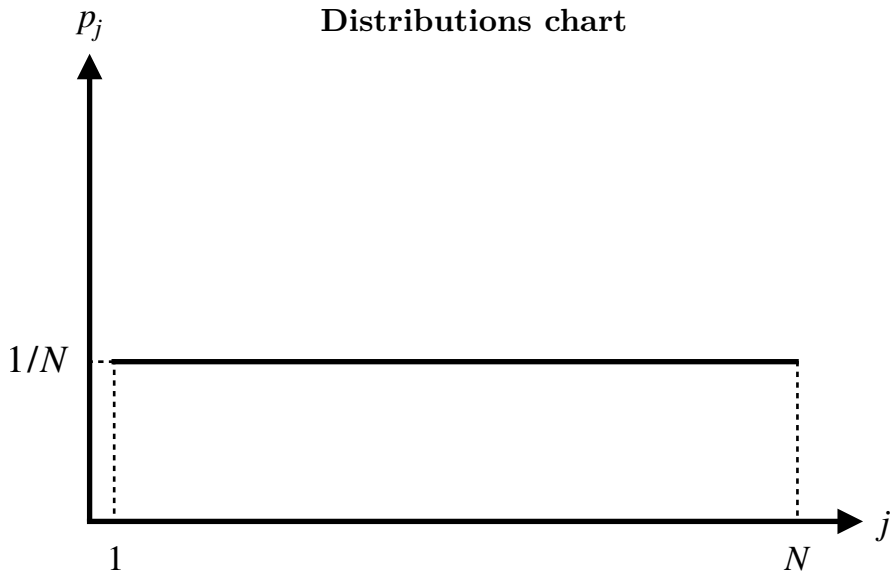
$$\overline{C}_N = 1p_1 + 2p_2 + \cdots + Np_N$$

ค่า $\overline{C}_N$ จะยิ่งน้อยหากโอกาสที่จะค้นเจอข้อมูลตัวที่อยู่ใกล้จุดเริ่มต้นสูง โดยค่าที่น้อยเกิดขึ้นเมื่อลักษณะของข้อมูลสอดคล้องเงื่อนไข $p_1 \geq p_2 \geq \cdots \geq p_N$ ตัวอย่างดังต่อไปนี้

I. กำหนดให้ $p_1 \geq p_2 \geq \cdots \geq p_N$

จากเงื่อนไขข้างต้น ถ้า $p_1 = p_2 = \cdots = p_N = \frac{1}{N}$ จะได้ $\overline{C}_N = \frac{N+1}{2}$

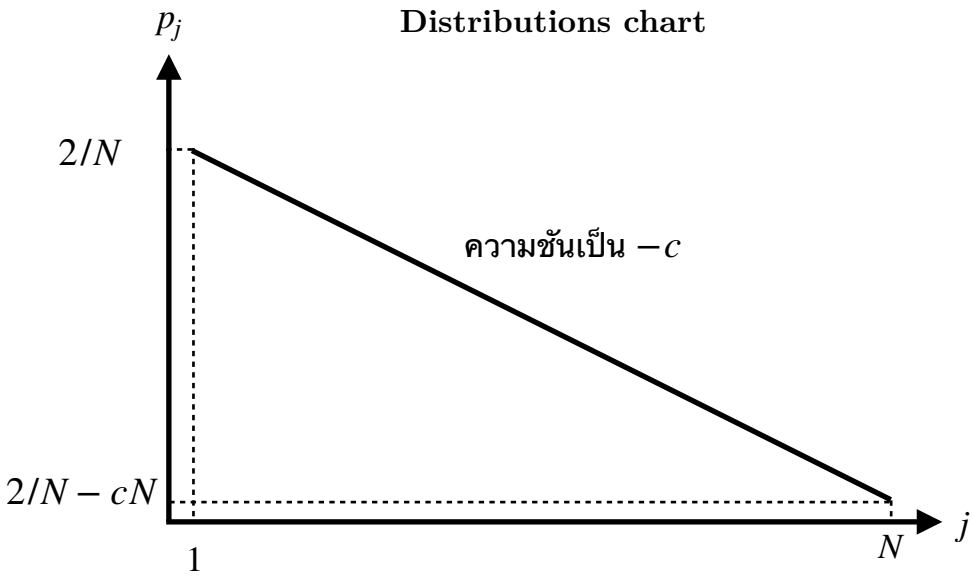
เช่นเดียวกับอัลกอริทึม A



II. กำหนดให้ $p_j = (N - j + 1)c$ โดยที่ $c = \frac{2}{N(N+1)}$

จะได้ $\overline{C}_N = \sum_j j p_j = \frac{N+2}{3}$

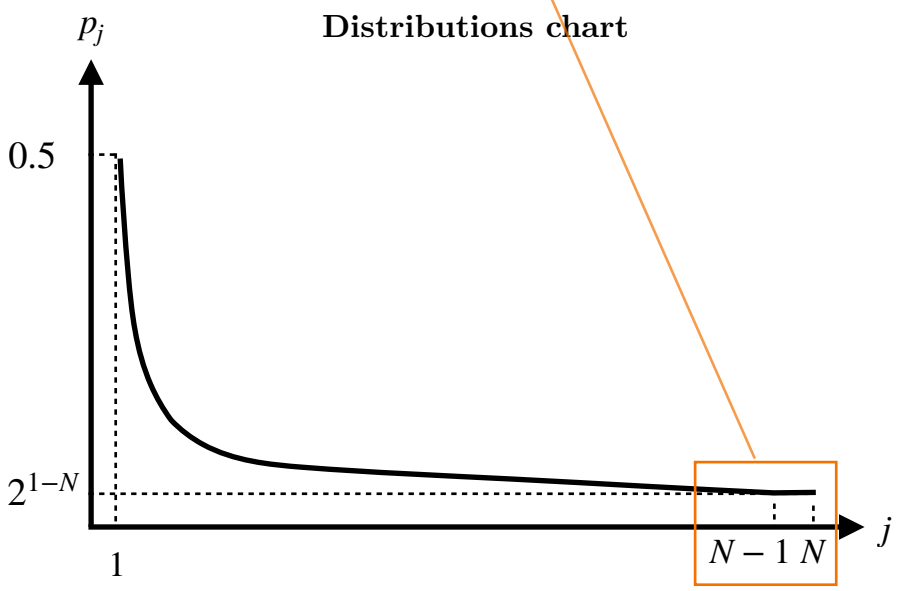
จัดรูป $p_j = -cj + \frac{2}{N}$



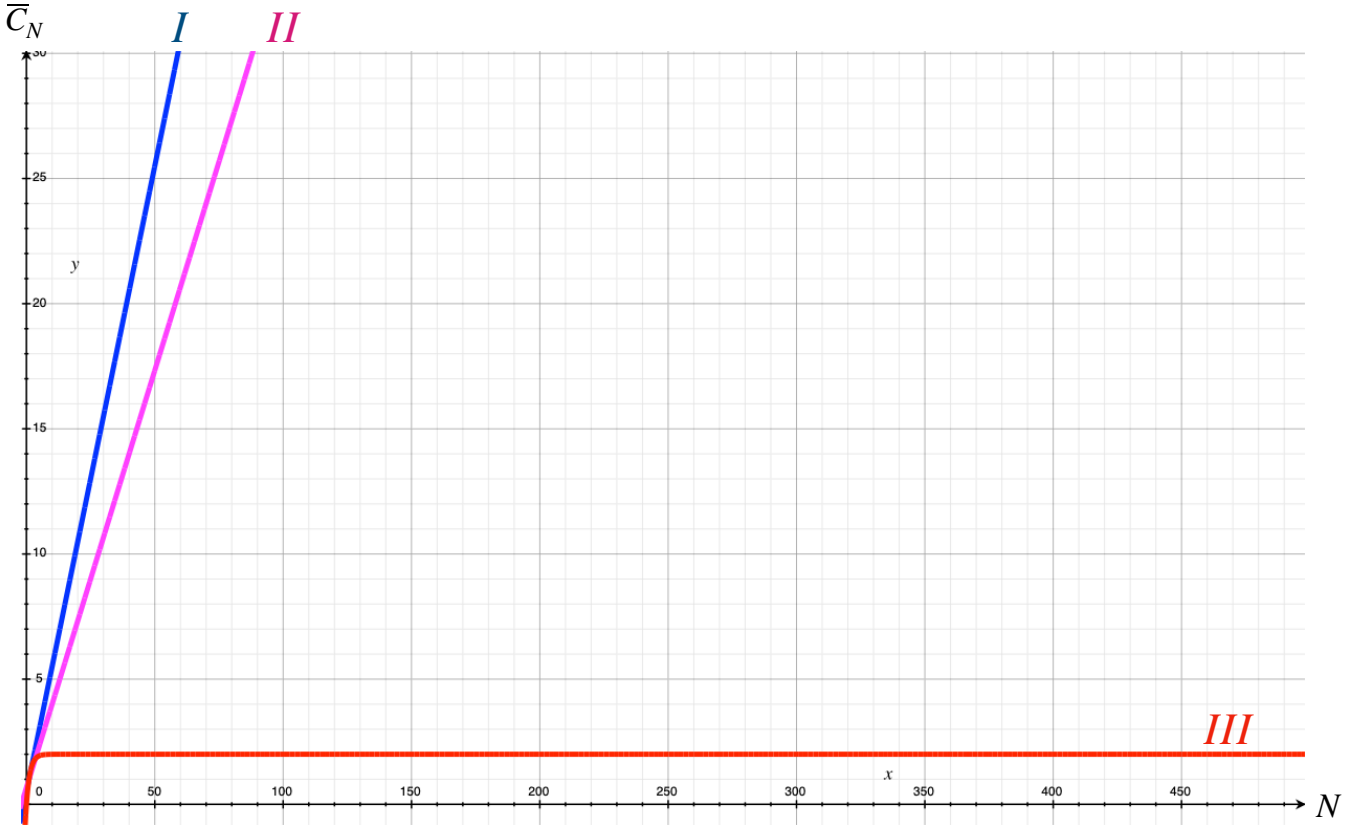
III.กำหนดให้

$$p_1 = \frac{1}{2},\ p_2 = \frac{1}{4},\ \cdots,\ p_{N-1} = \frac{1}{2^{N-1}},\ p_N = \frac{1}{2^{N-1}}$$

โอกาสที่จะค้นหาเจอใกล้บริเวณจุดเริ่มต้นสูงมาก จะได้ $\overline{C}_N = 2 - 2^{1-N}$ **เนื่องจากพจน์เอกโพแนนเชียลมากกว่า** 0 **เสมอ ดังนั้น** $\overline{C}_N < 2$



<u>กราฟ 1.4: เปรียบเทียบเวลาที่ใช้ในการทำงานเฉลี่ย (เชิงทฤษฎี) ของเงื่อนไขตัวอย่างแบบต่าง ๆ เทียบกับจำนวนข้อมูล</u>



1.7.Self-organizing list

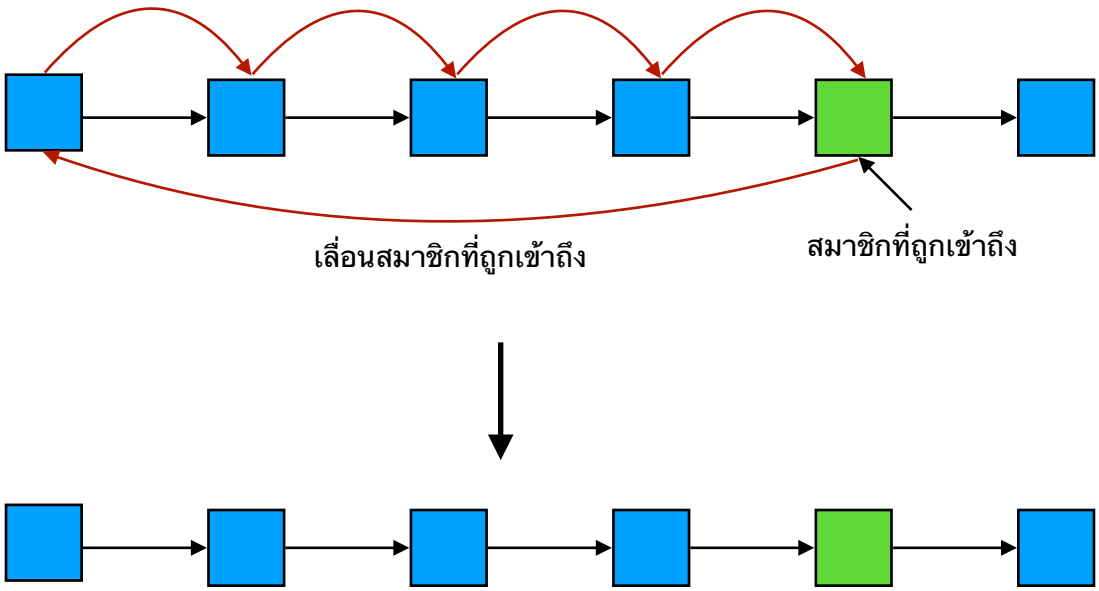
หนึ่งในวิธีที่ส่งผลดีต่อประสิทธิภาพของ Linear search คือการกระจายความน่าจะเป็นที่ดี แต่ข้อมูลส่วนใหญ่นั้นไม่ได้บอกความน่าจะเป็นโดยตรงอาจต้องใช้การนับจำนวนเองซึ่งต้องใช้หน่วยความจำเพิ่ม จึงเสนอ Self-organizing list เพื่อเลี่ยงการนับจำนวน

Self-organizing list คือลิสต์ที่ปรับลำดับใหม่จากด้วยหลายวิธี จุดมุ่งหมายเพื่อเพิ่มประสิทธิภาพของ linear search ด้วยการย้ายข้อมูลที่ถูกเข้าถึงบ่อยมายังหัวลิสต์ ซึ่งใช้เวลาค่อนข้างคงที่สำหรับ best case

1.7.1.Move to front method (MTF)

วิธีนี้จะย้าย element ที่ถูกเข้าถึงไปไว้หน้าสุดของรายการ ซึ่งจะไม่ต้องใช้หน่วยความจำเพิ่มแต่อย่างใด เมื่อเกิดขั้นตอนนี้ไปเรื่อย ๆ การกระจายความน่าจะเป็นจะเป็นไปตามความต้องการของหัวข้อที่แล้ว

วิธีการนี้ไม่เหมาะกับข้อมูลที่การเข้าถึงแต่ละครั้งแตกต่างกัน หรือเป็นข้อมูลส่วนรวมเหมาะสำหรับข้อมูลประจำบุคคล

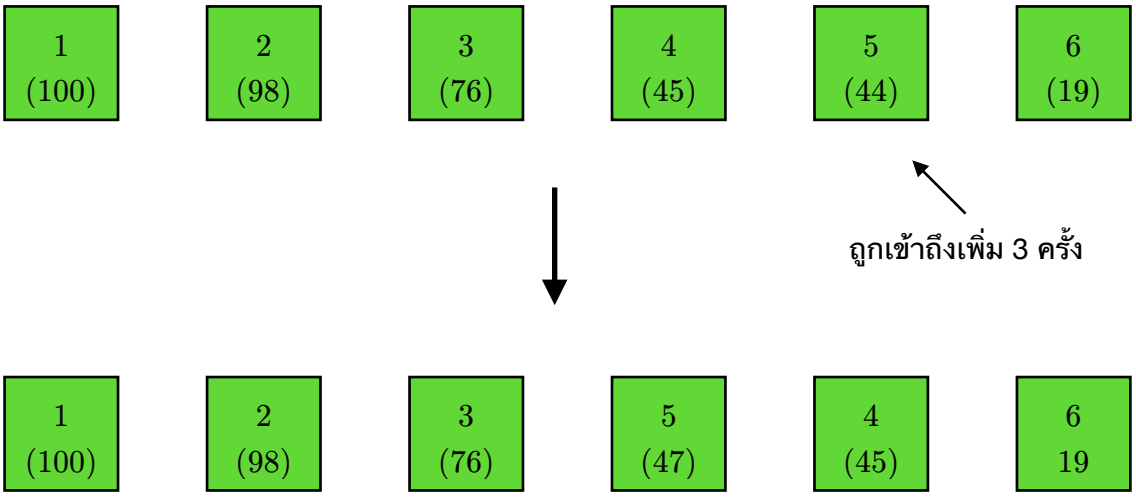



```
At the t-th item selection:
    if item i is selected:
        move item i to head of the list
```

1.7.2.Count method

วิธีการนี้ จะนับจำนวนครั้งการเข้าถึงสมาชิกตัวนั้น ๆ และจะจัดลำดับใหม่ด้วย key จำนวนครั้งการเข้าถึงจากมากไปน้อย อย่างไรก็ตามวิธีการนี้จำเป็นต้องใช้หน่วยความจำเพิ่มเติม สำหรับเก็บค่าจำนวนการเข้าถึง

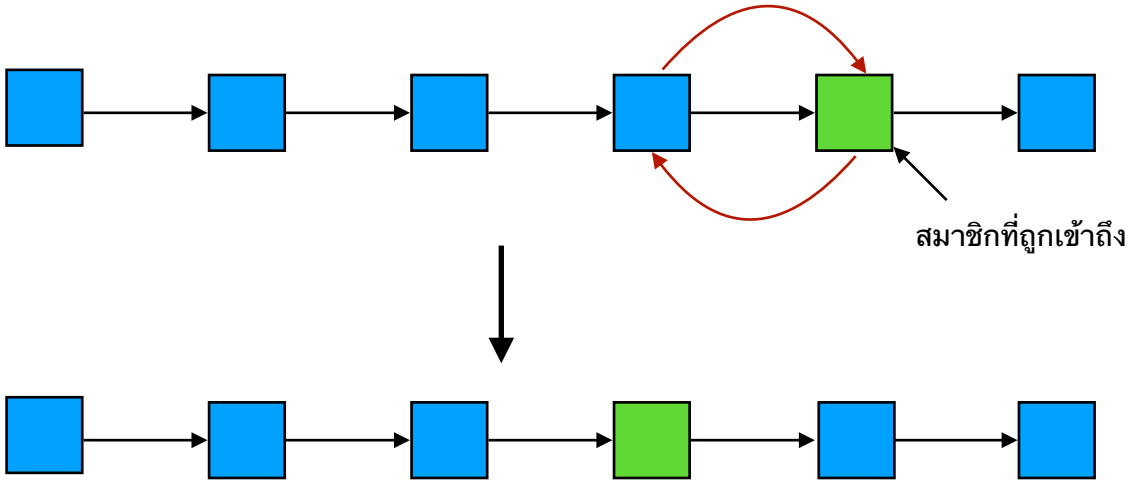
วิธีการนี้จะส่งผลเสียในกรณีที่มีข้อมูลตัวหนึ่งถูกเข้าถึงมากเกินไป ทำให้ข้อมูลตัวอื่น ๆ ไม่มีโอกาสถูกนำมาข้างหน้าเป็น Blocked organize




```
init: count(i) = 0 for each item i
     At t-th item selection:
          if item i is searched:
               count(i) = count(i) + 1
               rearrange items based on count
```

1.7.3.Transpose method

วิธีการนี้ จะกล่าวถึงการสลับสมาชิกที่ถูกเข้าถึงกับสมาชิกก่อนหน้า (ไม่รวมสมาชิกตัวแรก) เสมือนเป็นการเพิ่ม priority อัลกอริทึมนี้จะใช้งานง่ายและมีประสิทธิภาพเชิงหน่วยความจำมากกว่า Count method หากข้อมูลถูกเข้าถึงบ่อยก็จะถูกสลับมาต้นรายการเรื่อย ๆ




```
At the t-th item selection:
    if item i is selected:
        if i is not the head of list:
            swap item i with item (i - 1)
```

1.8.สรุป Linear search

Linear search เป็น search algorithm ที่มีความเข้าใจง่าย ใช้แก้ปัญหาในการค้นข้อมูลในรายการหรือโครงสร้างข้อมูลที่เป็นเส้นตรง ในส่วนต้นได้นำเสนอแนวคิดทั่วไปของอัลกอริทึมนี้พบว่ามี complexity เป็น $O(N)$ สามารถทำงานได้ดีกับข้อมูลที่มีจำนวนน้อย ในส่วนท้ายได้นำเสนอวิธีการเพิ่มประสิทธิภาพและพิจารณาถึงกรณีต่าง ๆ เช่น กรณีที่มีการเรียงข้อมูลมาแล้วซึ่งจะเป็นผลดีต่อการหาและลด complexity ลงมาเหลือเพียง O(index of key)

ความน่าจะเป็นของการค้นเจอข้อมูลที่มีการกระจายไม่เท่ากัน เมื่อนำมาพิจารณาได้หลายรูปแบบควบคู่ไปกับคณิตศาสตร์พบว่าทำให้การทำงานเร็วขึ้น ซึ่งเป็นส่วนที่สามารถไปศึกษาต่อได้อีกหลากหลาย

อย่างไรก็ตาม Linear search ยังไม่เหมาะกับการนำไปใช้งานจริงเพราะมีประสิทธิภาพการทำงานที่ต่ำกว่าอัลกอริทึมอื่น

2. 2-Dimensional closest pair algorithm

2.1.ปัญหาโดยทั่วไปและการพิจารณาสำหรับสองมิติ

ปัญหาทั่วไปเพื่อการหาจุดที่ใกล้กันที่สุดในเซตของพิกัด $(q_1, q_2, \ldots, q_N)$ สำหรับ $N \in \mathbb{N}/\{1\}$ มิติ สามารถทำได้ด้วยการหาระยะห่างแบบ Minkowski

$$\left(\sum_i |x_i - y_i|^p \right)^{\frac{1}{p}}, \; p := p_{minkowski} = 1, 2, \ldots, \infty$$

ในบทนี้จะกล่าวถึงการหาระยะห่างแบบ Minkowski ที่ $p_{minkowski} = N = 2$ โดยจุด $p_1(x_1, y_1)$ และ $p_2(x_2, y_2)$ มีระยะห่างเป็น $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ (Euclidean distance)

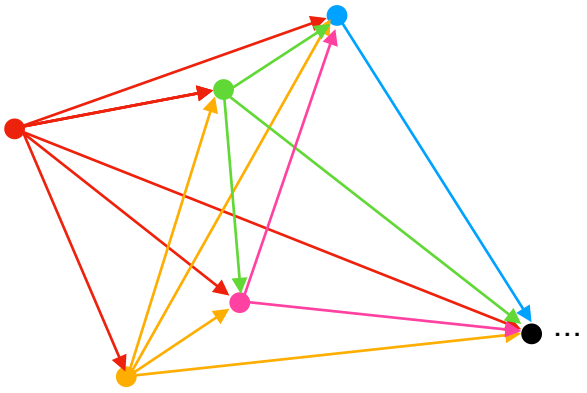
ตัวอย่างในเชิงการนำไปใช้งานอาทิเช่น การควบคุมการจราจรทางอากาศหรือทางเรือเพื่อป้องกันอุบัติเหตุ เป็นต้น

<u>หมายเหตุ</u> จุดสองจุดที่สามารถเป็นจุดเดียวกันซึ่งหมายความว่ามีระยะห่างเป็นศูนย์

2.2.Brute-force for 2-Dimensional closest pair

2.2.1.แนวคิด

เลือกจุดจากนั้นหาระยะห่างกับทุกจุดไปเรื่อย ๆ ดังภาพ



พิจารณาการทำงาน

$$(N\text{-}1) + (N\text{-}2) + (N\text{-}3) + \ldots + 2 + 1 + 0 = N^2 - \frac{N^2 + N}{2} \in \Theta(N^2)$$

หรือคำนวณจาก $\binom{N}{2}$ ได้เช่นเดียวกัน¶

2.2.2.Pseudocode และ diagram

Algorithm *D* (Brute-force 2D-Closest pair)

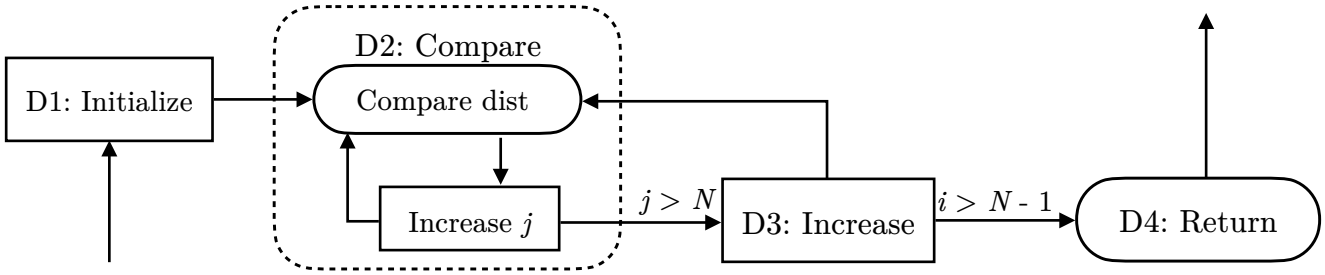
D0: **Input**: Set P contain points $p_1(x_1, y_1)$, ..., $p_N(x_N, y_N)$.
Output: Closest pair of two points.

D1: [Initialize] Set $dist \leftarrow +\infty$, $i = 1$.

D2: [Compare] Set j = i + 1
If $distance(p_i, p_j) \leq dist$, Set $dist = distance(p_i, p_j)$
Increase j by 1 and return to **D2 until** $j = N$.

D3: [Increment] Increase i by 1 and return to **D2 until** $i = N$ - 1.

D4: [Return] Return $dist$.



2.3.Divide-and-Conquer for 2-Dimensional closest pair

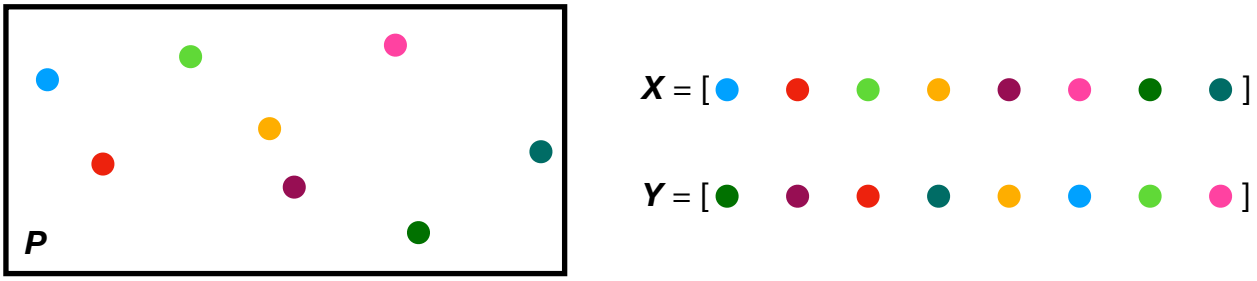
2.3.1.Motivation

หากพิจารณา running time จาก Master theorem แล้ว จะพบว่า

$$T(N) = 2T\left(\frac{N}{2}\right) + \Theta(N) \in \Theta(N \log(N))$$

และหากรวมผลจากการเรียงข้อมูลเช่น merge sort ที่ใช้เวลา $O(N \log(N))$ จะได้ Time complexity เป็น $\Omega(N \log(N))$ เท่านั้น ซึ่งเป็นที่น่าสนใจ

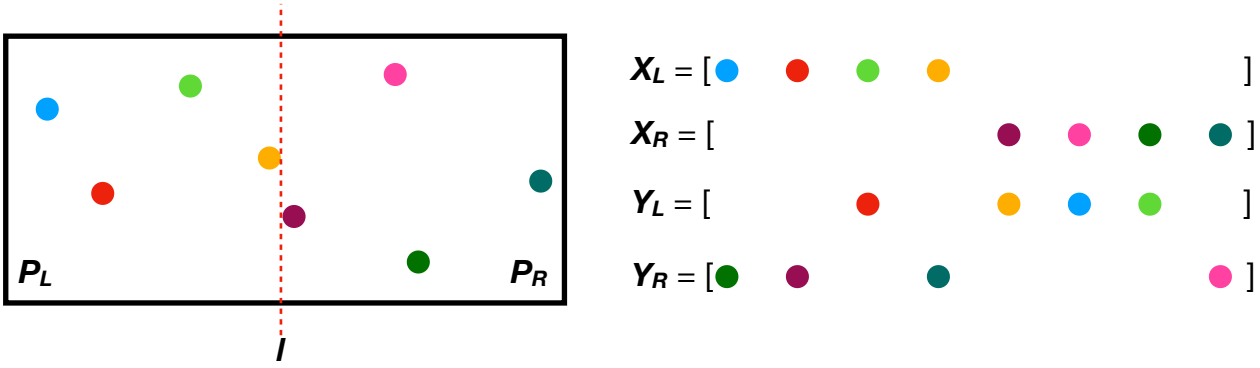
2.3.2.ออกแบบอัลกอริทึม



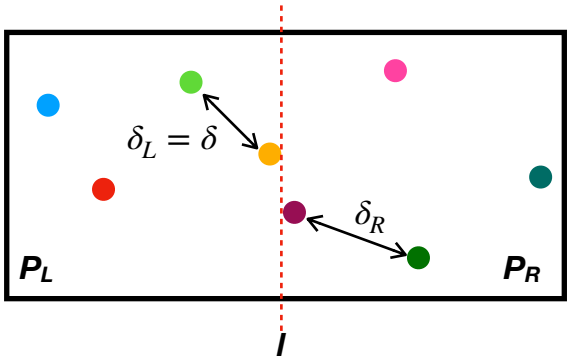
กำหนดเซตของจุดที่สนใจคือ $P \subseteq \mathbb{R}^2$ อยู่ใน X (อาเรย์ของจุดที่เรียงด้วยค่าแกน x จากน้อยไปมาก) และ Y (อาเรย์ของจุดที่เรียงด้วยค่าแกน y จากน้อยไปมาก)

สำหรับ $|P| \leq 3$ จะใช้การหาระยะห่างแบบ Brute-force **ซึ่งถือเป็น Base case และ** ถ้า $|P| > 3$ จะใช้วิธี Divide-and-Conquer ดังขั้นตอนต่อไปนี้

I. *Devide* หาเส้นแนวตั้ง l ที่แบ่งเซต P ออกแบบสองส่วน P_L และ P_R ซึ่งสอดคล้องกับเงื่อนไข $|P_L| = \lceil |P|/2 \rceil, |P_R| = \lfloor |P|/2 \rfloor$ แบ่ง X, Y เช่นเดียวกัน



II. *Conquer* *Recursive* แบ่งไปเรื่อย ๆ จนถึง *Base case* เพื่อหาระยะห่างที่สั้นที่สุด กำหนดให้ระยะห่างด้านซ้ายเป็น δ_L และด้านขวาเป็น δ_R จากนั้นนิยามให้ $\delta = \min(\delta_L, \delta_R)$

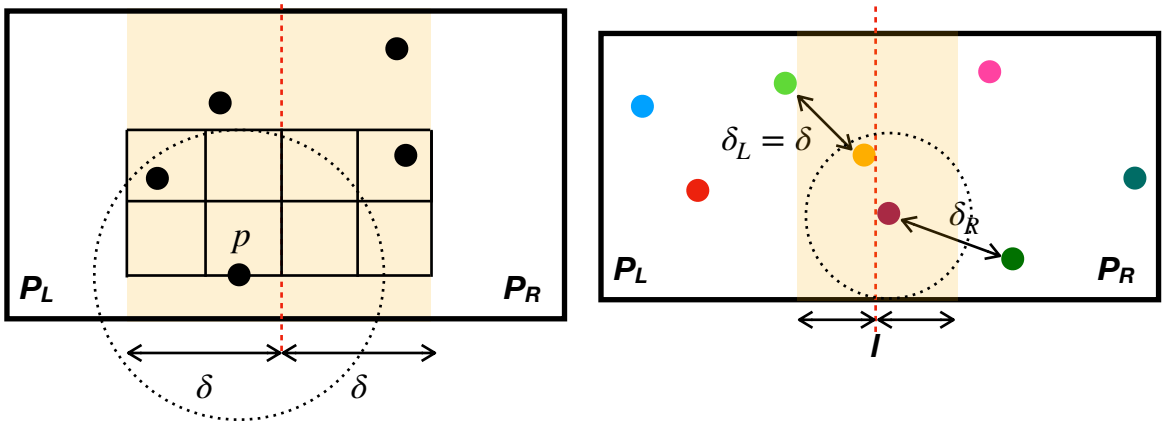


III. *Combine* ยังไม่ได้เปรียบเทียบข้ามฝั่ง? ค่าที่ยอมรับได้สำหรับระยะห่างจากเส้นแบ่ง l คือ δ เกิดเป็นแถบ 2δ หากเกินกว่านี้จะไม่ใช่ระยะที่สั้นที่สุด

i. กำหนด Y' ซึ่งทุก ๆ สมาชิกมาจาก Y $(= P_L \cup P_R)$ และอยู่ในแถบ 2δ

ii. สำหรับจุด p ใด ๆ ใน Y' วนรอบหาระยะห่าง รอบ ๆ ในรัศมี δ

Claim ซึ่งไม่เกิน 7 ช่องดังภาพซ้าย หากเริ่มจากจุดล่างสุดขึ้นไปเรื่อย ๆ จะทำงานแค่ $O(1)$ ในแต่ละจุด

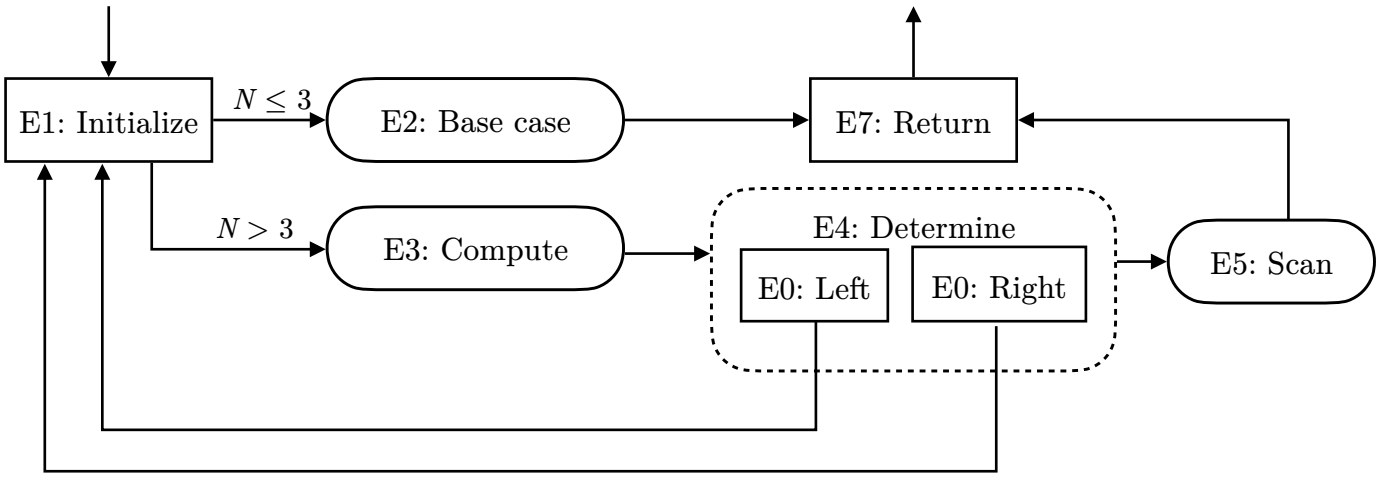


iii. แต่ละการวนรอบจะได้ระยะห่างเป็น δ' และเก็บค่าที่น้อยกว่าไปเรื่อย ๆ เมื่อครบทุกรอบจะ *return* ค่า $\min(\delta, \delta')$ เป็นระยะทางที่สั้นที่สุดใน P

2.3.3.Pseudocode และ diagram

อ้างชื่อตัวแปรมาจากหัวข้อ 2.3.2

Algorithm *E* (Devide-and-Conquer 2D-Closest pair)	
E0: **Input**: Set P. **Output**: Closest pair of two points.	
E1: [Initialize] Set X, Y	$O(1)$
E2: [Base case] **if** $\|P\| \leq 3$ **then Algorithm E**(P)	$O(1)$
E3: [Compute] Separation line l	$O(N \log(N))$
E4: [Determine] $\delta_L =$ **Algorithm E**(P_L) $\delta_R =$ **Algorithm E**(P_R) $\delta = \min(\delta_L, \delta_R)$	$2T(N/2) + O(1)$
E5: [Delete] Get Y'	$O(N \log(N))$
E6: [Scan] Find δ'	$O(N)$
E7: [Return] $\min(\delta, \delta')$	$O(1)$

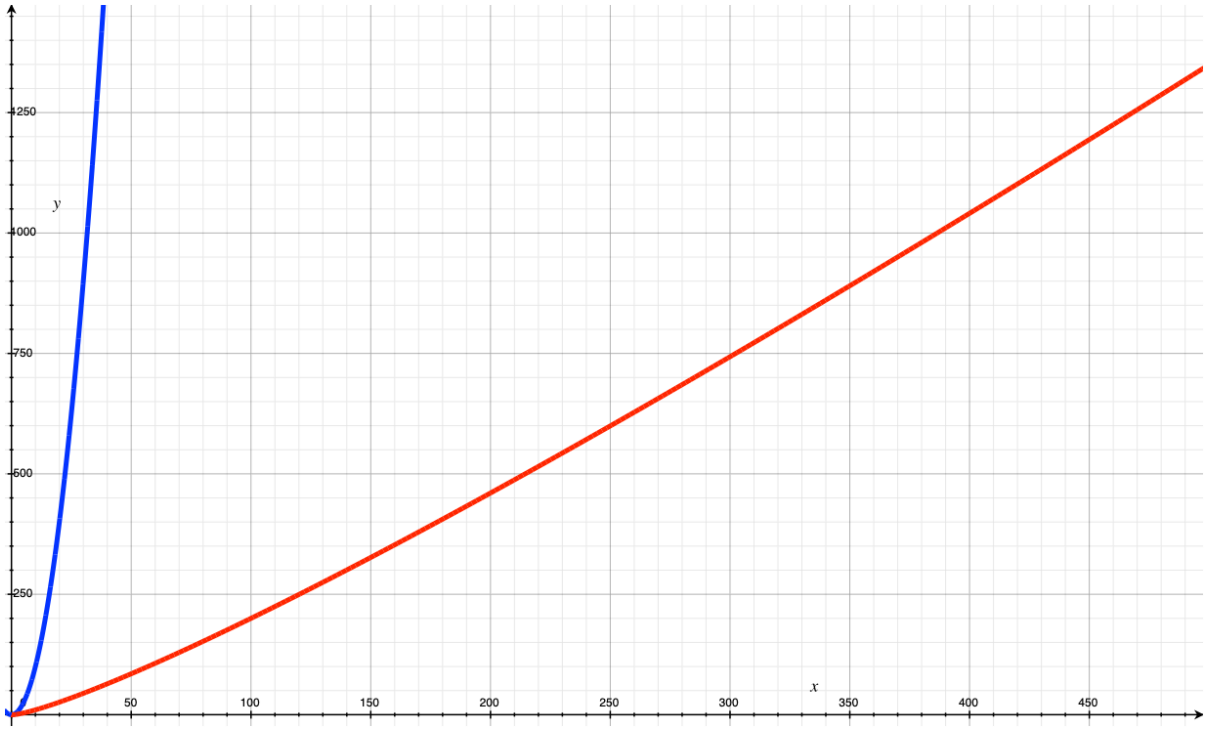


Time complexity: $T(N) = O(1) + O(1) + O(N\log(N)) + 2T(N/2) + O(1) + O(N\log(N)) + O(N) + O(1)$

$T(N) \in O(N\log(N))$

2.4.เปรียบเทียบ Time complexity ของอัลกอริทึม D และ E

กราฟ 2.1: เปรียบเทียบ Time complexity ของอัลกอริทึม D (สีน้ำเงิน) ซึ่งเป็น $O(n^2)$ และอัลกอริทึม E (สีแดง) ซึ่งเป็น $O(n\log(n))$



จำนวนข้อมูล

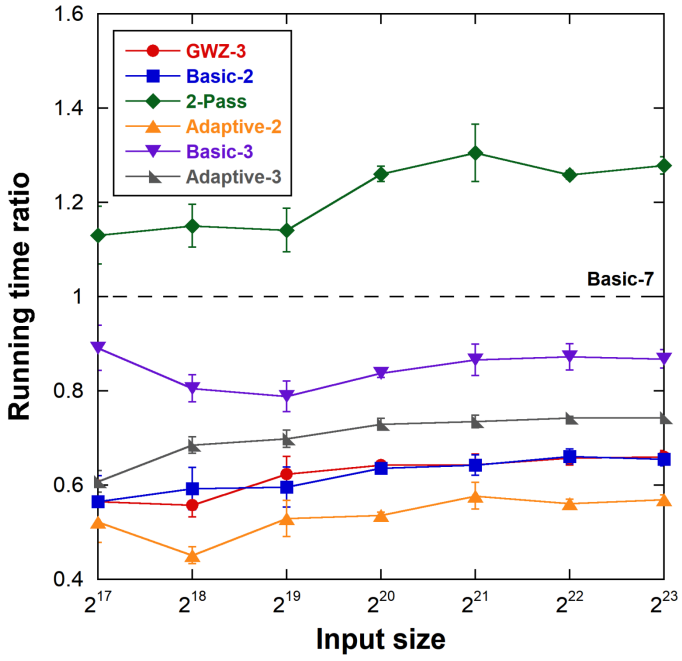
จากกราฟ อัลกอริทึม E ทำงานเร็วกว่าในทุก ๆ ระดับจำนวนข้อมูลในเชิงทฤษฎี

2.5.การลด execution time ด้วย Basic-2 Algorithm

ในบทความงานวิจัยเรื่อง **Two-Dimensional Closest Pair Problem: A Closer Look** (2020) ของ *Ovidiu Daescu* และ *Ka Yaw Teo* ได้เสนอสิ่งที่พวกเขาเรียกว่า "*Experimental results comparing the resulting improved version of the divide-and-conquer algorithm to other known variants in the literature for the planar closest-pair problem*" [2] ซึ่งได้ผลดังตาราง

Algorithm	**Number of Euclidean distances, $D(n)$**	**Time complexity**
Brute force	$\leq n^2$	$O(n^2)$
Basic-7	$\leq 7n \log n$	$O(n \log n)$
ZXZ-4	$\leq (4n/2) \log n$	$O(n \log n)$
GWZ-3	$\leq (3n/2) \log n$	$O(n \log n)$
Basic-2	$\leq 2n \log n$	$O(n \log n)$
2-Pass	$\leq 7n/2$ ($\leq 5n \log n$ c-linear distances)	$O(n \log n)$
Adaptive-2	$\leq 2n \log n$	$O(n \log n)$
QuickCP	$\leq n^2$ (Empirical: $O(n \log n)$)	$O(n^2)$
Pair-pruned CP	$\leq n^2$ (Empirical: $O(1)$)	$O(n^2)$
Basic-3	$\leq 3n \log n$	$O(n \log n)$
Adaptive-3	$\leq 3n \log n$	$O(n \log n)$

ประกอบกับผลเชิงการทดลอง



จากข้อมูล อัลกอริทึม Basic-2 มีประสิทธิภาพการทำงานที่ดีและสามารถทำความเข้าใจได้ง่าย ผู้จัดทำจึงเสนอวิธีการคิดของสองอัลกอริทึมนี้ เพื่อเป็นแนวทางในการต่อยอด

2.5.1.Basic-2 Algorithm

Peraira และ *Lobo* ได้เสนอวิธีการเพิ่มประสิทธิภาพด้วยการปรับเปลี่ยนขั้นตอน combine ในหัวข้อ 2.3.2 โดยเริ่มจากการพิจารณา Y' ว่าทั้งฝั่งซ้าย (Y_L') และขวา (Y_R') ของเส้นแบ่งจะต้องไม่เป็นพื้นที่ว่าง เลือกจุด p เป็น Y' ตัวแรก โดยทั้งนี้จะสมมุติให้ $p \in Y_L'$ เพื่อเป็นกรณีทั่วไปในการศึกษา จากนั้นจะคำนวณระยะห่างจากจุด p กับสองจุดแรกใน Y_R' สุดท้ายกำหนดใหม่ให้ $Y_L' = Y_L' \backslash \{p\}$ ซึ่งมีอย่างมาก 2 ระยะทางจะถูกคำนวณขึ้นสำหรับจุด p ใน Y_L'และ Y_R'[3]

I. Pseudocode สำหรับ combine

Algorithm Basic-2(combine)

0: **Input**: δ, Y_L' and Y_R'
Output: Closest pair of two points.

1: **Set** *left* $\leftarrow$ $Y_L'[0]$ and *right* $\leftarrow$ $Y_R'[0]$.

2: **While** there are points in Y_L' and Y_R' **do**

3: $dist \leftarrow$ distance(*left*, *right*)

4: **if** $dist < \delta$ **then** *closest* $\leftarrow$ (*left*, *right*) and $\delta = dist$

5: **if** $left.y \leq right.y$ **then**

6: **if** there is at least one more point in Y_R' **then**

7: $dist \leftarrow$ distance(*left*, next point of *right*)

8: **if** $dist < \delta$ **then** *closest* $\leftarrow$ (*left*, next point of *right*), $\delta = dist$

9: *left* $\leftarrow$ next point of *left*

6: **else if** there is at least one more point in Y_L' **then**

7: $dist \leftarrow$ distance(*right*, next point of *left*)

8: **if** $dist < \delta$ **then** *closest* $\leftarrow$ (next point of *left*, *right*), $\delta = dist$

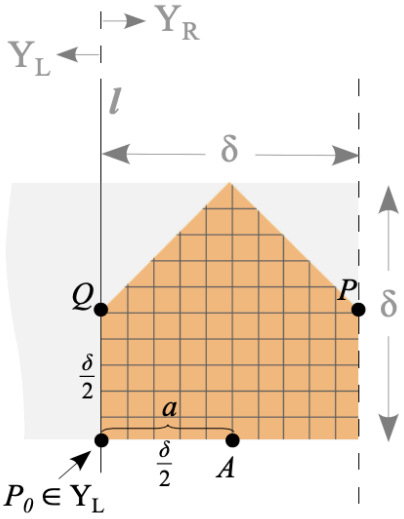
9: *right* $\leftarrow$ next point of *right*

10: **Return** δ

II. พิสูจน์ความถูกต้อง

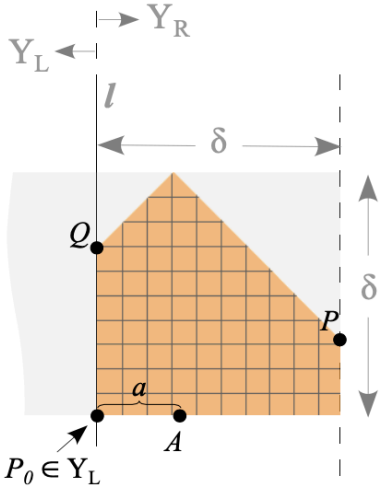
เริ่มพิสูจน์ด้วยการหาระยะทางแบบ Minkowski สำหรับ $p = 1$ โดยนิยามให้ $d_1(A, B) = |x_A - x_B| + |y_A - y_B|$ และ $p_0 = (0, 0) \in Y_L$, $Y_0 \subseteq Y_R{}'$ ให้ $A = (a, b)$, $0 \leq a \leq \delta$, $b \geq 0$ เป็นจุดแรกใน Y_0 เริ่มพิจารณาจากกรณี $b = 0$ ส่วนกรณีอื่น ๆ คือ $b > 0$ จะพิจารณาด้วยการเคลื่อนแกน ซึ่งตำแหน่งสัมพัทธ์ของแต่ละจุดจะไม่เปลี่ยน จึงให้ $A = (a, 0)$ และ $P = (x, y)$, $0 \leq x \leq \delta$, $y \geq 0$ เป็นจุดใด ๆ ใน Y_0 จะพิจารณาได้สามกรณีดังนี้

i. $a = \delta/2$

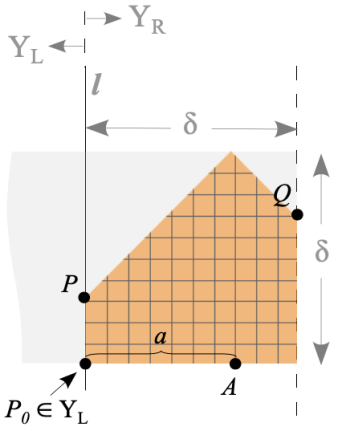


ตำแหน่งที่เป็นไปได้ของทั้งสามจุดแรกใน Y_0 นั้นพบว่า A และ Q เป็นจุดที่ใกล้ที่สุดจาก p_0

ii. $0 \leq a < \delta/2$ แยกพิจารณาสองกรณี $y > \delta/2$ และ $y \leq \delta/2$ พบว่าเมื่อพิสูจน์แล้ว $y \leq \delta/2$ เกิดข้อขัดแย้ง จึงนำ $y > \delta/2$ มาสรุป ได้ว่า A เป็นจุดที่ใกล้ที่สุดจาก p_0

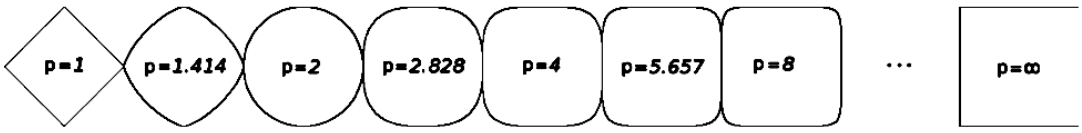


iii. $\delta/2 < a \leq \delta$ แยกพิจารณาสองกรณี $x \geq a$ และ $x < a$ พบว่าเมื่อพิสูจน์ $x < a$ ควบคู่กับ $x \geq a$ ทำให้ $x < a$ เกิดข้อขัดแย้ง จึงนำ $x \geq a$ มาสรุป ได้ว่า A เป็นจุดที่ใกล้ที่สุดจาก p_0



ทั้งสามกรณีที่กล่าวมาพบว่าจุดที่ใกล้ที่สุดของ p_0 เป็นเพียงสองจุดแรกใน Y_0 เท่านั้น ซึ่งเป็นการพิสูจน์ด้วย Minkowski distance สำหรับ $p = 1$

เพื่อขยายแนวคิดสำหรับ Minkowski distance สำหรับ $p > 1$ *Jiang* และ *Gillespie* ได้ให้ความเห็นว่า "This ordering means that the sparsity effect within the slab will be similar, but somewhat stronger, for larger values of p. Therefore, the precedent analysis of $p = 1$ not only remains valid for $p > 1$ but, in a sense, the corresponding geometric relations between elements of Y_0 are expected to be more 'tight' for all other Minkowski distances."



2.6.สรุป 2-Dimensional closest pair algorithm

ใช้แก้ปัญหาการหาคู่ของข้อมูลที่มีระยะห่างใกล้กันมากที่สุด ในโครงสร้างข้อมูลแบบ array มีค่าบงชี้ต่ำแหน่งสองมิติ ส่วนต้นได้นำเสนอแนวคิดเบื้องต้น พบว่าจะสามารถทำได้โดยง่ายด้วยการ brute force จะได้ complexity เป็น $O(N^2)$ ซึ่งสามารถทำงานได้ดีกับข้อมูลจำนวนน้อย ในส่วนท้ายได้นำเสนอวิธีการเพิ่มประสิทธิภาพของ algorithm ด้วยการใช้วิธี divide and conquer ซึ่งจะได้ complexity เป็น $O(N \log N)$ และการลด execution time ด้วย basic-2 algorithm

2-D Closest Pair เป็น algorithm ที่มีประสิทธิภาพ สามารถนำไปประยุกต์ใช้ได้หลากหลายในด้านเทคนิค สามารถศึกษา พัฒนาต่อและประยุกต์ใช้ได้อีกมาก

3. Unique element algorithm

การที่มีชุดข้อมูลที่ซ้ำกันจำเป็นต้องมีการคัดกรองเพื่อลดการใช้พื้นที่ในหน่วยความจำ การสร้างอัลกอริทึมที่เข้าใจง่ายและมีประสิทธิภาพจึงเป็นสิ่งที่ควรศึกษา

3.1.Unique element algorithm ทั่วไป

3.1.1.Pseudocode และ Diagram

Algorithm *F* (Ordinary Unique element algorithm)

F0: **Input**: Array A[0, …,N - 1]
Output: "True" if all the elements in A are distinct, otherwise "False"

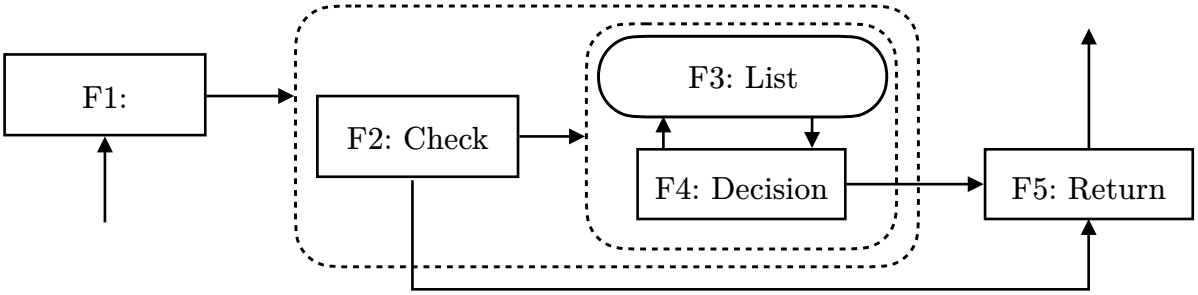
F1: [Initialize] Set $i \leftarrow 0$, $j \leftarrow i + 1$

F2: [Check] **for** $i \leftarrow 0$ to N - 2 **do**

F3: [List] **for** $j \leftarrow i + 1$ to N - 1 **do**

F4: [Decision] **if** $A[i] = A[j]$, **Return** False

F5: [End] **Return** True



3.1.2.Complexity

Operation	Cost
F1: [Initialize]	2
F2: [Check]	$i + 1 - 0 + 1 = i + 2$
F3: [List]	$N - 1 - i - 1 + 1 = N - 1 - i$
F4: [Decision] + F5: [End]	1 + 1

- $C_{worst}(N) = \sum_{i=1}^{N-2} (N - 1 - i) = \frac{(N-1)N}{2} \in O(N^2)$
- **Best case เกิดเมื่อ** $A[0] = A[1]$ **หรือ** $|A| = 1$, $C_{best} \in \Omega(1)$

3.2.Unique element algorithm แบบ Transform-and-Conquer

สืบเนื่องจากหัวข้อที่ 1.5 เกี่ยวกับข้อมูลที่เรียงลำดับแล้ว ทำให้การกระจายตัวของข้อมูลเป็นระบบมากขึ้น ในการแก้ปัญหา element uniqueness หากข้อมูลที่เหมือนกันวางตัวอยู่ติดกันก็จะสามารถหาข้อมูลที่ซ้ำกันได้ง่ายขึ้นเพียงการวนรอบในรอบถัดไป[4]

3.2.1.Pseudocode

Algorithm ***G*** (Presort Unique element algorithm)

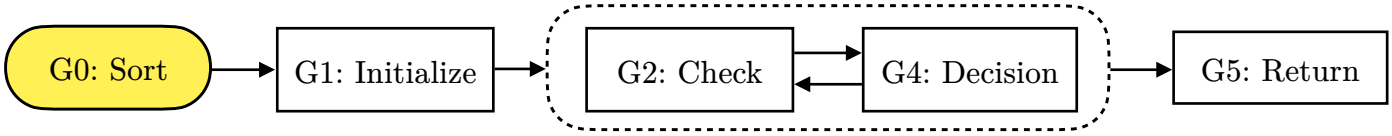
G0: **Input**: Array A[0, ...,N - 1] with $A[i] \leq A[i+1]$
Output: "True" if all the elements in A are distinct, otherwise "False"

G1: [Initialize] Set $i \leftarrow 0$

G2: [Check] **for** $i \leftarrow 0$ to N - 2 **do**

G3: [Decision] **if** $A[i] = A[i+1]$, **Return** False

G4: [End] **Return** True



3.2.2.Complexity

Operation	Cost	หมายเหตุ
G0:	$\Theta(N \log(N))$	Sort
G1: [Initialize]	1	
G2: [Check]	N - 2	
G3: [Decision]	S	$S :=$ (success, unsuccess) = (1, 0)
G4: [End]	1 - S	

- $T(N) = T_{G_0}(N) + T_{G_{>0}}(N) \in \Theta(N \log(N)) + \Theta(N) + \Theta(1) = \Theta(N \log(N))$
- **Best case เกิดเมื่อ** $A[0] = A[1]$ **หรือ** $|A| = 1$, $C_{best} \in \Omega(1) + \Omega(N \log(N))$ **เพราะจำเป็นต้องเรียงลำดับก่อนเสมอ**

Remark **สำหรับ** $x_1, \ldots, x_N \in \mathbb{R}$ **สามารถคำนวณได้จาก** $\prod_{i \neq j} (x_i - x_j)$ **ซึ่งใช้เวลา** $O(N \log(N))$ **เช่นกัน (ขั้นตอนการคำนวณอยู่ในหัวข้อ** 3.3.2 Claim 3)[5]

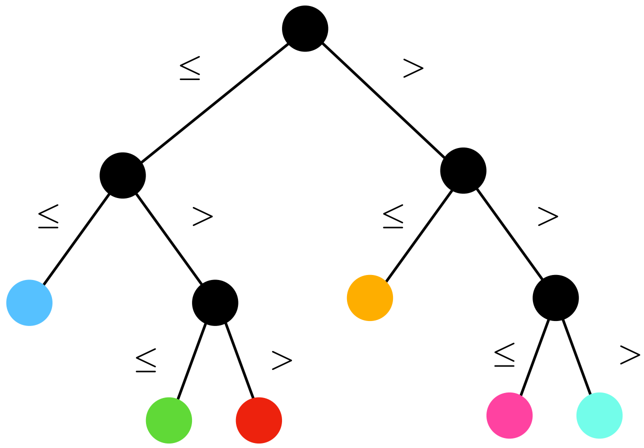
3.3.มีอัลกอริทึมสำหรับปัญหานี้ที่สามารถทำงานได้เร็วกว่า $\Theta(N \log(N))$ หรือไม่?

ในหัวข้อนี้จะกล่าวถึงการหา Lower-bound ของปัญหา ซึ่งเป็นการบอกถึง complexity ขั้นต่ำที่จำเป็นต้องใช้

3.3.1.Element uniqueness problem

Element uniqueness problem เป็นการตรวจสอบว่าข้อมูลในอาร์เรย์ N ตัว มีข้อมูลที่ซ้ำกันหรือไม่ (ในที่นี้ให้ตอบ Ture หากไม่ซ้ำและ False หากซ้ำ)

แนวคิดที่ใช้ในการพิสูจน์จะใช้ decision tree model ที่มี leave ทั้ง $N!$ เป็น True หรือ False โดยเส้นทางจะเริ่มจาก root



3.3.2.บทพิสูจน์

Claim 1: $S(L)$ เป็นเอกภพสัมพัทธ์ของ leave ซึ่งเป็นคำตอบทั้งหมดจะเชื่อมต่อถึงกัน

<u>พิสูจน์</u> ให้ $x = (x_1, ..., x_n)$ และ $y = (y_1, ..., y_n)$ เป็นสองจุดใน $S(L)$

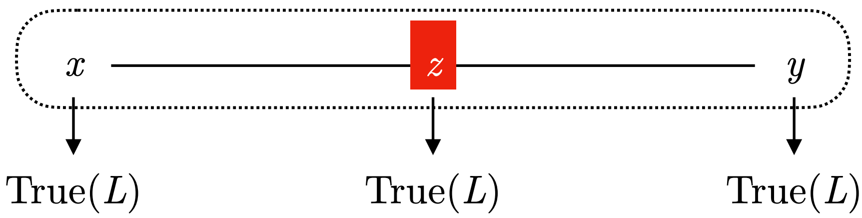
สมมุติ เริ่มจาก x จะไปถึง leave L, เริ่มจาก y จะไปถึง leave L

เริ่มจาก z, z นิยามเป็น $z_1, ..., z_n$ เมื่อ $z_i = \lambda x_i + (1 - \lambda)y_i,$

λ เป็นค่าตั้งแต่ 0 ถึง 1 ดังนั้น z จะอยู่บนเส้นเชื่อมระหว่าง x และ y หากไปยังเส้นทาง '$\leq$' โดยเริ่มจาก x ดังนั้น y จะใช้เส้นทางนี้เหมือนกัน เนื่องจากไปยัง L เหมือนกัน

- $x_i < x_j$; $y_i < y_j$
- $\lambda x_i + (1 - \lambda)y_i < \lambda x_j + (1 - \lambda)y_j \rightarrow z_i < z_j$

z ใช้เส้นทางเดียวกัน แสดงว่าใน $S(L)$ เชื่อมต่อถึงกัน ($S(L)$ is convex)

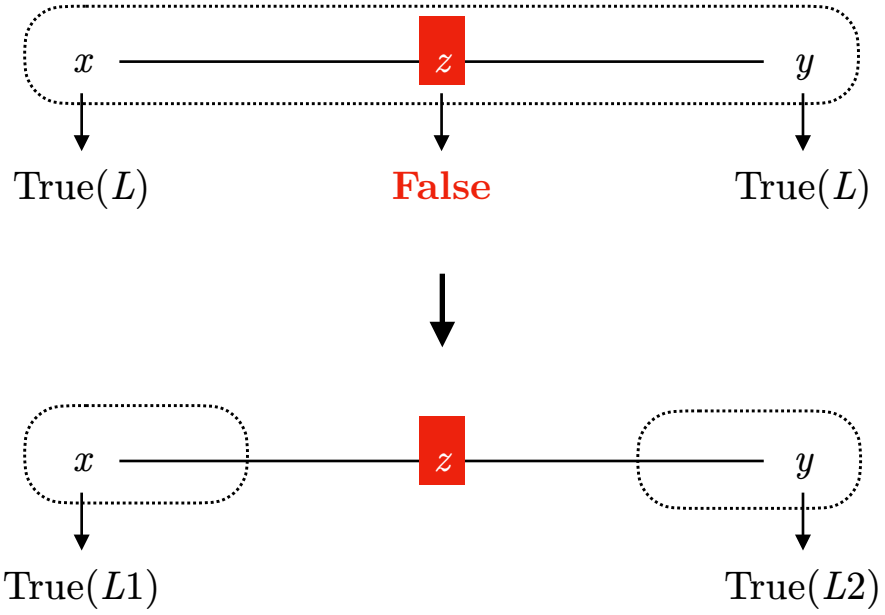


Claim 2: ให้ $x = (x_1, ..., x_n)$ เป็นลิสต์ของ element แต่ละตัวและ $y = (y_1, ..., y_n)$ เป็น permutation ของ x โดยทั้ง x และ y ไปยัง leave True ได้

<u>พิสูจน์</u> ให้ x และ y เก็บค่าที่ต่างกัน และ y เป็น permutation ของ x โดยทั้งสองไปยัง leave True ที่ต่างกัน

ในหัวข้อนี้ จะพิสูจน์โดยใช้ข้อขัดแย้งโดยจะแสดงว่ามี z ไปยัง False

- ถ้า z ไปยัง False แสดงว่ามี $x_i < x_j$ และ $y_i > y_j$
- ถ้าเงื่อนไขบนเป็นจริง สามารถนิยาม $f(w) = w_i\text{-}w_j$; $f(x) < 0$, $f(y) > 0$ เช่น $x = [1, 2, 3]$, $y = [3, 1, 2]$
- จาก mean value theorem จะมีเส้นทางที่ $f(z) = 0$ หรือ $z_i = z_j$ นั่นหมายถึง จะไปถึง False ได้



Claim 3: จำนวนของ leave True $\geq n!$

<u>พิสูจน์</u> จำนวนของ permutation ของอาร์เรย์ขนาด n จะเป็น $n!$

จำนวนของ leave True เป็น $2^h \geq n!$ ที่ worst case, h คือความสูงต้นไม้

ดังนั้น $h \geq \log(n!) = \log[(n)(n - 1)...(2)(1)] = \log(n^n) = n \log(n)$: lower-bound (ตัดพจน์อื่น ๆ ออกนอกจาก n)

พิจารณา n/c จำนวน: $n! \geq (n/c)^{(n/c)} \rightarrow \log(n!) \geq (n/c)(\log(n) - 1)$

จาก $h \geq (n/c)(\log(n) - 1)$ จะได้ $h \geq C\log(n)$

ทำให้ $T(n) \geq C\log(n) \rightarrow T(n) \in \Omega(n \log(n))$

3.3.3.สรุป lower-bound

จากการพิสูจน์พบว่าคำตอบใน $S(L)$ จะเชื่อมต่อไปยัง True ที่ต่างกันทั้งหมด มีจำนวนไม่ต่ำกว่าการ permutation ของสมาชิกในอาร์เรย์ และสามารถสรุปได้ว่า lower-bound ของปัญหานี้คือ $\Omega(N \log(N))$

อ้างอิงจากบทความงานวิจัยของ Michael Ben-Or เรื่อง Lower Bounds For Algebraic Computation Trees พบว่า "***Theorem 1.*** *Any algebraic computation tree that solves the n-element distinctness problem must have complexity of at least Ω(n log n).*" เช่นกัน[5]

3.4.สรุป Unique element algorithm

Unique element algorithm สามารถเข้าใจได้ง่าย ไม่ค่อยมีความซับซ้อน ถ้าตรวจสอบแบบปกติโดยไม่ได้เรียงลำดับ time complexity จะเป็น $O(N^2)$ แต่ถ้ามีการเรียงลำดับก่อนจะมี time complexity เป็น $\Theta(N \log(N))$

Lower-bound สามารถพิสูจน์โดยใช้ decision tree model ได้ค่า $\Omega(N \log(N))$

หากต้องการนำอัลกอริทึมนี้ไปใช้ ควรทำการเรียงลำดับข้อมูลก่อน เพื่อเพิ่มประสิทธิภาพ

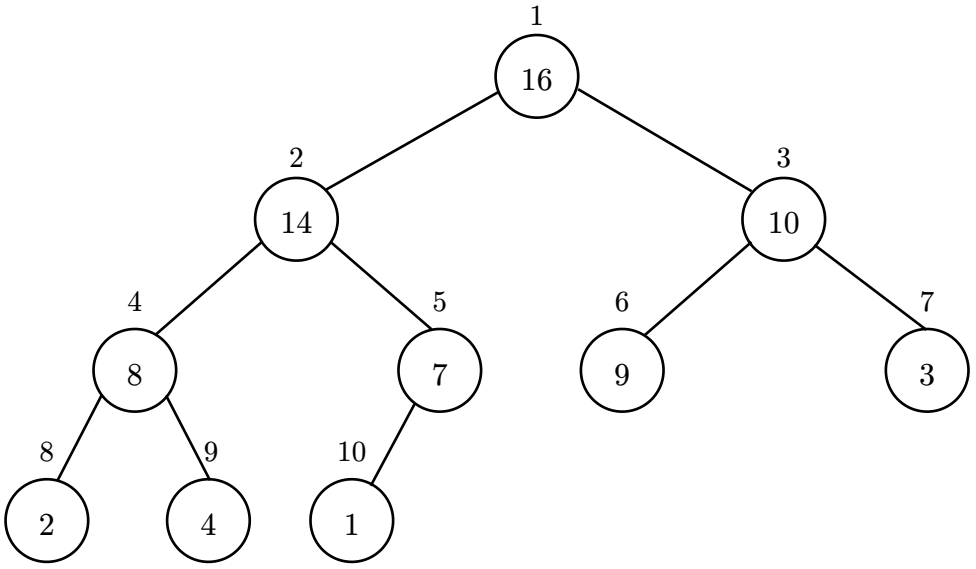
4. Heapsort

4.1.Sort algorithm

สืบเนื่องจากหัวข้อที่ 1. การจัดการข้อมูลเพื่อการนำกลับมาใช้ได้เร็วขึ้น ที่กล่าวถึงการค้นหาข้อมูลว่าเป็นสิ่งสำคัญ อีกหนึ่งวิธีที่สำคัญคือการเรียงลำดับข้อมูล เพื่อประโยชน์ในการเรียกใช้หรือประยุกต์ใช้ในอัลกอริทึมอื่น ๆ อาทิหมวด Transform-and-Conquer

4.2.Heaps

พิจารณาอาร์เรย์ที่มีสมาชิก $K_1,\ K_2,\ ...,\ K_N$ จะเป็น *heap* เมื่อ $K_{\lfloor j/2 \rfloor} \geq K_j$ สำหรับ $1 \leq \lfloor j/2 \rfloor < j \leq N$ นั่นหมายถึง $K_1 = \max(K_1, K_2, \ \ldots, K_N)$ ดังตัวอย่าง (max-heap)[1]



1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1

4.3.การคงไว้ซึ่งความเป็น Heap

วิธีการที่เรียกว่า Max-heapify ใช้สำหรับการทำให้ binary tree ที่อาจละเมิดเงื่อนไขความเป็น heap

พิจารณาอาร์เรย์ A และ index i , สมมุติว่ามี binary tree ที่ root เป็น $\text{Left}(i) = 2i$ และ $\text{Right}(i) = 2i + 1$ เป็น Max-heap แต่กระนั้น $A[i]$ อาจน้อยกว่าโหนดลูก ซึ่งละเมิดเงื่อนไขความเป็น Heap ในกรณีนี้ Max-heapify จะทำให้ค่าที่ $A[i]$ “เลื่อนลง” (ถอดภาษามาจาก sift down) ทำให้ subtree root ที่ i เป็นไปตามเงื่อนไขความเป็น heap

4.3.1.Pseudocode และ ตัวอย่าง

Algorithm *H* (Max-Heapify)

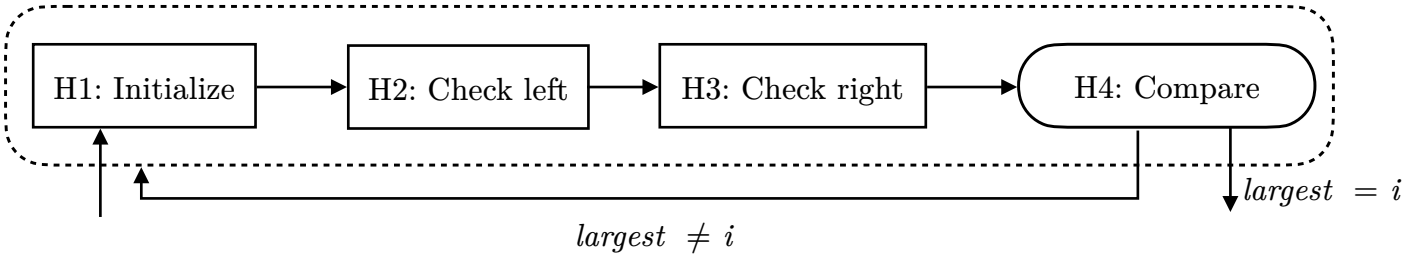
H0: **Input**: Array A with some index i

H1: [Initialize] Set $l \leftarrow \text{Left}(i)$, $r \leftarrow \text{Right}(i)$

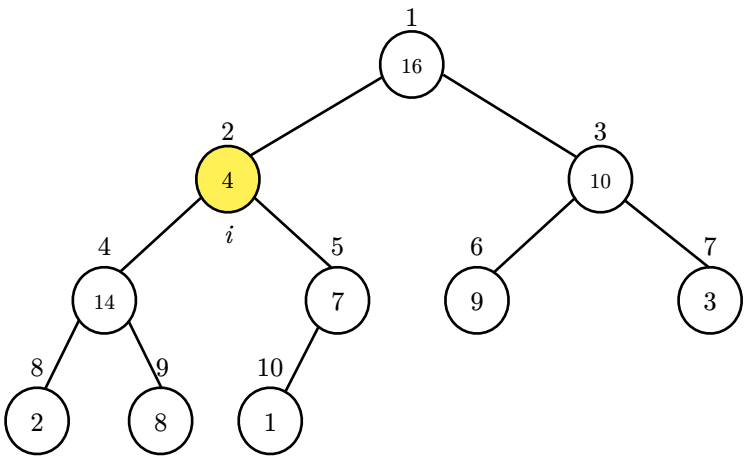
H2: [Check left] **if** $l \leq A$.heap-size **and** $A[l] > A[i]$ **then** $largest \leftarrow l$
else $largest \leftarrow i$

H3: [Check right] **if** $r \leq A$.heap-size **and** $A[r] > A[largest]$ **then** $largest \leftarrow r$

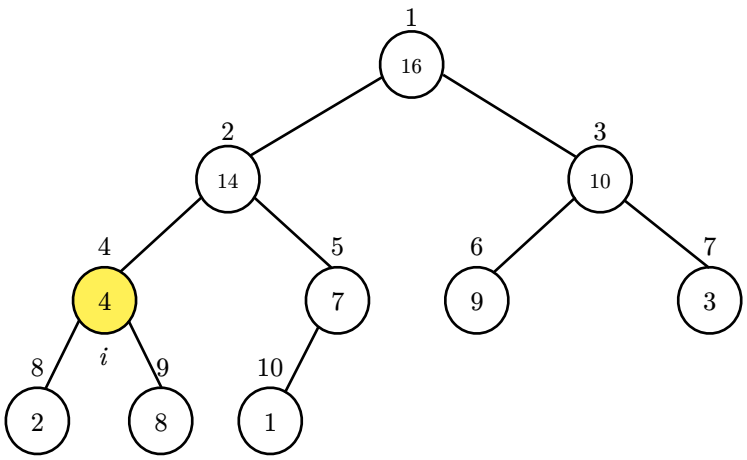
H4: [Compare] **if** $largest \neq i$
exchage($A[I]$, $A[largest]$), Max-Heapify(A, $largest$)



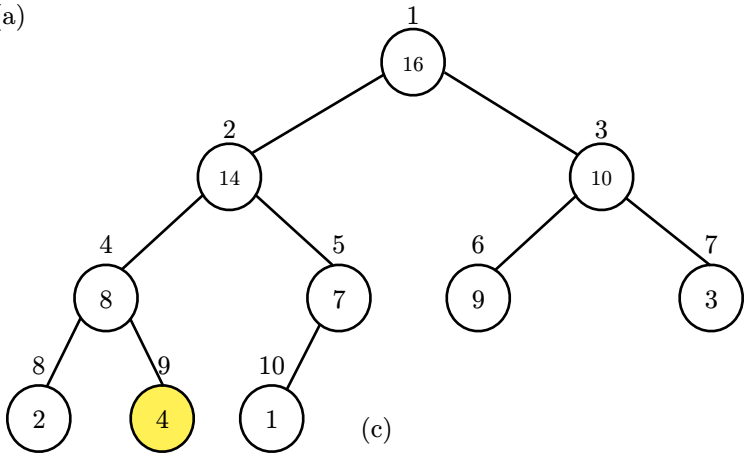
ตัวอย่าง Max-Heapify(A, 2)



(a)



(b)



(c)

4.3.2.Running time

running time ของ Max-Heapify ของ subtree ขนาด N ที่มี root เป็นโหนด i คือ

$$\begin{aligned} T(N) = & \quad T_{\text{หาความสัมพันธ์ของตัวแปร } A[i],\ A[l],\ A[r]} + \text{T}_{\text{รัน Max-Heapify ของ subtree}}(\text{เกิด recursive}) \\ \leq & \quad \Theta(1) + T\left(\frac{2N}{3}\right) \end{aligned}$$

จาก Master theorem $f(n) = \Theta(n^{\log_b a+\epsilon}) \rightarrow T(n) = \Theta(n^{\log_b a} \log n)$ [6]

จะได้ $T(N) = O\log(N)$

4.4.สร้าง Heap อย่างไร

สามารถใช้ Max-Heapify ในการ “Bottom-up” เพื่อแปลงอาร์เรย์ $A[1...N]$ เมื่อ N เป็นขนาดของอาร์เรย์ ไปเป็น max-heap และยังทราบอีกว่าอาร์เรย์ย่อย $A[(\lfloor N/2 \rfloor + 1)...N]$ เป็น leave ของต้นไม้ใหญ่ทั้งหมด ดังนั้นแต่ละโหนดจะเริ่มด้วย heap ที่มีสมาชิกตัวเดียว

Build-Max-Heap จะกระทำด้วยโหนดที่เหลือและใช้ Max-Heapify ในแต่ละโหนด

4.4.1.Pseudocode และ ตัวอย่าง (Obvious process)

Algorithm *I* (Build-Max-Heap)

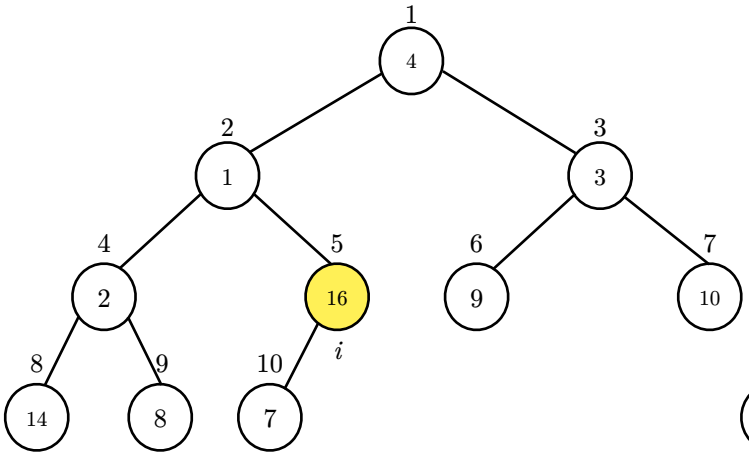
I0: **Input**: Array A with some index i

I1: [Initialize] Set $A.heap\text{-}size \leftarrow A.lenght$

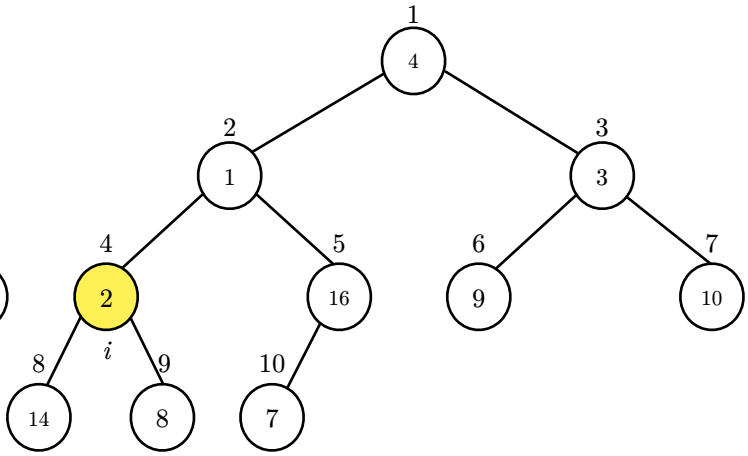
I2: [Max-Heapify] **for** $i = \lfloor$A.length$/2\rfloor$ **downto** 1 **do** Max-Heapify(A, i)

ตัวอย่าง

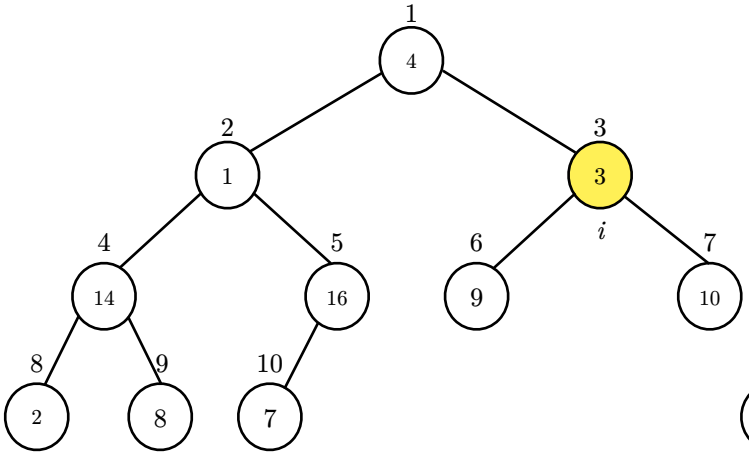
A	4	1	3	2	16	9	10	14	8	7



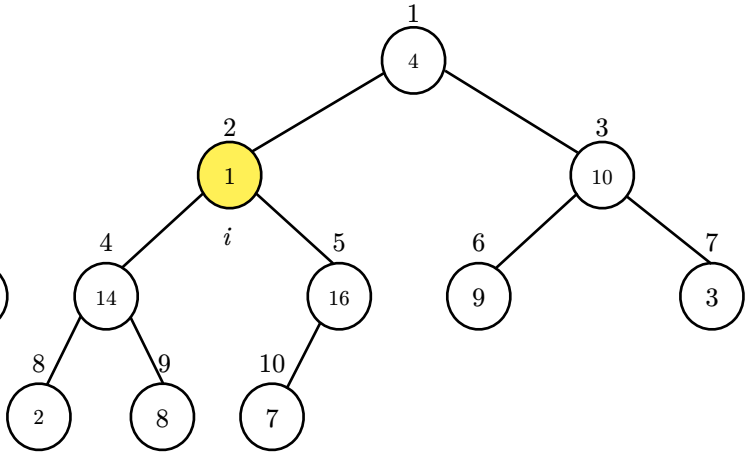
(a)



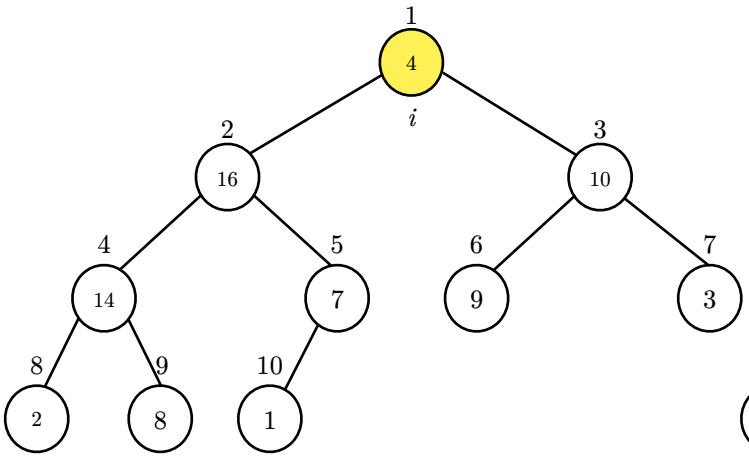
(b)



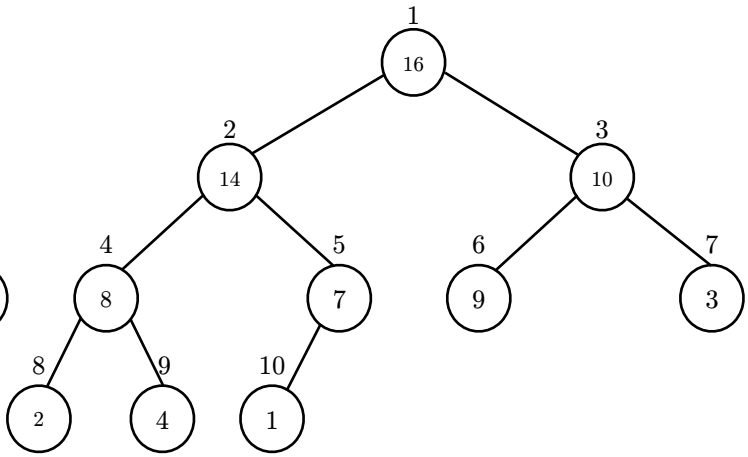
(c)



(d)



(e)



(f)

4.4.2.Running time

แต่ละรอบของ Max-Heapify จะมี cost $O(\log(N))$ time และ Build-Max-Heap มี cost $O(N)$ จึงมี running time เป็น $O(N \log(N))$ เป็น upper-bound อย่างกว้าง

สามารถวิเคราะห์ขอบเขตที่แคบลงโดยพิจารณาจากเวลาสำหรับ Max-Heapify ที่ดำเนินการกับโหนดซึ่งขึ้นกับความสูงของโหนดนั้น ๆ ในต้นไม้ และความสูงส่วนใหญ่จะต่ำ (โหนดลูกเป็นส่วนใหญ่)

การวิเคราะห์ขอบเขตขึ้นกับสมบัติ: heap ที่มี N สมาชิกมีความสูง $\lfloor \log N \rfloor$ และมีอย่างมาก $\lceil N/2^{h+1} \rceil$ โหนดสำหรับความสูง h (พิสูจน์ที่มาในหัวข้อถัดไป)

เวลาที่ใช้สำหรับ Max-Heapify เมื่อดำเนินการกับโหนดที่มีความสูง h เป็น $O(h)$ และสามารถแสดง cost รวมของ Build-Max-Heap ซึ่งมีเพดานเป็น

$$\sum_{h=0}^{\lfloor \log N \rfloor} \left\lceil \frac{N}{2^{h+1}} \right\rceil O(h) = O\left(N \sum_{h=0}^{\lfloor \log N \rfloor} \frac{h}{2^h} \right)$$

พิจารณา Integrating and differentiating series

$$\sum_{k=0}^{\infty} k x^2 = \frac{x}{(1-x)^2}, \quad |x| < 1$$

คำนวณผลบวกตัวสุดท้ายโดยแทนค่า $x = 1/2$ ในอนุกรมข้างต้น

$$\sum_{h=0}^{\infty} \frac{h}{2^h} = \frac{1/2}{(1-1/2)^2} = 2$$

ดังนั้นได้ขอบเขต running time ของ Build-Max-Heap เป็น

$O\left(N \sum_{h=0}^{\lfloor \log N \rfloor} \frac{h}{2^h} \right) = O\left(N \sum_{h=0}^{\infty} \frac{h}{2^h} \right) = O(N)$: Linear time[6]

4.4.3. โหนดอย่างมากสำหรับต้นไม้ที่มีความสูง h ใน heap ที่มี n สมาชิก

ใช้อุปนัยเชิงคณิตศาสตร์ในการวิเคราะห์

Basis step: $h = 0$, จำนวน leave คือ $\lceil n/2 \rceil = \lceil n/2^{0+1} \rceil$: inductive hypothesis

Inductive step: พิจารณาความสูง h - 1 พิจารณาต้นไม้ย่อยนั้นแล้วลบ leave อื่นทั้งหมดออก จะได้ต้นไม้ใหม่ที่มี $n - \lceil n/2 \rceil = \lfloor n/2 \rfloor$ สมาชิก คำนวณจำนวนโหนดของต้นไม้ใหม่โดยใช้ inductive hypothesis จะได้โหนดเป็น

$$T = \left\lceil \lfloor n/2 \rfloor / 2^{h-1+1} \right\rceil < \left\lceil (n/2)/2^{h} \right\rceil = \left\lceil \frac{n}{2^{h+1}} \right\rceil$$

4.5. The heap sort algorithm

heap sort เริ่มต้นด้วย Build-Max-Heap เพื่อสร้าง max-heap จากอาร์เรย์อินพุต $A[1...N]$ เนื่องจากสมาชิกที่มีค่ามากที่สุดจะอยู่ที่ $A[1]$ เราสามารถส่งไปยังตำแหน่งสุดท้ายโดยการสลับกับ $A[N]$ ถ้าขณะนี้ลบโหนด N ออกจาก heap และลดขนาดของ heap ได้

พบว่าโหนดลูกของ root ยังเป็น max-heap แต่ root ใหม่อาจขัดเงื่อนไขของ max-heap จึงต้องทำ Max-Heapify เพื่อคงไว้ซึ่งความเป็น max-heap

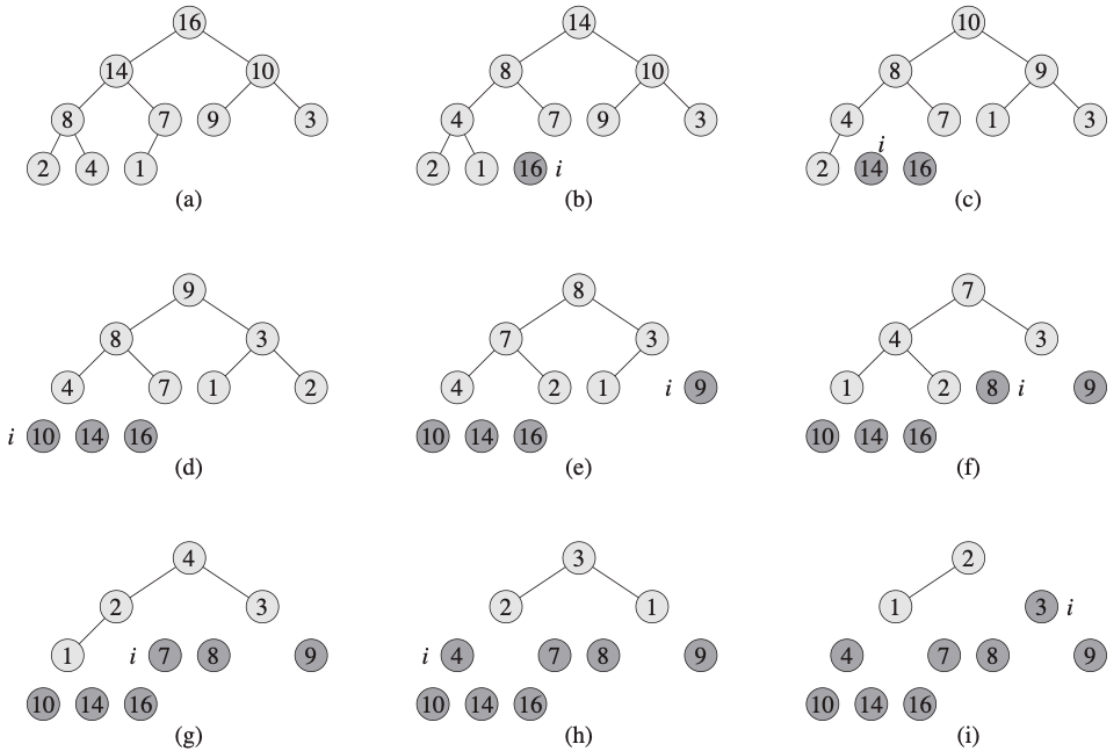
4.5.1. Pseudocode และ ตัวอย่าง (Obvious process)

Algorithm *J* (Heap sort)

J0:	**Input**: Array A	
J1:	[Build] Build-Max-Heap(A)	$O(N)$
J2:	[Heapify] **for** i = A.length **downto** 2 **do**	
	swap($A[1]$, $A[i]$)	$O(1)$
	decease $A.heap\text{-}size$ by 1	$O(1)$
	Max-Heapify(A, 1)	$O(\log N)$

Running time $T(N) = O(N) + \sum_{i=2}^{N} \left[O(\log N) + O(1) + O(1) \right] \in O(N \log N)$

ตัวอย่าง



A	1	2	3	4	7	8	9	10	14	16

4.6.QuickHeapsort

ในหัวข้อนี้จะแนะนำอัลกอริทึมที่มีประสิทธิภาพสูงและใช้หลักการ in-place sorting ซึ่งรวมสองอัลกอริทึมเข้าด้วยกันคือ Quicksort และ Heapsort หรือในรายละเอียดคือ ขั้นตอนการ partition ของ Quicksort และ Max-Heapify มารวมกัน โดยใช้หน่วยความจำเท่าเดิม

Complexity ของอัลกอริทึมเชิงจำนวนของการเปรียบเทียบ มีค่าเฉลี่ยอยู่ที่ $(n \log n) + 2.996n + O(n)$ สำหรับอินพุตขนาด n และ worst-case จะเป็น complexity ของ Quciksort ปกติ

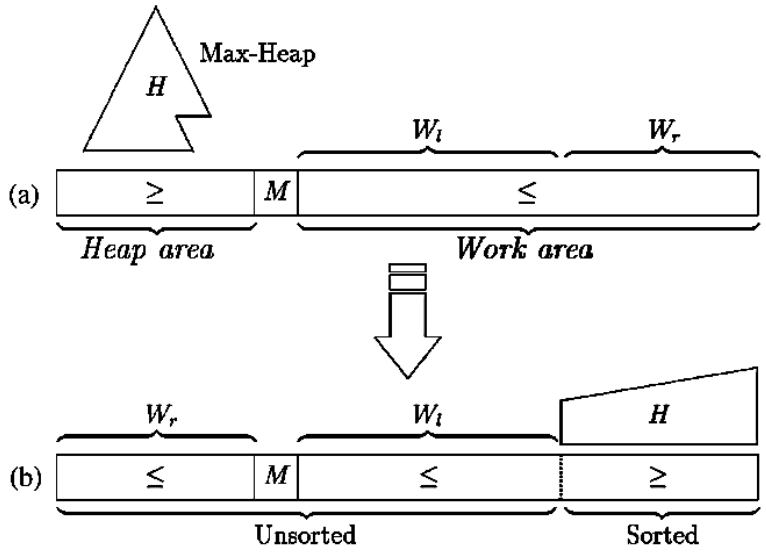
การทำงานเหมือนกับ Quicksort ที่ตอนแรกจะเลือก pivot จากการแบ่งอาร์เรย์ โดยส่วนที่เล็กกว่าจะเป็น heap area และอีกส่วนเรียกว่า work area ในตอนท้ายสมาชิกจะย้ายไปยัง work area แบบเรียงแล้ว และวนรอบทำไปเรื่อย ๆ

4.6.1.ขั้นตอนการทำงาน

I. กำหนดอาร์เรย์ $A[1...n]$ เป็นอินพุต, pivot M จาก index m ใน $\{A[1], ..., A[n]\}$ (ในส่วนของ Quicksort pivot สามารถหาได้หลายวิธี)

II. เกิดอาร์เรย์ใหม่จากการแบ่งเป็น $A[1...m - 1]$ และ $A[m + 1...n]$ โดย $A[m] = M$ และ $A[1...m - 1] \geq M \geq A[m + 1...n]$ (reverse) จากนั้นเลือกอาร์เรย์ที่เล็กกว่าเป็น heap area

III. ดำเนินการ Max-heapify กับ heap area



IV. เมื่อเรียงเสร็จจะย้ายไปขวาสุดจากนั้นจะหา pivot ใหม่และวนรอบทำไปเรื่อย ๆ

4.6.2.วิเคราะห์ Average-case

ในเบื้องต้นจะวิเคราะห์ในกรณีที่ pivot ถูกเลือกแบบตายตัว เช่น เลือกสมาชิกตัวแรกเป็น pivot เสมอ โดยจะใช้ทฤษฎีบทที่ 5 ใน *QuickHeapsort, an efficient mix of classical sorting algorithms* ของ *D. Cantone* และ *G. Cincotti* ได้ผลค่าเฉลี่ยอยู่ที่ $(n \log n) + 2.996n + O(n)$ สามารถศึกษารายละเอียดทั้งหมดได้ในงานวิจัยฉบับเดียวกัน[7]

4.7.สรุป Heap sort

การกระทำการใน Heap มีหลายแบบ เช่น ลบโหนด root โดยการสลับ root node กับ node ท้ายสุดและลบออกไป นอกจากนี้ยังมีการ sort ข้อมูลที่เรียกว่า heapsort โดยการนำโหนด root บนสุดออกจากข้อมูล และทำการ heapify ทำการนำโหนดบนสุดออกไปเรื่อยๆจนได้ข้อมูลที่เรียงกันมี running time ของ heapify จะเป็น $N \log(N)$ และขอบเขตเป็น N ในส่วนของ heapsort จะเป็น $N \log(N)$ เช่นเดียวกัน

QuickHeapsort เป็นอัลกอริทึมที่ถูกดัดแปลงมาจาก Heapsort โดยการนำไปรวมกับ Quicksort เพื่อเพิ่มประสิทธิภาพการทำงานของอัลกอริทึ่มโดยการแบ่งส่วนข้อมูลด้วย pivot และแบ่งโซนการทำงานอัลกอริทึ่มเหมือน quicksort มี Average case เป็น $(N \log(N)) + 2.996N + O(N)$

5. Code and documentation

5.1.Linear search algorithm

5.1.1.Algorithm A (Linear search)

โปรแกรมภาษา: ไพธอน

ที่อยู่ไฟล์: `code/linear_search/algorithm_A.py`

```
 1 # Algorithm A (Linear search)
 2 def algorithmA(R, K):
 3
 4     i = 0# Initialize
 5     N = len(R)
 6     for i in range(0, N):# Increment
 7
 8         if (R[i] == K):# Compare
 9             print("A terminate successfully.")
10             break
11
12     if (i == N - 1 and R[i] != K):
13         print("A terminate unsuccessfully.")
```

ข้อมูลรับเข้า: `R` อาร์เรย์ของจำนวนจริงเป็นฐานข้อมูลที่ต้องการค้น
`K` จำนวนจริงที่ต้องการค้นหาในอาร์เรย์ `R`

ข้อมูลส่งออก: ไม่มี

การทำงาน

บรรทัดที่ `4 - 6`: เริ่ม `i = 0` เพื่อวนรอบจนถึง `N = len(R)` (ขนาดของอาร์เรย์ `R`)

บรรทัดที่ `8`: เปรียบเทียบว่าค่าของจำนวนที่ตำแหน่ง `i` เท่ากับค่า `K` หรือไม่ โดยถ้า

บรรทัดที่ `9 - 10`: เท่ากัน โปรแกรมจะจบแบบ success แสดงผลและสิ้นสุดการทำงาน แต่ถ้าไม่เท่าจะวนรอบทำงานต่อ

```
A terminate successfully.
```

บรรทัดที่ `12 - 13`: ไม่พบค่าที่ต้องการ โปรแกรมจะจบแบบ unsuccess แสดงผลและสิ้นสุดการทำงาน

```
A terminate unsuccessfully.
```

5.1.2. Algorithm B (Quick Linear search)

โปรแกรมภาษา: ไพธอน

ที่อยู่ไฟล์: code/linear_search/algorithm_B.py

```
1  # Algorithm B (Quick Linear search)
2  def algorithmB(R, K):
3
4      R.append(K)
5      i = 0# Initialize
6      N = len(R)
7
8      for i in range(0, N):# Increment
9
10         if (R[i] == K):# Compare
11             if(i < N - 1):# End?
12                 print("B terminate successfully.")
13                 break
14             else:
15                 print("B terminate unsuccessfully.")
16                 break
```

ข้อมูลรับเข้า: R อาร์เรย์ของจำนวนจริงเป็นฐานข้อมูลที่ต้องการค้น
K จำนวนจริงที่ต้องการค้นหาในอาร์เรย์ R

ข้อมูลส่งออก: ไม่มี

การทำงาน

บรรทัดที่ 4: เพิ่ม K ที่ตำแหน่ง N ของ R

บรรทัดที่ 5 – 8: เริ่ม i = 0 เพื่อวนรอบจนถึง N = len(R) (ขนาดของอาร์เรย์ R)

บรรทัดที่ 10: เปรียบเทียบว่าค่าของจำนวนที่ตำแหน่ง i เท่ากับค่า K หรือไม่ โดยถ้า

บรรทัดที่ 11 – 13: เท่ากัน จะตรวจสอบ i < N – 1 ถ้าจริงโปรแกรมจะจบแบบ success แสดงผลและสิ้นสุดการทำงาน

```
B terminate successfully.
```

บรรทัดที่ 14 – 16: มิฉะนั้น โปรแกรมจะจบแบบ unsuccess แสดงผลและสิ้นสุดการทำงาน

```
B terminate unsuccessfully.
```

5.1.3. Algorithm C (Presort Linear search)

โปรแกรมภาษา: ไพธอน

ที่อยู่ไฟล์: code/linear_search/algorithm_C.py

```
1  # Algorithm C (Presort Linear search)
2  def algorithmC(R, K):
3
4      R.sort()
5      R.append(float("inf"))
6      i = 0# Initialize
7      N = len(R)
8
9      for i in range(0, N):# Increment
10
11         if (K <= R[i]):# Compare
12             if(K == R[i]):# End?
13                 print("C terminate successfully.")
14                 break
15             else:
16                 print("C terminate unsuccessfully.")
17                 break
```

<u>ข้อมูลรับเข้า</u>: R อาร์เรย์ของจำนวนจริงเป็นฐานข้อมูลที่ต้องการค้น
K จำนวนจริงที่ต้องการค้นหาในอาร์เรย์ R

<u>ข้อมูลส่งออก</u>: ไม่มี

<u>การทำงาน</u>

บรรทัดที่ 4, 5: เรียงอาร์เรย์จากน้อยไปมาก, เพิ่ม ∞ ที่ตำแหน่ง N ของ R

บรรทัดที่ 6 – 9: เริ่ม i = 0 เพื่อวนรอบจนถึง N = len(R) (ขนาดของอาร์เรย์ R)

บรรทัดที่ 11: เปรียบเทียบว่า K ≤ R[i] หรือไม่ โดยถ้า

บรรทัดที่ 12 – 14: จริง จะตรวจสอบ K == R[i] ถ้าจริงโปรแกรมจะจบแบบ success แสดงผลและสิ้นสุดการทำงาน

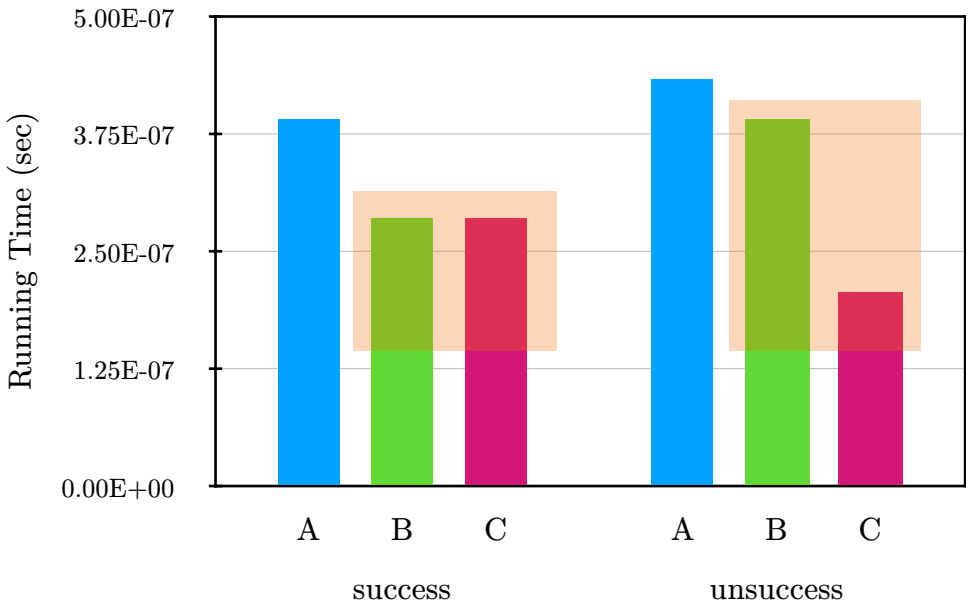
```
C terminate successfully.
```

บรรทัดที่ 15 – 17: มิฉะนั้น โปรแกรมจะจบแบบ unsuccess แสดงผลและสิ้นสุด

```
C terminate unsuccessfully.
```

5.1.4.กรณีทดสอบ

I. `R = [50, 4, 95, 11, 16, 71, 25, 69, 7, 92]`
`K = 7 (success), K = 8 (unsuccess)`

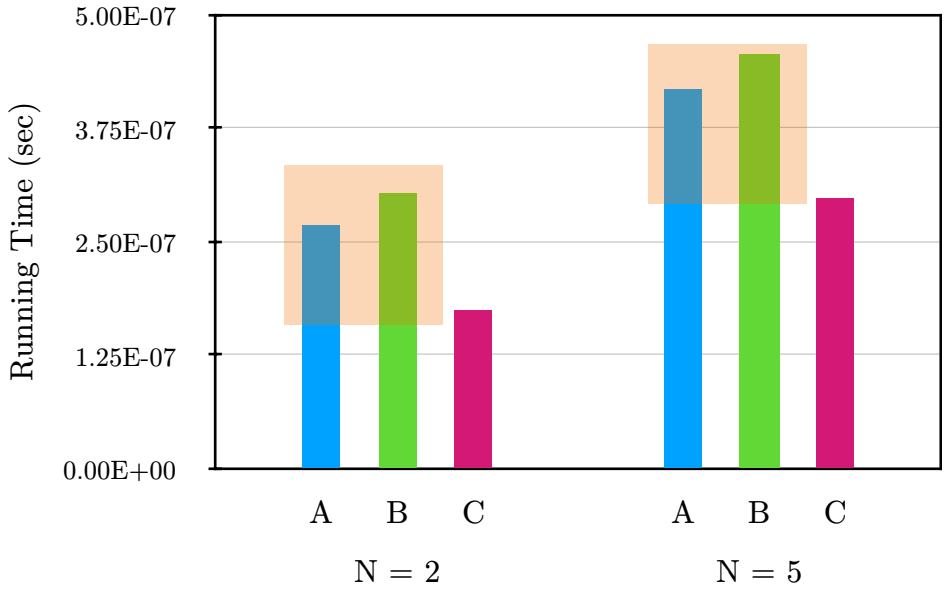


กราฟ 6.1

ในกรณี success จะเห็นว่า B และ C ทำงานในเวลาที่แทบจะไม่ต่างกัน แต่ในกรณีที่ unsuccess จะเห็นว่า C (presort) ทำงานเร็วกว่าถึงเท่าตัว

II.เปรียบเทียบกรณี unsuccessfull เพื่อพิจารณา Upper-bound

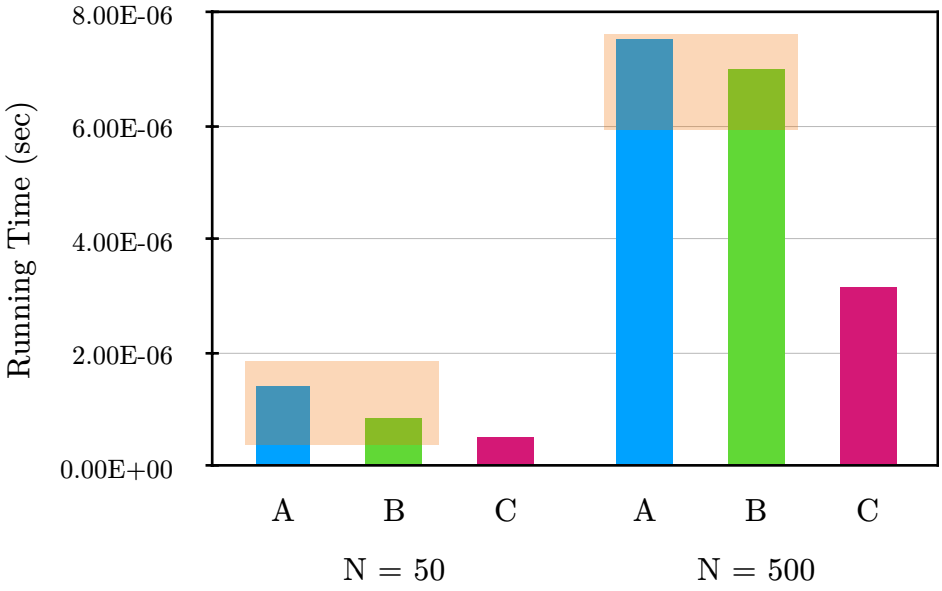
`i. N = 2, N = 5, K = Average`



กราฟ 6.2

ในขณะที่ยังมีข้อมูลไม่ถึง 8 ตัวจะเห็นว่าอัลกอริทึม B ไม่ได้ทำงานเร็วกว่าอัลกอริทึม A แต่อย่างใด ซึ่งสอดคล้องกับบทพิสูจน์ในหัวข้อ 1.4.2.

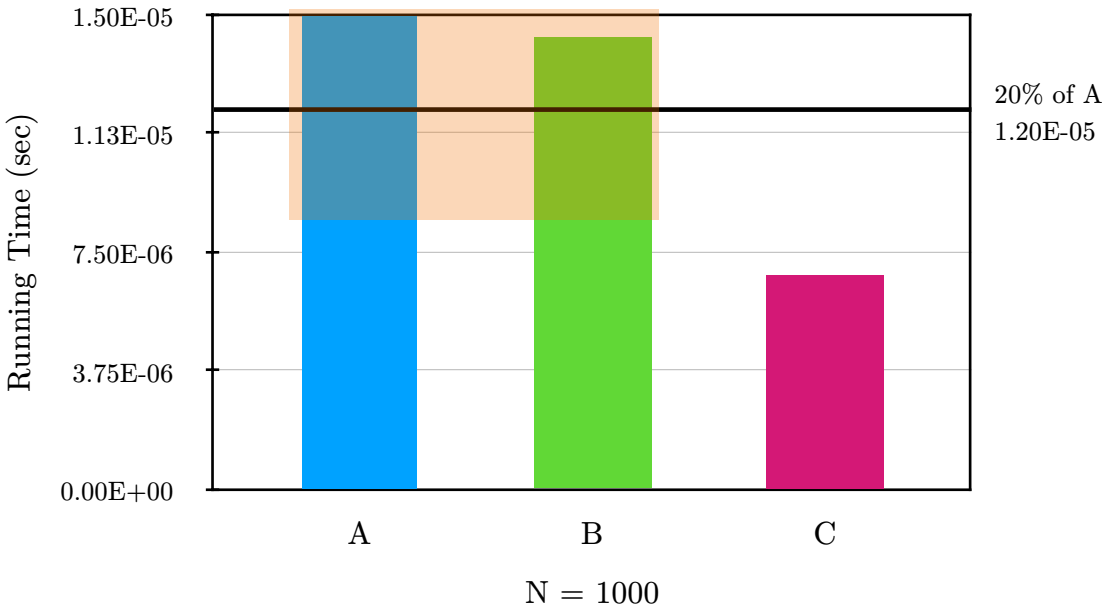
ii.N = 50, N = 500, K = Average



กราฟ 6.3

มีข้อมูลเกิน 8 ตัวจะเห็นว่าอัลกอริทึม B ทำงานเร็วกว่าอัลกอริทึม A ซึ่งสอดคล้องกับบทพิสูจน์ในหัวข้อ 1.4.2.

iii.N = 1000, K = Average



กราฟ 6.4

มีข้อมูลจำนวนมากจะเห็นว่าอัลกอริทึม B ทำงานเร็วกว่าอัลกอริทึม A ไม่เกิน 20% ซึ่งเป็นผลจากการลู่เข้าของฟังก์ชันในหัวข้อเดียวกัน

5.2.2-Dimensional closest pair algorithm

5.2.1.class ที่มีข้อมูล x และ y

โปรแกรมภาษา: ไพธอน

ที่อยู่ไฟล์: code/two_dim_closest_pair/algorithm_D_E.py

```
1 class Point:
2       def __init__(self, x, y):
3             self.x = x
4             self.y = y
```

5.2.2.เมธอดที่ใช้ในการหาระยะห่างระหว่างจุดสองจุด

โปรแกรมภาษา: ไพธอน

ที่อยู่ไฟล์: code/two_dim_closest_pair/algorithm_D_E.py

```
1 def distance(p1, p2):
2       return math.sqrt(((p1.x - p2.x)*(p1.x - p2.x))
3             + ((p1.y - p2.y)*(p1.y - p2.y)))
```

ข้อมูลรับเข้า: p1, p2 จุดที่ต้องการทราบระยะห่าง

ข้อมูลส่งออก: ระยะห่างของสองจุด

การทำงาน คำนวณหา 2-*Minkowski distance*

5.2.3. Algorithm *D* (Brute-force 2D-Closest pair)

โปรแกรมภาษา: ไพธอน

ที่อยู่ไฟล์: `code/two_dim_closest_pair/algorithm_D_E.py`

```
 1 # Algorithm D (Brute-force 2D-Closest pair)
 2 def algorithmD(P):
 3
 4         dist = float('inf')
 5
 6         for i in range(0, len(P) - 1):
 7             for j in range(i + 1, len(P)):
 8
 9                 if (Point.distance(P[i], P[j]) <= dist):
10                     dist = Point.distance(P[i], P[j])
11
12         return dist
```

ข้อมูลรับเข้า: `P` อาร์เรย์ของตัวแปรที่ต้องการหาคู่ข้อมูลที่ระยะใกล้กันที่สุด

ข้อมูลส่งออก: `dist` ระยะที่ใกล้ที่สุดของคู่ข้อมูลที่รับเข้า

การทำงาน

บรรทัดที่ `4`: ประกาศตัวแปร `dist`

บรรทัดที่ `6 - 10`: เริ่มต้นวนรอบจาก `i = 1` จับคู่ หาระยะห่างด้วยฟังก์ชัน `distance` กับข้อมูลถัดไปทุก ๆ ตัว เริ่มต้นจาก `j = i + 1` จนถึงตัวสุดท้าย เมื่อจับคู่ทุกตัวกับข้อมูลตัวที่ 1 เสร็จ `i` จะขยับไปที่ตำแหน่งถัดไป แล้วจับคู่หา `distance` ต่อจนครบทุกคู่ของข้อมูล และทุกๆครั้งที่จับคู่ `distance` เมื่อได้ค่า `dist` ที่น้อยกว่า จะอัพเดทค่า `dist`

บรรทัดที่ `12`: เมื่อคำนวณเสร็จสิ้น โปรแกรมจะส่งออกค่า `dist` ซึ่งเป็นระยะที่ใกล้ที่สุดของคู่ข้อมูลที่รับเข้า

5.2.4. Algorithm E (Devide-and-Conquer 2D-Closest pair)

โปรแกรมภาษา: ไพธอน

ที่อยู่ไฟล์: `code/two_dim_closest_pair/algorithm_D_E.py`

```
 1 # Algorithm E (Devide-and-Conquer 2D-Closest pair)
 2
 3     def algorithmE(P):
 4         X = sorted(P, key=lambda P: P.x)
 5         Y = sorted(P, key=lambda P: P.y)
 6
 7         if (len(P) <= 3):
 8             return Point.algorithmD(P)
 9
10         Pl = X[:math.ceil(len(P) / 2)]
11         Pr = X[math.floor(len(P) / 2):]
12
13         dl = Point.algorithmE(Pl)
14         dr = Point.algorithmE(Pr)
15         d = min(dl, dr)
16
17         Yp = []
18         for p in Y:
19             if (abs(p.x) <= 2 * d):
20                 Yp.append(p)
21
22         dp = float('inf')
23         for i in range(len(Yp)):
24             for j in range(i + 1, min(i + 7, len(Yp))):
25                 if (dp < Point.distance(Yp[i], Yp[j])):
26                     dp = Point.distance(Yp[i], Yp[j])
27
28         return min(d, dp)
```

<u>ข้อมูลรับเข้า</u>: `P` อาร์เรย์ของตัวแปรที่ต้องการหาคู่ข้อมูลที่ระยะใกล้กันที่สุด

<u>ข้อมูลส่งออก</u>: `dist` ระยะที่ใกล้ที่สุดของคู่ข้อมูลที่รับเข้า

<u>การทำงาน</u>

บรรทัดที่ `4 – 5`: จัดเรียงข้อมูลตามแกน x และ y ของ อาร์เรย์ของข้อมูลรับเข้า

บรรทัดที่ `7`: เช็คว่าจำนวนข้อมูลน้อยกว่าหรือเท่ากับ 3 หรือไม่ ถ้าใช่จะถ้าสู่การ

คำนวณแบบ base case เพราะไม่สามารถแบ่งครึ่งต่อได้ซึ่งจะนำไปคำนวณแบบ brute force

บรรทัดที่ 10 – 14: เข้าขั้นตอนการแบ่งครึ่งข้อมูล แบบ recursive โดยจะแบ่งข้อมูลไปเรื่อยๆ จนกว่าจำนวนข้อมูลจะสามารถคำนวณในเงื่อนไข base case ได้

บรรทัดที่ 15: เมื่อหาคู่ข้อมูลที่มีระยะใกล้กันที่สุดของข้อมูลได้แล้ว จะนำคำตอบที่ได้จากขอบเขตที่ติดกัน มาเปรียบเทียบหาคำตอบที่สั้นที่สุด โดยคำตอบจากฝั่งซ้ายเป็น dl ฝั่งขวาเป็น dr และคำตอบที่สั้นที่สุดเป็น d

บรรทัดที่ 18 – 20: วนรอบหาค่าในเซตแกน y ที่อยู่ในแถบ 2d

บรรทัดที่ 22 – 26: คำนวณการหาระยะห่างของจุดข้อมูลข้ามขอบเขตโดยจนรอบจับคู่ แบบ brute force ในขอบเขตข้อมูลความกว้าง 2d จากจุดแบ่งข้อมูล โดยแต่ละจุดวนรอบจับคู่อย่างมากภายในวงกลมที่หาจุดไว้ในโค้ดบรรทัดที่ 18 – 20 และวนรอบจับคู่มากสุดเท่ากับ 7 ครั้ง หรือจับคู่ตามจำนวนที่จับคู่ได้ในกรณีที่จับคู่ได้น้อยกว่า 7 ภายในขอบเขต

บรรทัดที่ 28: เมื่อคำนวนแบบ recursive เสร็จสิ้น โปรแกรมจะส่งออกข้อมูล ที่เป็นระยะที่ใกล้ที่สุดของคู่ข้อมูลที่รับเข้า สิ้นสุดการทำงาน

5.2.5.กรณีทดสอบ

I. จาก geeks4geek

```
P = {2, 3}, {12, 30}, {40, 50}, {5, 1}, {12, 10}, {3, 4}
return 1.4142
```

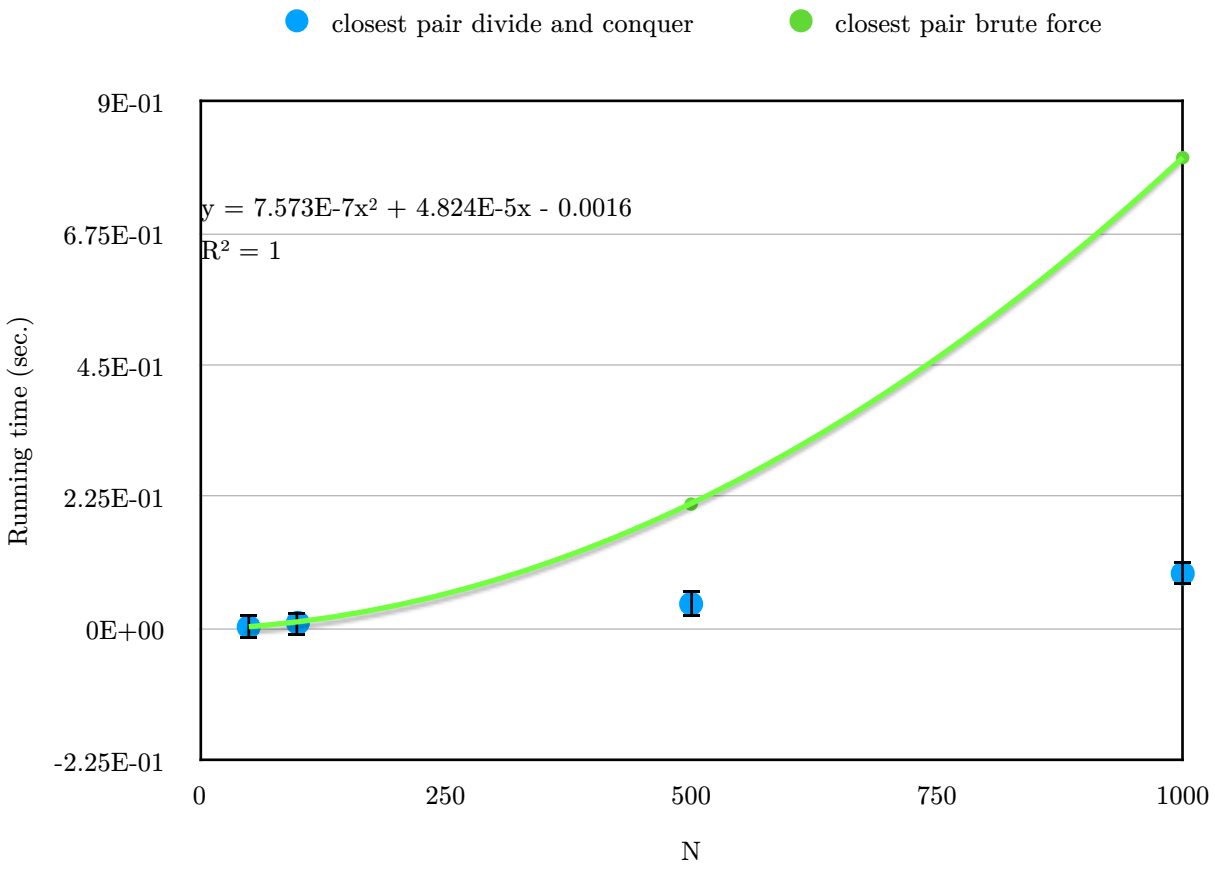
II. ชุดข้อมูลที่ตำแหน่งใกล้เคียงกัน

```
P = {1, 1}, {1, 2}, {2, 2}, {2, 3}, {3, 3}, {3, 4}
return 1
```

III. ชุดข้อมูลที่ตำแหน่งเดียวกันทั้งหมด

```
P = {1, 1}, {1, 1}, {1, 1}, {1, 1}, {1, 1}, {1, 1}
return 0
```

IV.เปรียบเทียบ running time



กราฟ 6.5 โปรแกรมไม่สามารถ fit-graph N log(N)

กราฟเป็นไปตามทฤษฎีอย่างแม่นยำส่งผลต่อ Effectiveness **ที่สูง และเนื่องจากไม่สามารถแสดงกราฟ** Nlog(N) **ได้ จึงนำเสนอความเบี่ยงเบนผ่านค่า** SD

5.3.Unique element algorithm

5.3.1.Algorithm F (Ordinary unique algorithm)

โปรแกรมภาษา: ไพธอน

ที่อยู่ไฟล์: `code/unique_element/algorithm_F.py`

```
1  # Algorithm F (Ordinary unique algorithm)
2  def algorithmF(F):
3        for i in range(0, len(F) – 2):
4              for j in range(i + 1, len(F) – 1):
5                    if F[i] == F[j]:
6                          return False
7        return True
```

ข้อมูลรับเข้า: `F` **ชุดตัวเลขจำนวนจริงที่ต้องการตรวจสอบว่ามีตัวเลขหมือนกันหรือไม่**
ข้อมูลส่งออก: **ส่งค่าความจริง เพื่อบอกว่ามีจำนวนซ้ำหรือไม่**

การทำงาน
บรรทัดที่ `3:` **เป็นการกำหนดค่า** `i = 0` **วนรอบถึง** `len(F) – 2` **(ขนาดอาเรย์ -** 2**)**
บรรทัดที่ `4:` **เป็นการกำหนดค่า** `j = i – 1` **วนรอบถึง** `len(F) – 1`
บรรทัดที่ `5:` **เปรียบเทียบว่าจำนวนตำแหน่งที่** `i` **เท่ากับค่าที่อยู่ตำแหน่งที่** `j` **หรือไม่**
ถ้าไม่ บรรทัดที่ `6:` **ส่งค่าความจริง** `False` **(มีตัวซ้ำ)**
บรรทัดที่ `7:` **ส่งค่าความจริง** `True` **(ไม่มีตัวซ้ำ)**

5.3.2. Algorithm G (Presort unique element algorithm)

โปรแกรมภาษา: ไพธอน

ที่อยู่ไฟล์: `code/unique_element/algorithm_G.py`

```
1  # Algorithm G (Presort unique algorithm)
2  def algorithmG(G):
3        G.sort()#sorting
4        for i in range(0, len(G) – 2):
5              if G[i] == G[i + 1]:
6                    return False
7        return True
```

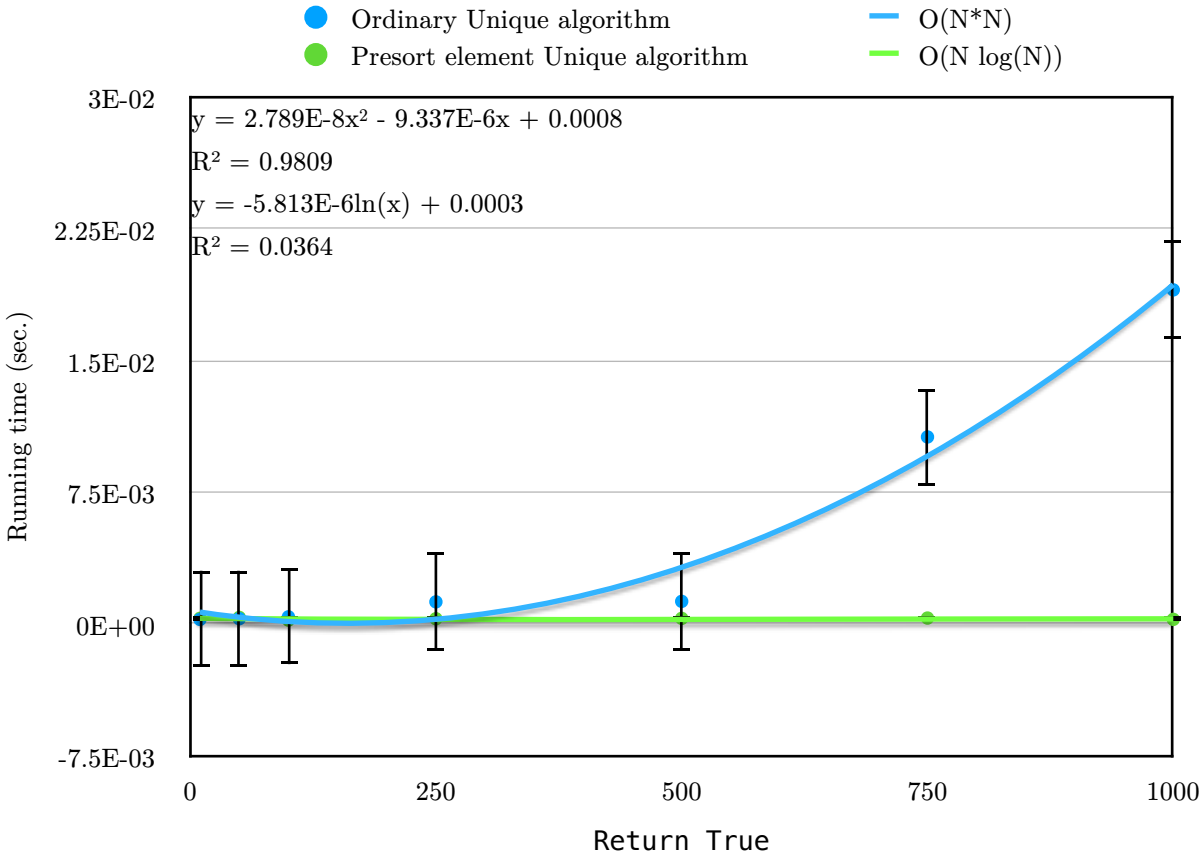
ข้อมูลรับเข้า: `G` ชุดตัวเลขจำนวนจริงที่ต้องการตรวจสอบว่ามีตัวเลขหมือนกันหรือไม่
ข้อมูลส่งออก: ส่งค่าความจริง เพื่อบอกว่ามีจำนวนซ้ำหรือไม่

การทำงาน

บรรทัดที่ `3:` เรียงลำดับข้อมูลก่อนการตรวจสอบ
บรรทัดที่ `4:` เป็นการกำหนดค่า `i = 0` วนรอบถึง `len(F) – 2` (ขนาดอาเรย์ - 2)
บรรทัดที่ `5:` เปรียบเทียบว่าจำนวนตำแหน่งที่ `i` เท่ากับค่าที่อยู่ตำแหน่งที่ `i + 1` หรือไม่ ถ้าไม่ บรรทัดที่ `6:` ส่งค่าความจริง `False` (มีตัวซ้ำ)
บรรทัดที่ `7:` ส่งค่าความจริง `True` (ไม่มีตัวซ้ำ)

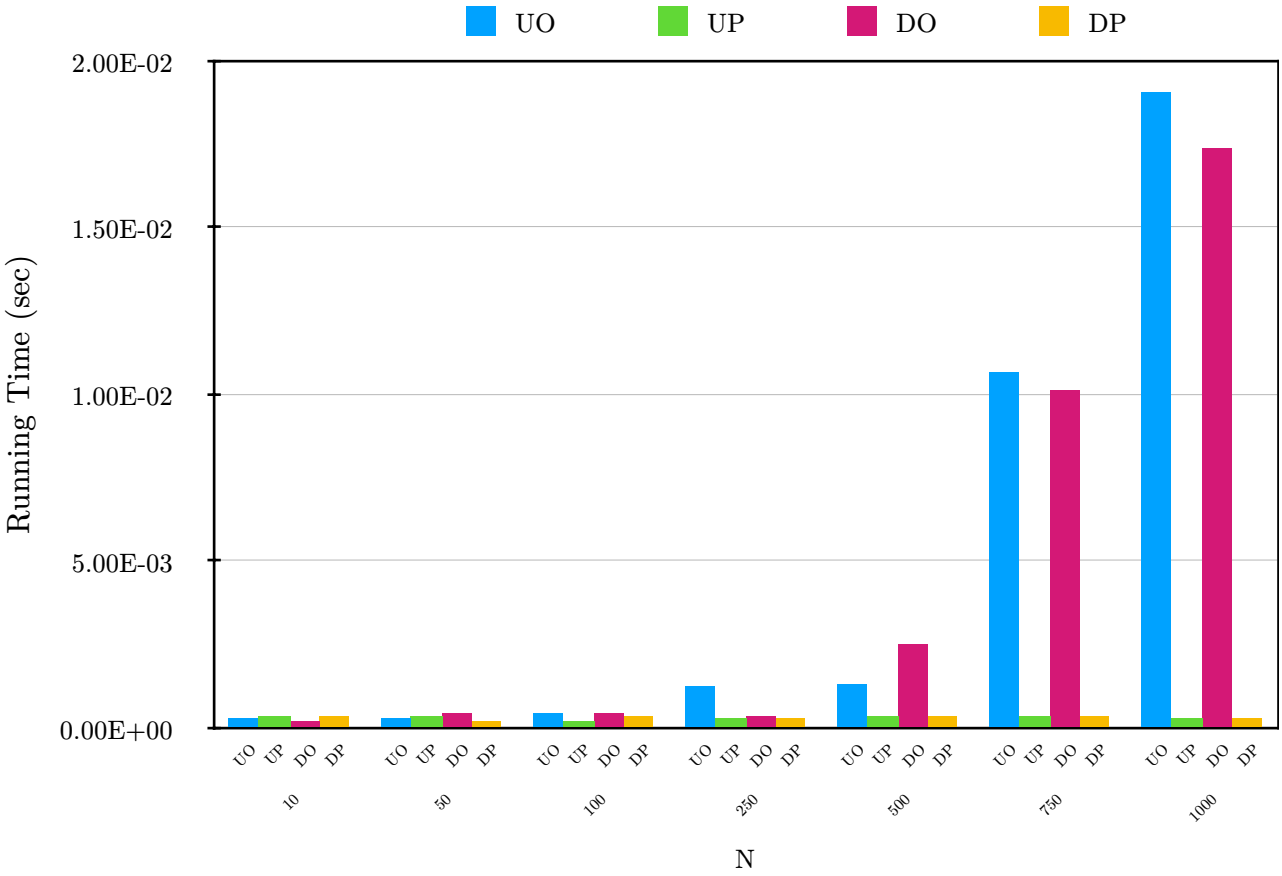
5.3.3.กรณีทดสอบ

I. กราฟแสดงให้เห็นฟังก์ชันการทำงานจริงซึ่งสอดคล้องกับ Complexity ในหัวข้อที่ 3. ที่ Ordinary เป็น $O(N^2)$ และ Presort เป็น $O(N \log N)$



กราฟ 6.6

II. กราฟเปรียบเทียบเวลาในการทำงานของกรณี Unique และ Duplicate



กราฟ 6.7 UO: *Unique with Ordinary*, UP: *Unique with Presort*
DO: *Duplicate with Ordinary*, DP: *Duplicate with Presort*

ส่วนใหญ่พบว่าในกรณีที่ Dupliacate **จะทำงานเร็วกว่า** Unique **ซึ่งอันที่จริงก็เป็นที่แน่นอนเพราะหากพบตัวซ้ำโปรแกรมจะสิ้นสุดการทำงานทันที**

5.4.Heap sort algorithm

5.4.1.Algorithm H (Max-Heapify)

โปรแกรมภาษา: จาวา

ที่อยู่ไฟล์: code/heap_sort/algorithm_H.py

```
 1 /** Algorithm H (Max-Heapify) */
 2 static void algorithmH(int arr[], int n, int i)
 3 {
 4       int largest = i;
 5       int l = 2 * i + 1;
 6       int r = 2 * i + 2;
 7       if (l < n && arr[l] > arr[largest])
 8             largest = l;
 9       if (r < n && arr[r] > arr[largest])
10             largest = r;
11       if (largest != i)
12       {
13             int temp = arr[i];
14             arr[i] = arr[largest];
15             arr[largest] = temp;
16             algorithmH(arr, n, largest);
17       }
18 }
```

<u>ข้อมูลรับเข้า</u>: `int arr[]` อาเรย์ของข้อมูลที่ต้องการสร้าง heap
`int n` ความยาว array เริ่มต้น
`int i` คือตำแหน่งที่ถูกกำหนดโดยการวนรอบใน algorithm I เริ่มจากครึ่งนึงของขนาดอาเรย์ - 1

<u>ข้อมูลส่งออก</u>: ไม่มี

<u>การทำงาน</u>

บรรทัดที่ `4 – 6`: กำหนดค่าตัวแปรต่างๆในฟังก์ชันโดยรับค่า `i` จาก `algorithm I`

บรรทัดที่ `7 – 8`: เปรียบเทียบว่าค่า `l` น้อยกว่าขนาดอาเรย์ `(n)` และอาเรย์ตำแหน่งที่ `l` มากกว่า อาเรย์ตำแหน่งที่ `largest` (ณ ขณะนั้น หรือไม่) ถ้าจริงทั้งสองกรณี ให้ค่า `largest = l`

บรรทัดที่ `9 - 10:` **เปรียบเทียบว่าค่า** `r` **น้อยกว่าขนาดอาเรย์** `(n)` **และอาเรย์ตำแหน่งที่** `r` **มากกว่า อาเรย์ตำแหน่งที่** `largest` **(ณ ขณะนั้น หรือไม่) ถ้าจริงทั้งสองกรณี ให้ค่า** `largest = l`

บรรทัดที่ `11 - 16:` **เปรียบเทียบว่าถ้าค่า** `largest` **ไม่เท่ากับค่า** `i` **(เกิดการสลับค่า** `largest` **กับ** `l/r` **ขึ้น) ให้จำค่า** array **ตำแหน่ง** `i` **ใส่ค่า** `temp` **และให้** array **ตำแหน่งที่** `i` **มีค่าเท่ากับ** array **ตำแหน่งที่** `largest` **ให้** array **ตำแหน่งที่** `largest` **จำค่า** `temp` **เข้าไป หลังจากนั้นทำการวนรอบ** Heapify **ไปเรื่อยๆจนกว่าจะไม่เจอค่า** `largest` **ที่ไม่เท่ากับค่า** `i`

5.4.2. Algorithm I (Build-Max-Build)

โปรแกรมภาษา: จาวา

ที่อยู่ไฟล์: `code/heap_sort/algorithm_I.py`

```
1 /** Algorithm I (Build-Max-Heap) */
2 static void algorithmI(int arr[], int n)
3 {
4       int startIndex = (n / 2) - 1;
5       for (int i = startIndex; i >= 0; i--)
6             algorithmH(arr, n, i);
7 }
```

ข้อมูลรับเข้า: `int arr[]` **เป็นอาเรย์จากอาเรย์เริ่มต้นที่ผู้ใช้ป้อนข้อมูลเข้าไปเพื่อสร้าง** heap **และอาเรย์เริ่มต้นจะเปลี่ยนไปหลังจากผ่านการดำเนินการ**

`int n` **เป็นขนาดของอาเรย์เริ่มต้นที่ผู้ใช้ป้อนเข้าไป**

ข้อมูลส่งออก: ไม่มี

การทำงาน

บรรทัดที่ `4:` **ให้ค่า** `startIndex` **เป็นครึ่งนึงลบหนึ่งของขนาดอาเรย์**

บรรทัดที่ `6:` **ทำการ** heapify **ข้อมูล** array **ใหม่ไปจน** `startIndex` **เป็น** `0`

5.4.3.Algorithm J (Build-Max-Build)

โปรแกรมภาษา: จาวา

ที่อยู่ไฟล์: `code/heap_sort/algorithm_J.py`

```
 1 /** Algorithm J (Heap sort) */
 2 public void algorithmJ(int arr[])
 3 {
 4       int n = arr.length;
 5       for (int i = n / 2 - 1; i >= 0; i--)
 6             algorithmH(arr, n, i);
 7       for (int i = n - 1; i > 0; i--)
 8       {
 9             int temp = arr[0];
10             arr[0] = arr[i];
11             arr[i] = temp;
12             algorithmH (arr, i, 0);
13       }
14 }
```

ข้อมูลรับเข้า: `int arr[]` **เป็นอาเรย์จากอาเรย์เริ่มต้นที่ผู้ใช้ป้อนข้อมูลเข้าไปเพื่อสร้าง** heap **และอาเรย์เริ่มต้นจะเปลี่ยนไปหลังจากผ่านการดำเนินการ**

ข้อมูลส่งออก: ไม่มี

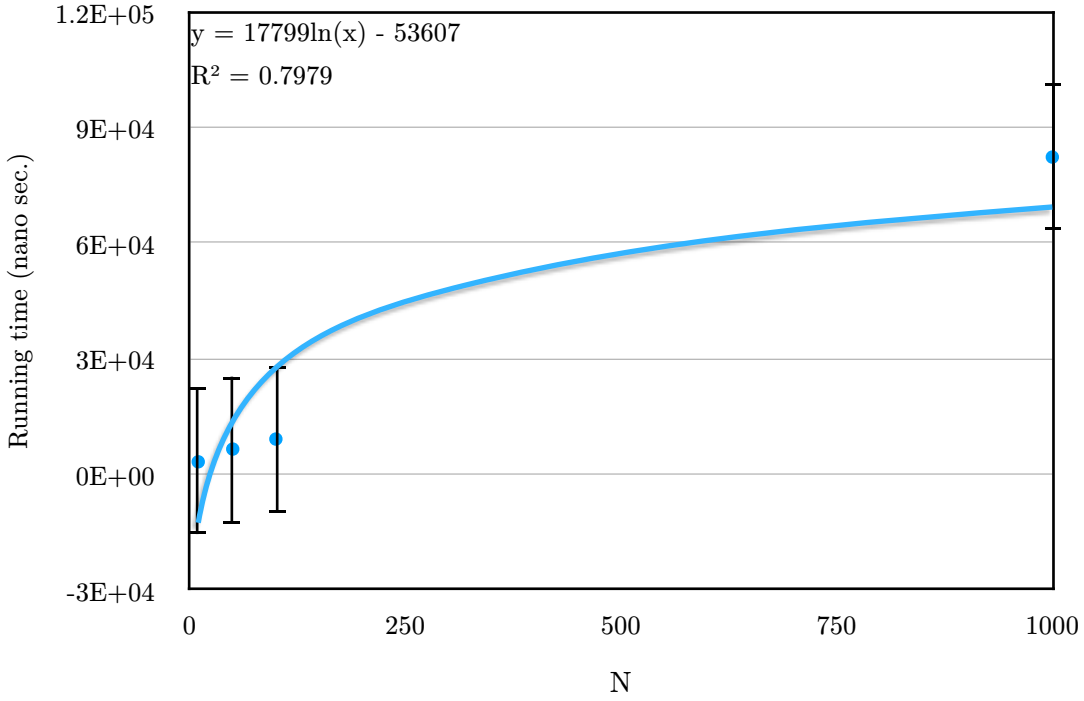
การทำงาน

บรรทัดที่ `4 - 6`**: เริ่มทำการสร้างโครงสร้าง** heap **จากอาเรย์เริ่มต้น**

บรรทัดที่ `9 - 12`**: ทำการสลับข้อมูลโหนดบนสุด(**`arr[0]`**) และโหนดล่างสุด(**`arr[i]`**) และทำการลดขนาดของ** Heap **จากนั้นทำการ** Heapify **ข้อมูล**

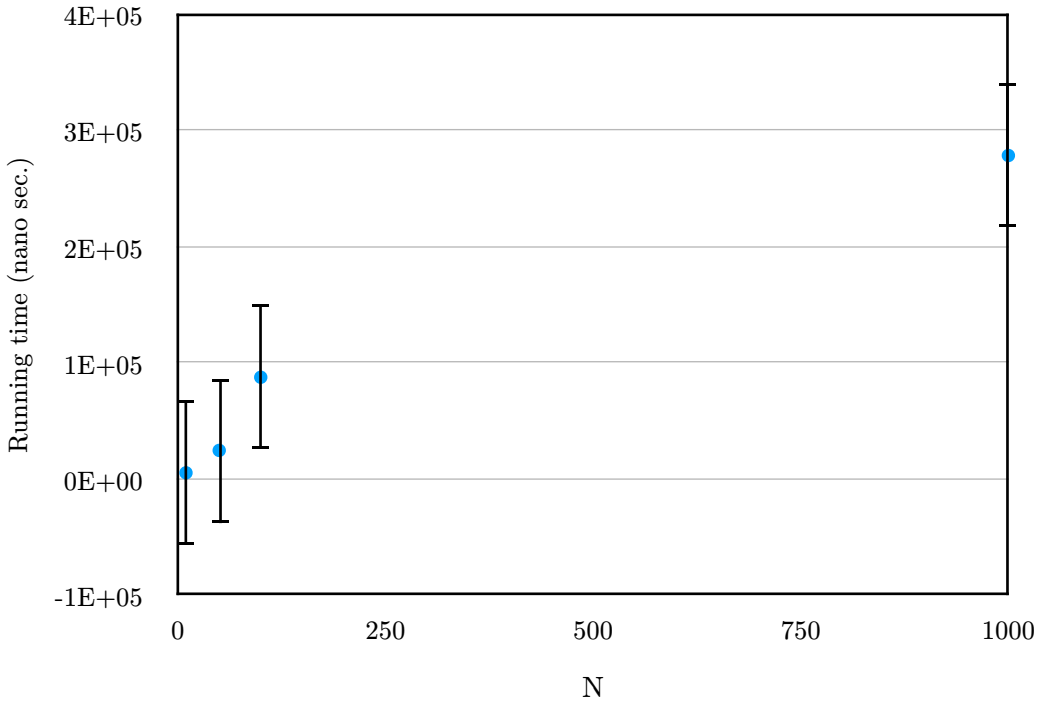
5.4.4.กรณีทดสอบ

I. Heapify: $O(\log(N))$



กราฟ 6.8

II. Heap sort: $O(N \log(N))$



กราฟ 6.9 โปรแกรมไม่สามารถ fit-graph N log(N)

6. References

[1] Knuth, Donald. Sorting and Searching. The Art of Computer Programming, Volume 3, Second Edition. (1998) Reading, MA: Addison-Wesley Professional. ISBN 0-201-89685-0

[2] Ovidiu Dăescu, Ka Taw Teo, Two-Dimensional Closest Pair Problem: A Closer Look. (2020)

[3] José C. Pereria, Fernando G. Lobo, An Optimized Devide-and-Conquer Algorithm for the Closest-Pair Problem in the Planar Case.

[4] Anany Levitin, Introduction to The Design and Analysis of Algorithms, Third Edition. pp. 203 - 204

[5] Ben-Or, Michael, Lower bounds for algebraic computation trees, Proc. 15th ACM Symposium on Theory of Computing,(1983) pp. 80–86, doi:10.1145/800061.808735

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Revest, Clifford Stein, Introduction To Algorithms, Third Edition. (2009). ISBN 978-0-262-03384-8

[7] D. Cantone, G. Cincotti, QuickHeapsort, an efficient mix of classical sorting algorithms. (2001)

7. Appendix

7.1.การวัดประสิทธิภาพของอัลกอริทึมหมวด Linear search

กำหนดอาร์เรย์ให้เหมาะสมกับปัญหาและครอบคลุม จากนั้นวนรอบเก็บค่าจำนวน loop รอบเพื่อมาหาค่าเฉลี่ย

```
1   import time
2
3   import algorithm_A, algorithm_B, algorithm_C
4
5   R = […]
6   K = …
7   A = B = C = 0
8   loop = …
9   for i in range(loop):
10      # Determine A running time
11      R_A = R.copy()
12      t_A = time.time()
13      algorithm_A.algorithmA(R_A, K)
14      res_A = time.time()
15      A += res_A - t_A
16
17      # Determine B running time
18      R_B = R.copy()
19      t_B = time.time()
20      algorithmB(R_B, K)
21      res_B = time.time()
22      B += res_B - t_B
23
24      # Determine C running time
25      R_C = R
26      R_C.sort()
27      R_C.append(float("inf"))
28      t_C = time.time()
29      algorithmC(R_C, K)
30      res_C = time.time()
31      C += res_C - t_C
32
33  print(A/loop) print(B/loop) print(C/loop)
```

8. ข้อมูลดิบ

I. ข้อมูลกราฟ 6.1

R = [50, 4, 95, 11, 16, 71, 25, 69, 7, 92]

	K = 7 (success):	K = 8 (unsuccess):
A	0.000000391006469726562	0.000000431537628173828
B	0.000000286102294921875	0.000000388622283935547
C	0.000000283718109130859	0.000000205039978027344

II. ข้อมูลกราฟ 6.2

N = 2: [3830.27,8735.12]
K = 6282.695
N = 5: [3293.70, 6748.66, 1373.59, 6102.24, 383.25]
K = 3580.288

	N = 2	N = 5
A	0.00000026702880859375	0.00000041723251342773 4
B	0.00000030517578125	0.00000045537948608398 4
C	0.00000017404556274414 1	0.00000029802322387695 3

III.ข้อมูลกราฟ 6.3

N = 50:
[4320.98, 3462.29, 7518.26, 3553.07, 2663.27, 8686.79, 7866.66, 10.51, 6566.55, 1555.68, 3243.42, 8935.77, 9074.02, 8402.14, 9565.08, 1307.86, 444.56, 8242.63, 8278.06, 2707.24, 8409.89, 4022.06, 4090.39, 9762.19, 6556.12, 855.71, 4125.87, 6406.71, 3901.78, 5015.50, 6922.05, 9333.93, 2662.68, 4017.42, 7800.13, 2257.45, 947.90, 2431.62, 2980.55, 3635.03, 9100.22, 3868.95, 3955.46, 133.68, 3877.84, 5599.82, 1961.97, 1550.99, 9952.37, 8288.60]
K = 5016.5944
N = 500:
[9556.16, 2827.89, 7774.11, 3461.96, 9516.94, 740.23, 7707.11, 1772.03, 9122.75, 1818.00, 4629.81, 9434.31, 3510.64, 1568.89, 8319.42, 6131.99, 8346.04, 3215.70, 1928.94, 7935.03, 7717.14, 4350.35, 7956.79, 9514.60, 3963.54, 9431.31, 2898.77, 2421.98, 1721.67, 7597.23, 8753.41, 2636.94, 4415.13, 5150.61, 7085.70, 9506.75, 1968.72, 3961.16, 7969.56, 1920.61, 1599.93, 4112.29, 9770.16, 7291.18, 1524.82, 9230.57, 9458.55, 4636.69, 8469.18, 2767.15, 1046.37, 4312.92, 3812.93, 5070.07, 6777.28, 9486.14, 4026.90, 5755.27, 443.15, 8326.35, 7057.71, 5010.51, 2289.69, 8978.34, 1702.00, 5642.54, 7370.02, 7279.82, 1503.21, 1212.30, 1651.01, 9818.97, 9163.40, 2907.58, 7279.78, 8417.43, 1996.06, 4191.76, 1433.24, 8517.45, 9192.29, 9153.76, 3589.94, 4997.34, 2586.80, 4398.50, 8921.51, 7604.56, 2567.35, 2317.74, 6317.82, 5392.24, 7968.11, 6285.67, 1636.00, 5321.94, 8292.99, 6681.43, 9904.54, 2324.04, 4936.56, 2404.50, 271.30, 314.04, 7525.82, 6512.24, 8538.39, 5048.58, 7555.85, 4351.89, 473.33, 1910.57, 3746.44, 9557.47, 1437.71, 6219.94, 9433.35, 2746.57, 8971.07, 201.61, 2698.43, 7472.45, 6604.80, 461.23, 4626.73, 5387.31, 1332.22, 3791.43, 5100.67, 2984.55, 2504.46, 1966.18, 5350.19, 2673.17, 6788.03, 7489.00, 2247.27, 9935.68, 7201.96, 1322.86, 5066.06, 6319.10, 2421.20, 5025.25, 2835.36, 6660.89, 4977.67, 8201.22, 9068.61, 3491.98, 1433.65, 2551.42, 5042.81, 1157.75, 158.62, 6795.48, 7885.94, 6129.10, 4706.04, 5835.95, 8215.86, 9858.37, 8443.69, 5859.45, 251.29, 1457.50,

8876.21, 3677.78, 9471.84, 6839.67, 4716.28, 3872.75, 3725.65, 4836.44,
5168.06, 9356.38, 2471.82, 5744.03, 1899.02, 9218.11, 3670.57, 1069.88,
7105.87, 6464.28, 8172.92, 1351.30, 6694.84, 8025.66, 2097.05, 5756.33,
8125.22, 303.10, 3995.05, 2064.03, 1501.17, 9392.16, 2983.47, 8107.81, 4015.29,
9583.70, 1390.91, 846.29, 4041.35, 9719.05, 6107.89, 9058.81, 4804.38, 6671.53,
1246.41, 5823.06, 9619.58, 7547.18, 9832.23, 4218.05, 6781.08, 9313.30,
4314.96, 7548.13, 5555.57, 415.21, 3471.55, 2339.28, 4387.48, 8383.11, 182.12,
3625.45, 6753.28, 4091.98, 7260.58, 5456.55, 7457.64, 3189.27, 1350.07, 581.40,
7401.77, 6190.95, 9159.48, 2028.66, 6357.93, 4317.14, 6986.29, 3456.41,
5983.15, 1731.30, 2599.98, 3033.75, 7484.98, 3231.29, 548.87, 1466.09, 6474.41,
9849.29, 6567.34, 6733.14, 7668.25, 3557.03, 2954.02, 8702.74, 4743.91,
7131.92, 4160.00, 6351.88, 6135.24, 9511.95, 5551.92, 8374.30, 3104.45,
2991.98, 9419.49, 1649.13, 5874.65, 4525.52, 6551.45, 4375.82, 27.40, 8071.02,
2136.89, 3156.98, 5822.20, 5168.86, 1600.06, 1708.46, 2163.42, 6322.16,
3288.91, 5806.40, 5767.24, 835.97, 500.32, 5246.18, 6036.74, 7034.21, 6989.42,
789.21, 4004.34, 9564.33, 1671.25, 5276.91, 218.65, 8323.72, 3769.93, 1082.99,
7050.40, 9415.79, 9706.54, 2588.66, 7401.28, 7470.47, 4896.71, 7907.86,
7850.07, 1381.11, 8083.73, 7173.31, 9350.40, 4995.02, 7725.34, 5189.38,
6598.56, 4230.31, 4166.48, 7889.85, 2358.11, 6805.71, 3751.01, 7867.06,
8569.99, 727.54, 6141.14, 9863.75, 8187.79, 3364.61, 8532.21, 605.61, 9338.99,
5293.34, 5045.74, 150.99, 6798.71, 4711.17, 4839.72, 3372.70, 5667.68, 4350.28,
5733.08, 9904.52, 7989.62, 193.84, 5992.91, 9117.04, 2855.59, 3418.69, 695.83,
5615.21, 2400.68, 950.72, 4836.14, 4453.75, 2402.21, 8348.03, 9415.49, 8586.99,
9044.97, 2402.35, 4972.29, 1979.61, 8836.23, 9054.10, 8756.93, 5.10, 2838.23,
2456.22, 6911.85, 3083.91, 8487.74, 4492.81, 4284.32, 4182.04, 9175.25,
7337.65, 7526.20, 3360.07, 6276.90, 2287.78, 1128.33, 6615.56, 4265.12,
1973.11, 9324.78, 1151.87, 1507.60, 6981.50, 3769.99, 4266.94, 9677.78,
8570.46, 9835.50, 5053.34, 5380.16, 3457.05, 9271.44, 4966.70, 2641.33,
9571.57, 8414.29, 3349.33, 6047.05, 6648.45, 9365.08, 9971.54, 9799.01,
9037.20, 5814.26, 5986.47, 9490.97, 3373.29, 4022.51, 9861.76, 5912.36,
4004.95, 3892.46, 595.56, 6022.51, 6158.34, 1479.14, 381.58, 9486.59, 8118.72,
7565.61, 3594.31, 3662.64, 993.74, 6831.43, 8498.10, 5316.84, 6899.14, 4718.56,
4682.41, 7658.82, 4533.53, 5885.46, 4298.68, 774.16, 7802.62, 5160.48, 7795.19,
8726.58, 9782.26, 588.51, 7391.11, 8643.58, 8054.31, 7831.63, 2605.99, 2204.98,
6238.85, 6589.93, 9121.59, 3611.67, 5185.30, 7210.53, 6270.30, 2502.69,
1788.58, 1635.93, 4545.80, 1011.45, 0.44, 8673.46, 1.57, 4050.42, 3792.89,
2795.66, 3884.23, 3962.25, 3268.89, 3326.17, 8203.47, 5703.09, 4178.30,
5941.66, 624.60, 8252.84, 1418.02, 1938.59, 5625.58, 7312.85, 4410.66, 4798.23,
4293.70, 4288.05, 4904.94, 9966.74, 6358.50, 2466.69, 3970.52, 9845.30, 844.74,
3623.60, 3792.01]

K = 5215.03068

	N = 50	N = 500
A	0.000001430511474609 38	0.000007529258728027 34
B	0.000008440017700195 31	0.000007011890411376 95
C	0.000005173683166503 91	0.000003151893615722 66

IV.ข้อมูลกราฟ 6.4

N = 1000:

[6881.30, 5388.83, 3979.72, 4031.86, 4862.18, 6403.98, 9338.50, 5043.89, 7291.79,
7057.44, 2103.96, 8137.63, 4214.61, 3570.16, 9766.58, 4766.65, 4526.54, 1299.56,
6392.47, 822.79, 8064.02, 1159.53, 7475.60, 4671.68, 3553.65, 1541.86, 9367.73,
676.96, 2102.77, 2724.77, 4745.87, 7483.61, 9895.82, 7069.93, 3765.32, 4567.69,
603.75, 3560.68, 783.41, 6930.12, 3584.78, 3173.29, 4089.29, 2999.15, 2750.00,
4306.01, 9494.77, 427.19, 8036.75, 8983.52, 3882.08, 7455.28, 295.85, 2791.52,
9776.42, 2031.38, 7183.46, 4294.95, 8148.49, 8143.22, 7871.19, 4977.26, 8079.24,
9892.71, 6790.45, 132.15, 334.68, 3229.14, 3898.03, 895.55, 306.84, 3185.68, 5982.85,
7075.26, 2013.11, 4139.97, 9342.43, 9237.58, 1228.83, 3123.89, 4833.27, 7629.73,
2404.98, 7043.48, 5869.31, 2648.11, 9041.36, 1702.81, 9834.18, 7305.66, 4299.35,
1858.37, 8058.38, 3939.15, 5639.41, 4767.91, 926.09, 8383.48, 8612.86, 7649.78,
5746.99, 8855.78, 9728.20, 2257.26, 1656.97, 3525.38, 9074.06, 7844.35, 9840.83,
3715.22, 6201.11, 6742.92, 3213.84, 5882.76, 313.78, 3593.81, 4610.07, 6185.24,
3647.26, 3829.83, 5690.30, 2079.01, 5183.10, 6311.92, 6725.06, 6726.42, 5836.00,
1585.59, 4485.59, 8047.03, 3044.63, 5349.92, 5754.40, 4685.43, 9424.69, 8928.02,

1163.18, 8722.69, 1045.42, 5635.60, 1228.09, 8333.95, 1605.72, 6367.25, 2427.03,
6948.16, 3690.06, 2914.56, 8234.82, 2370.91, 61.55, 5952.73, 5213.10, 3745.01,
9528.47, 154.19, 393.57, 607.57, 9769.38, 4688.16, 3316.67, 6983.58, 9532.97,
2712.46, 9628.74, 6609.84, 3745.58, 4836.06, 1062.84, 4063.00, 7314.52, 5857.45,
8272.41, 958.06, 4878.55, 7028.02, 8952.46, 8246.16, 8084.44, 5663.00, 3093.77,
3679.48, 226.73, 2519.63, 9666.89, 8544.25, 7884.30, 7308.46, 1621.19, 9829.37,
3840.47, 21.03, 6176.98, 7270.55, 8491.56, 8262.21, 4986.27, 1111.70, 9711.67,
4456.09, 8241.03, 8707.20, 8947.21, 2987.56, 6169.59, 8255.89, 8208.84, 3845.69,
4888.20, 4719.25, 4583.41, 1792.53, 265.47, 4482.90, 7238.32, 9338.18, 5361.63,
5419.01, 362.00, 7936.09, 2647.17, 9548.71, 4031.62, 1811.94, 6371.36, 6264.99,
5256.72, 8662.92, 4143.95, 6297.16, 201.04, 5343.23, 6929.09, 2022.60, 9223.82,
8215.81, 3331.48, 9316.99, 1665.74, 3236.54, 9199.96, 1813.64, 7968.16, 4030.52,
4267.08, 3656.01, 5253.59, 9671.46, 4705.22, 9597.21, 5634.28, 5438.80, 1863.36,
9579.13, 2077.04, 4320.56, 4703.07, 2534.44, 8120.67, 2314.64, 6464.84, 3879.56,
2166.13, 9631.16, 3683.80, 8657.97, 9150.87, 6614.95, 1327.44, 6081.59, 9561.14,
5142.45, 4766.53, 5301.19, 8901.51, 3063.46, 77.60, 2684.62, 863.51, 1885.71,
6423.88, 9466.91, 5553.63, 131.90, 694.68, 7407.42, 2040.82, 8543.32, 1314.21,
8111.49, 2029.27, 73.38, 9126.56, 1585.22, 4100.18, 8369.86, 9167.59, 3527.63,
9490.38, 5763.04, 2119.18, 1221.70, 8042.53, 2885.81, 4741.12, 7692.25, 5577.35,
4761.89, 6002.69, 9646.94, 3454.92, 9233.57, 457.46, 7564.86, 8777.64, 9814.59,
6673.10, 2099.28, 1558.47, 7118.76, 1570.98, 4924.43, 2062.49, 2033.97, 9936.39,
7064.22, 2402.80, 8296.62, 30.31, 5214.67, 237.53, 1040.37, 7523.17, 5127.16,
4135.44, 2851.52, 9036.83, 3324.25, 2467.34, 8440.83, 3917.21, 2643.73, 6458.34,
8916.42, 8517.41, 3447.01, 1659.56, 4628.68, 4069.86, 3292.92, 2573.80, 8062.93,
4004.85, 9832.25, 2694.60, 8458.78, 7135.26, 2201.77, 460.64, 3738.90, 3356.91,
8553.12, 110.06, 5254.87, 343.89, 7142.73, 2877.10, 353.55, 4729.58, 1635.03,
4147.35, 8886.35, 5435.94, 3248.78, 4614.66, 4904.00, 276.97, 7964.95, 8289.92,
1808.34, 9802.48, 9190.46, 8904.50, 662.87, 392.06, 7504.54, 9935.20, 8752.27,
9854.72, 3328.02, 4231.87, 1468.33, 6632.09, 6168.88, 4054.43, 2620.45, 1594.43,
7927.61, 5776.90, 3506.78, 2178.42, 286.53, 7349.04, 783.26, 7399.97, 7436.37,
3683.72, 1232.22, 7343.00, 8225.18, 1523.54, 9702.28, 2617.70, 9844.31, 3883.54,
3963.20, 3597.88, 3729.64, 3889.64, 3326.95, 1554.05, 4508.18, 5764.18, 9891.97,
6286.83, 1223.68, 197.78, 2412.10, 1231.94, 5046.49, 219.55, 6446.41, 4456.90,
9724.81, 1903.71, 8074.27, 2363.18, 5881.94, 4386.48, 7342.50, 3680.69, 890.28,
3621.38, 3891.06, 6898.91, 5295.29, 6508.09, 6800.70, 9587.48, 26.32, 9666.84,
2314.96, 787.24, 3155.32, 6665.60, 5602.48, 5600.74, 8829.28, 3225.86, 8991.77,
820.58, 9858.69, 8004.34, 732.58, 346.89, 6869.29, 5586.35, 9840.38, 1114.97, 190.04,
3400.81, 693.51, 9193.40, 1627.49, 21.39, 3232.79, 8397.09, 174.28, 2037.85, 3267.27,
8881.23, 9335.66, 4837.75, 6413.84, 9160.54, 7971.53, 3214.24, 7529.83, 5287.34,
3947.28, 8851.35, 3367.52, 8916.50, 779.95, 9392.67, 6706.31, 8210.56, 1439.85,
7223.62, 6015.49, 4670.50, 7412.60, 6764.15, 7244.26, 7752.12, 3200.56, 1114.77,
7053.95, 5988.74, 5411.58, 3800.21, 9937.21, 1383.77, 8315.79, 5692.24, 2108.13,
327.16, 3646.06, 5874.02, 2448.89, 7248.57, 2251.95, 5221.35, 1214.45, 6571.39,
518.45, 9427.75, 9776.71, 3285.89, 3599.72, 5751.80, 7413.81, 7284.27, 4279.42,
7471.65, 6248.43, 2168.27, 477.93, 1324.09, 3966.46, 426.21, 8776.04, 9871.16,
6599.31, 9489.48, 1594.68, 5957.14, 5254.62, 8696.62, 1700.46, 1377.50, 5868.05,
2069.22, 2624.29, 3558.15, 5952.41, 632.34, 7650.60, 9231.17, 2333.56, 2398.90,
1952.41, 8165.07, 547.98, 4116.86, 8360.14, 9427.03, 9488.02, 9522.17, 7297.91,
1293.25, 9104.99, 1965.98, 5974.02, 9965.56, 9460.56, 9394.22, 4284.52, 3117.73,
3589.37, 8227.99, 2705.29, 6480.85, 8456.66, 6169.03, 6764.28, 946.80, 9647.91,
3139.12, 8607.40, 2109.13, 699.83, 4075.20, 42.10, 8191.37, 1992.73, 7605.12,
2350.45, 3682.18, 9937.04, 1036.95, 2952.91, 1182.69, 6000.55, 5613.27, 4699.64,
368.23, 7305.55, 9072.37, 2188.82, 7636.62, 2932.17, 3531.99, 2221.74, 7846.38,
898.91, 5443.79, 6511.92, 2670.95, 1611.11, 5666.00, 3974.13, 9824.27, 7224.05,
1044.09, 8039.99, 7395.09, 8357.95, 2491.11, 6565.89, 6271.41, 9719.51, 5510.62,
5766.48, 3225.71, 324.54, 6931.69, 2344.06, 8912.42, 9002.74, 72.83, 7421.74,
1460.96, 2375.29, 3948.31, 1704.30, 8353.87, 8019.21, 2084.47, 9420.76, 5272.18,
3323.53, 1116.26, 6799.72, 1017.25, 2318.25, 2996.19, 8446.30, 3979.57, 6239.35,
499.41, 4233.64, 5776.45, 2685.07, 6973.19, 607.47, 3664.53, 9824.10, 1408.57,
8856.60, 4736.08, 5382.07, 5620.21, 5883.44, 8143.47, 1048.29, 1382.58, 590.67,
5134.40, 8748.16, 2507.83, 3409.69, 8577.65, 2866.12, 651.56, 2821.46, 746.51,
2931.57, 5812.83, 3650.43, 8843.34, 7979.00, 3100.19, 681.34, 4437.34, 90.40,
7396.95, 9306.96, 5743.54, 3996.70, 9294.38, 2931.70, 1904.30, 7084.30, 4830.45,
506.56, 2020.13, 2987.41, 4826.98, 1606.80, 9140.97, 9948.43, 3759.83, 370.66,
5490.24, 2794.52, 7228.53, 3848.84, 9361.22, 8686.17, 5035.30, 293.00, 9109.39,
8865.64, 9191.00, 8124.27, 9031.89, 445.28, 4961.97, 8014.09, 9595.96, 3105.51,
7978.04, 7902.73, 2117.03, 3692.48, 7185.11, 4804.88, 3246.15, 4812.52, 1097.61,

946.32, 4827.42, 6783.05, 5901.80, 8276.27, 5925.60, 3616.38, 261.99, 5366.01,
8828.69, 1206.28, 4529.29, 3017.05, 3989.86, 3347.29, 4363.77, 3830.78, 2673.70,
8639.25, 7400.73, 4338.74, 2903.57, 6301.81, 2600.22, 9381.45, 248.05, 8337.89,
1926.57, 2322.84, 4219.02, 9989.53, 465.88, 3455.39, 4765.03, 2014.17, 733.05,
6718.75, 8391.46, 1221.92, 4845.09, 6126.33, 9954.66, 8443.52, 252.66, 1442.91,
5952.24, 8245.03, 9882.46, 9265.34, 1111.54, 1.63, 3134.64, 1679.85, 6548.53,
7448.36, 4664.22, 1349.86, 1529.03, 4264.88, 1762.41, 4330.58, 1457.39, 6825.94,
3934.55, 4638.43, 9202.24, 9714.59, 1713.53, 6889.45, 4747.87, 5585.91, 208.67,
4600.21, 9922.96, 1814.27, 9753.04, 8679.11, 4429.63, 8479.23, 8418.03, 4719.42,
1699.60, 1764.99, 5168.71, 8469.11, 7467.28, 9332.98, 3170.47, 3114.48, 1256.74,
2308.74, 247.07, 4220.40, 561.52, 4885.84, 2752.49, 4623.76, 1206.24, 7312.97,
4994.35, 5010.83, 6839.82, 1205.33, 4353.10, 9108.59, 786.88, 2944.19, 1812.69,
7426.41, 4399.44, 1053.15, 2370.37, 1453.36, 1109.30, 6.65, 7135.16, 7216.96,
8579.42, 2144.96, 3470.00, 9438.66, 2619.55, 6995.19, 8269.45, 8111.60, 752.52,
771.05, 8824.56, 3507.68, 8950.64, 9194.47, 2820.50, 5081.77, 9948.44, 9578.98,
6268.73, 3897.77, 570.96, 4223.42, 8876.75, 3886.38, 7132.18, 6131.33, 8458.72,
5920.70, 9354.18, 7792.55, 4055.29, 4437.60, 5462.27, 3103.93, 5367.01, 3220.72,
6301.25, 6072.03, 9631.44, 7793.62, 8142.47, 4979.60, 6188.26, 6658.44, 2772.92,
6426.64, 361.62, 2088.38, 3652.86, 7781.86, 5976.39, 6518.74, 2298.28, 4022.32,
9005.50, 2206.61, 2897.06, 8244.15, 8333.71, 9282.13, 4227.79, 8842.99, 4285.85,
2747.47, 9501.62, 3799.54, 5745.67, 7908.20, 3732.28, 9955.46, 3402.67, 4269.60,
2246.09, 9306.14, 1993.98, 7020.12, 9215.93, 3944.71, 211.05, 5528.61, 1853.49,
2158.31, 452.98, 6534.05, 8720.57, 7379.41, 9921.25, 7455.89, 1809.33, 3473.30,
2250.03, 463.81, 4191.65, 9736.79, 8399.42, 5286.88, 6239.72, 6580.08, 6730.10,
6506.65, 3063.91, 8592.14, 9774.94, 1964.78, 4930.19, 3774.05, 946.62, 3850.79,
7666.11, 5945.35, 3073.64, 7447.07, 1535.73, 1876.82, 2513.70, 5481.80, 291.88,
4805.62, 6581.93, 8177.11, 3262.22, 2005.04, 2944.33, 98.05, 3720.88, 4105.87,
1763.07, 418.81, 707.37, 5924.30, 6921.10, 9077.61, 353.43, 9945.63, 9082.06,
7824.38, 7010.90, 9764.22, 7773.32, 2991.89, 5102.14]

K = 5013.10626626627

N = 1000

A 0.000014960765838623

B 0.0000142908096313477

C 0.00000680923461914062

V. ข้อมูลกราฟ 6.5

N = 50 (return 50.11985634456667)

126, 2, 449, 866, 687, 749, 447, 135, 216, 257, 192, 542, 881, 222, 4, 202, 896, 879,
247, 384, 2, 446, 17, 987, 602, 163, 330, 860, 8, 730, 69, 225, 794, 145, 685, 211,
694, 805, 534, 266, 820, 651, 188, 924, 162, 53, 781, 209, 599, 256, 298, 711, 379,
24, 594, 402, 362, 928, 990, 190, 626, 486, 483, 338, 544, 553, 938, 979, 138, 214,
651, 18, 957, 461, 584, 855, 925, 65, 174, 847, 146, 255, 818, 189, 979, 272, 727,
104, 736, 846, 210, 147, 35, 877, 507, 149, 785, 717, 604, 384

N = 100 (return 22.847319317591726)

753, 508, 968, 718, 760, 75, 550, 426, 259, 916, 639, 30, 454, 402, 154, 313, 292,
701, 525, 398, 516, 272, 849, 530, 942, 582, 751, 159, 139, 912, 633, 612, 754, 704,
94, 104, 858, 679, 621, 520, 213, 568, 127, 230, 791, 791, 112, 424, 266, 797, 76,
38, 176, 940, 119, 751, 847, 324, 666, 369, 621, 804, 704, 58, 6, 454, 336, 263, 102,
471, 510, 158, 133, 787, 52, 519, 292, 78, 374, 907, 170, 901, 897, 167, 624, 873,
868, 630, 510, 837, 42, 458, 620, 516, 988, 180, 253, 317, 581, 910

N = 500 (return 5.0)

508, 1, 960, 571, 813, 657, 161, 761, 472, 44, 383, 707, 533, 828, 612, 677, 465,
664, 12, 413, 837, 857, 799, 681, 868, 964, 180, 942, 407, 262, 403, 70, 591, 234,
828, 519, 711, 628, 893, 270, 587, 688, 833, 824, 730, 109, 48, 42, 316, 315, 246,
513, 223, 89, 578, 902, 235, 936, 792, 915, 852, 904, 798, 237, 428, 146, 100, 43,
21, 777, 630, 195, 21, 19, 819, 568, 34, 513, 295, 70, 705, 447, 887, 64, 868, 758,
204, 221, 494, 656, 649, 842, 934, 747, 235, 150, 934, 417, 265, 956, 512, 614, 780,
889, 921, 430, 166, 411, 700, 957, 568, 805, 43, 758, 5, 410, 447, 443, 190, 952,
236, 215, 37, 56, 23, 361, 741, 64, 18, 414, 358, 424, 619, 682, 144, 341, 820, 729,
228, 739, 899, 820, 644, 196, 53, 542, 658, 480, 834, 673, 229, 300, 535, 752, 364,
858, 779, 645, 233, 775, 804, 708, 226, 349, 513, 905, 147, 917, 734, 695, 560, 411,

310, 530, 797, 295, 775, 913, 973, 208, 331, 229, 399, 482, 691, 957, 256, 447, 587,
885, 150, 415, 895, 418, 252, 182, 114, 54, 624, 773, 371, 352, 271, 196, 348, 5,
357, 99, 432, 886, 735, 879, 215, 825, 303, 981, 521, 410, 33, 112, 277, 200, 936,
721, 743, 281, 471, 472, 251, 655, 264, 526, 394, 102, 705, 604, 510, 231, 869, 455,
447, 911, 63, 147, 996, 429, 710, 225, 20, 789, 483, 953, 650, 159, 423, 973, 873,
75, 77, 232, 910, 744, 986, 151, 396, 833, 308, 674, 455, 576, 200, 653, 847, 70,
737, 835, 802, 321, 258, 159, 624, 22, 320, 66, 803, 719, 471, 207, 555, 653, 544,
628, 380, 715, 508, 261, 682, 843, 899, 574, 243, 280, 194, 843, 105, 687, 667, 119,
496, 831, 454, 228, 288, 253, 228, 52, 470, 318, 892, 969, 159, 851, 936, 55, 920,
704, 431, 878, 184, 138, 432, 48, 123, 705, 593, 275, 65, 690, 615, 430, 910, 591,
122, 486, 300, 742, 126, 243, 417, 237, 180, 859, 480, 342, 509, 329, 632, 466, 866,
61, 134, 37, 435, 935, 92, 33, 724, 608, 848, 324, 552, 2, 339, 674, 892, 67, 868,
384, 63, 750, 654, 120, 604, 232, 752, 933, 894, 724, 184, 591, 454, 998, 564, 29,
288, 898, 785, 219, 922, 925, 224, 722, 91, 917, 451, 924, 304, 11, 491, 142, 293,
395, 517, 194, 521, 920, 636, 592, 361, 580, 567, 875, 670, 524, 416, 93, 518, 10,
738, 4, 774, 888, 482, 175, 286, 558, 284, 480, 401, 383, 446, 336, 758, 823, 931,
365, 992, 757, 193, 759, 422, 540, 884, 51, 744, 964, 446, 697, 153, 400, 213, 399,
717, 347, 15, 607, 542, 774, 811, 675, 229, 953, 878, 812, 664, 441, 8, 828, 241,
390, 990, 753, 235, 521, 182, 39, 333, 30, 561, 326, 542, 943, 749, 89, 793, 849,
742, 869, 435, 156, 446, 392, 720, 357, 315, 791, 452, 591, 213, 805, 359, 564, 208,
503, 293, 604, 753, 543, 117, 690, 331, 303, 78, 526, 143, 908, 41, 920, 899, 324,
794, 947, 224, 547, 301, 446, 909, 681, 415, 802, 189, 455, 949, 428, 188, 127, 323,
48, 192, 869, 564, 363, 642, 207, 298, 715, 708, 219, 66, 614, 676, 405, 150, 218,
360, 271, 834, 59, 894, 648, 228, 355, 636, 449, 786, 397, 690, 182, 239, 232, 300,
285, 789, 215, 576, 695, 318, 80, 131, 597, 281, 134, 590, 596, 245, 332, 279, 570,
889, 624, 736, 932, 650, 232, 820, 937, 491, 779, 12, 171, 808, 667, 913, 219, 894,
770, 531, 732, 612, 177, 182, 580, 621, 593, 772, 538, 113, 621, 805, 472, 855, 992,
175, 914, 799, 109, 701, 410, 281, 291, 934, 255, 330, 762, 264, 775, 511, 775, 400,
706, 646, 767, 953, 227, 63, 405, 357, 632, 664, 261, 55, 611, 518, 333, 289, 348,
44, 471, 999, 18, 928, 992, 803, 30, 179, 607, 353, 188, 852, 651, 275, 455, 142,
289, 613, 377, 18, 493, 419, 620, 178, 895, 514, 271, 899, 220, 257, 310, 170, 438,
60, 859, 158, 216, 918, 943, 832, 683, 199, 512, 784, 779, 432, 382, 217, 404, 170,
468, 151, 588, 2, 230, 235, 205, 384, 542, 86, 848, 471, 915, 800, 846, 307, 303,
472, 34, 468, 692, 649, 805, 838, 910, 323, 664, 222, 809, 34, 570, 282, 532, 652,
171, 268, 347, 958, 106, 908, 701, 878, 584, 783, 145, 3, 538, 2, 208, 257, 837, 404,
477, 806, 766, 131, 966, 255, 235, 72, 101, 721, 310, 88, 598, 29, 287, 728, 586,
165, 580, 60, 7, 630, 249, 613, 557, 857, 303, 710, 735, 522, 627, 484, 899, 238,
360, 270, 666, 551, 864, 648, 831, 789, 975, 618, 437, 619, 457, 302, 47, 257, 996,
894, 972, 169, 221, 222, 642, 94, 632, 635, 279, 884, 754, 646, 147, 555, 487, 126,
244, 323, 679, 958, 942, 911, 171, 826, 308, 794, 11, 104, 46, 995, 472, 615, 151,
438, 825, 803, 896, 824, 95, 499, 505, 323, 25, 218, 704, 406, 201, 988, 500, 855,
502, 855, 358, 758, 650, 162, 128, 954, 17, 936, 138, 83, 240, 701, 480, 771, 592,
590, 145, 934, 137, 13, 997, 891, 397, 260, 613, 993, 770, 653, 408, 211, 736, 460,
475, 303, 301, 904, 976, 926, 429, 195, 181, 424, 777, 299, 945, 688, 980, 71, 81,
335, 730, 16, 835, 675, 486, 902, 311, 265, 469, 340, 959, 742, 793, 643, 997, 476,
777, 220, 167, 355, 772, 219, 679, 755, 367, 981, 126, 748, 912, 279, 686, 164, 911,
6, 206, 409, 85, 10, 193, 312, 251, 123, 105, 31, 781, 563, 222, 202, 193, 996, 398,
621, 295, 288, 3, 450, 733, 465, 859, 857, 94, 736, 851, 136, 682, 310, 995, 539,
497, 824, 163, 220, 354, 751, 11, 746, 474

N = 1000 (return 2.0)

62, 464, 498, 90, 889, 963, 982, 218, 665, 780, 180, 277, 985, 544, 437, 350, 536,
159, 877, 283, 141, 326, 928, 514, 823, 383, 298, 260, 880, 986, 928, 268, 108, 707,
639, 423, 174, 746, 110, 518, 735, 690, 972, 471, 531, 889, 851, 904, 59, 611, 637,
118, 206, 319, 856, 764, 245, 386, 652, 123, 737, 40, 630, 271, 900, 869, 249, 972,
920, 965, 327, 405, 230, 519, 813, 172, 930, 985, 662, 135, 877, 225, 889, 30, 909,
258, 535, 12, 260, 469, 755, 202, 800, 271, 845, 395, 909, 301, 904, 223, 285, 841,
155, 434, 88, 709, 901, 14, 209, 54, 651, 703, 565, 666, 271, 257, 65, 107, 615, 772,
883, 175, 377, 741, 115, 701, 983, 529, 995, 723, 494, 203, 617, 158, 344, 259, 239,
183, 493, 246, 40, 767, 18, 944, 995, 368, 827, 110, 532, 753, 897, 474, 689, 816,
388, 78, 398, 318, 208, 610, 927, 55, 585, 140, 497, 822, 599, 26, 74, 153, 254, 652,
310, 608, 526, 581, 220, 554, 411, 241, 727, 972, 712, 30, 760, 158, 823, 32, 485,
260, 111, 705, 719, 162, 260, 957, 237, 11, 56, 601, 336, 615, 112, 505, 58, 729,
666, 457, 256, 138, 743, 62, 790, 901, 280, 995, 872, 831, 624, 361, 543, 801, 622,
190, 26, 255, 355, 978, 691, 933, 864, 82, 926, 509, 857, 249, 626, 781, 955, 220,
471, 504, 240, 541, 441, 346, 493, 40, 721, 132, 827, 169, 204, 674, 945, 422, 303,
831, 847, 444, 113, 699, 553, 549, 541, 186, 834, 244, 432, 283, 344, 531, 912, 303,
266, 847, 106, 502, 717, 517, 805, 997, 986, 962, 855, 890, 641, 521, 402, 692, 927,

837, 287, 753, 969, 850, 213, 804, 1, 772, 726, 440, 770, 672, 35, 985, 879, 995,
990, 375, 991, 300, 540, 661, 514, 757, 132, 216, 507, 675, 454, 733, 545, 885, 596,
972, 923, 572, 642, 374, 424, 404, 40, 650, 222, 724, 191, 174, 674, 700, 387, 821,
121, 718, 797, 92, 602, 605, 503, 168, 505, 916, 686, 922, 523, 145, 47, 352, 965,
547, 599, 28, 600, 92, 29, 776, 292, 179, 631, 712, 318, 171, 704, 328, 525, 84, 275,
699, 23, 568, 56, 140, 993, 524, 541, 394, 487, 204, 914, 865, 326, 654, 677, 297,
91, 186, 154, 558, 580, 817, 640, 122, 266, 698, 86, 839, 792, 197, 652, 291, 982,
780, 462, 434, 174, 946, 822, 814, 571, 64, 357, 558, 679, 530, 735, 424, 35, 145,
420, 463, 943, 989, 469, 312, 38, 573, 106, 886, 925, 476, 469, 452, 227, 353, 323,
526, 111, 619, 880, 236, 605, 413, 364, 120, 331, 197, 754, 616, 904, 631, 135, 948,
957, 91, 95, 20, 197, 611, 748, 232, 727, 732, 337, 547, 88, 386, 612, 453, 809, 583,
517, 317, 152, 425, 923, 83, 647, 242, 874, 630, 250, 447, 376, 823, 957, 766, 833,
201, 183, 404, 136, 132, 114, 236, 720, 475, 379, 314, 882, 340, 271, 637, 548, 330,
918, 723, 366, 696, 7, 947, 99, 90, 244, 790, 581, 216, 140, 643, 778, 4, 768, 931,
807, 202, 475, 25, 931, 893, 486, 563, 437, 352, 741, 229, 745, 61, 86, 391, 293,
373, 611, 2, 710, 556, 917, 813, 36, 48, 632, 899, 228, 686, 397, 942, 186, 723, 103,
24, 506, 133, 40, 40, 291, 336, 747, 324, 396, 916, 960, 667, 48, 136, 347, 655, 22,
489, 189, 962, 54, 240, 899, 972, 699, 603, 111, 375, 578, 238, 298, 546, 808, 58,
641, 582, 590, 59, 772, 649, 304, 807, 158, 904, 912, 626, 983, 516, 587, 618, 6,
787, 676, 679, 878, 993, 488, 964, 928, 516, 532, 81, 497, 272, 714, 36, 647, 891,
607, 293, 488, 631, 838, 985, 137, 954, 337, 260, 279, 131, 982, 79, 400, 503, 487,
628, 879, 614, 787, 658, 822, 889, 32, 742, 642, 480, 54, 672, 371, 437, 18, 373,
121, 551, 959, 196, 59, 940, 538, 902, 57, 703, 693, 354, 8, 202, 970, 639, 230, 860,
383, 963, 92, 198, 585, 657, 89, 598, 636, 665, 753, 570, 49, 492, 961, 839, 916,
782, 215, 269, 692, 603, 930, 256, 396, 432, 780, 203, 46, 31, 6, 525, 422, 448, 300,
768, 827, 773, 534, 150, 531, 306, 938, 981, 75, 331, 137, 259, 71, 276, 293, 764,
468, 539, 978, 969, 925, 979, 236, 451, 663, 45, 952, 707, 749, 533, 988, 742, 712,
271, 206, 127, 743, 19, 52, 592, 327, 642, 606, 496, 251, 477, 791, 248, 937, 113,
158, 473, 849, 93, 137, 802, 37, 246, 588, 213, 628, 372, 873, 449, 633, 398, 35,
329, 863, 298, 57, 516, 245, 374, 744, 909, 46, 295, 9, 212, 111, 0, 983, 748, 125,
294, 907, 849, 188, 114, 187, 472, 369, 397, 419, 370, 18, 803, 274, 589, 591, 924,
772, 924, 952, 866, 977, 304, 184, 245, 460, 954, 388, 46, 139, 139, 909, 421, 736,
260, 426, 224, 444, 425, 83, 906, 523, 96, 770, 98, 602, 763, 154, 795, 519, 514, 35,
844, 428, 102, 985, 595, 142, 699, 652, 738, 517, 891, 775, 860, 933, 146, 444, 9,
215, 761, 742, 196, 134, 536, 228, 216, 701, 945, 556, 11, 426, 940, 657, 41, 630,
101, 232, 862, 41, 666, 769, 591, 80, 608, 597, 451, 988, 711, 233, 242, 135, 375,
934, 686, 560, 151, 551, 85, 165, 885, 847, 385, 774, 408, 183, 508, 472, 794, 866,
641, 148, 41, 978, 736, 90, 512, 225, 521, 931, 912, 729, 914, 523, 135, 703, 172,
705, 23, 890, 104, 41, 816, 472, 132, 480, 42, 865, 677, 709, 315, 539, 22, 182, 50,
441, 780, 651, 637, 596, 602, 282, 988, 294, 259, 677, 175, 224, 745, 383, 297, 852,
376, 743, 860, 679, 612, 499, 426, 304, 167, 351, 254, 839, 862, 282, 305, 17, 360,
118, 657, 353, 917, 755, 442, 513, 336, 17, 661, 347, 240, 58, 276, 266, 854, 276,
197, 623, 135, 3, 744, 766, 865, 266, 197, 114, 200, 739, 503, 203, 516, 291, 461,
896, 325, 108, 259, 43, 937, 619, 867, 102, 290, 921, 322, 476, 994, 496, 191, 856,
456, 298, 915, 571, 470, 542, 578, 179, 661, 294, 502, 239, 13, 697, 746, 80, 419,
826, 219, 650, 883, 607, 501, 767, 594, 418, 877, 518, 510, 345, 717, 279, 819, 100,
800, 881, 978, 287, 587, 146, 997, 535, 111, 100, 805, 941, 211, 977, 79, 197, 388,
182, 888, 585, 262, 823, 230, 798, 477, 722, 833, 296, 809, 722, 677, 672, 695, 656,
397, 133, 691, 874, 767, 219, 656, 489, 528, 755, 144, 677, 296, 176, 532, 568, 329,
134, 192, 709, 948, 916, 820, 459, 733, 993, 488, 905, 498, 993, 901, 516, 923, 656,
341, 372, 288, 533, 300, 318, 86, 899, 709, 752, 645, 251, 414, 847, 773, 147, 833,
471, 399, 621, 741, 535, 516, 179, 834, 869, 889, 914, 644, 191, 73, 965, 580, 127,
998, 715, 732, 984, 723, 943, 944, 375, 72, 802, 569, 891, 111, 220, 771, 897, 814,
779, 91, 68, 465, 170, 893, 538, 272, 200, 297, 806, 581, 473, 171, 229, 283, 595,
152, 717, 480, 59, 554, 42, 65, 128, 837, 184, 242, 716, 975, 609, 327, 31, 403, 490,
18, 756, 980, 618, 295, 867, 223, 233, 446, 67, 414, 644, 863, 124, 650, 888, 462,
686, 594, 620, 558, 773, 736, 614, 643, 968, 559, 29, 69, 284, 346, 691, 870, 386,
429, 169, 484, 988, 659, 258, 658, 134, 706, 537, 9, 79, 94, 495, 179, 266, 733, 341,
506, 601, 871, 840, 898, 400, 786, 388, 278, 615, 714, 659, 635, 307, 125, 344, 730,
173, 504, 987, 920, 10, 361, 799, 567, 381, 29, 221, 99, 262, 38, 99, 417, 136, 208,
730, 384, 922, 199, 157, 752, 485, 595, 134, 386, 251, 697, 948, 449, 600, 795, 15,
264, 981, 926, 665, 908, 195, 199, 404, 351, 319, 214, 433, 883, 386, 781, 580, 320,
500, 853, 728, 809, 206, 673, 987, 621, 153, 919, 808, 234, 615, 87, 988, 745, 60,
536, 25, 884, 388, 633, 663, 345, 831, 6, 396, 445, 209, 896, 523, 484, 668, 508,
691, 12, 489, 722, 78, 584, 814, 108, 413, 951, 270, 829, 173, 981, 803, 401, 34,
559, 651, 937, 608, 949, 213, 649, 91, 863, 331, 471, 398, 981, 350, 352, 63, 736,
197, 709, 983, 330, 990, 587, 41, 365, 840, 121, 216, 921, 100, 682, 796, 770, 554,

17, 153, 468, 572, 305, 151, 507, 523, 300, 512, 285, 116, 874, 138, 878, 11, 177,
277, 249, 321, 450, 224, 708, 48, 234, 703, 802, 782, 319, 412, 382, 824, 174, 905,
531, 271, 695, 116, 526, 204, 618, 309, 956, 500, 391, 652, 675, 718, 984, 800, 998,
222, 679, 510, 932, 858, 909, 51, 334, 843, 584, 259, 891, 474, 529, 912, 859, 102,
716, 586, 747, 337, 55, 901, 92, 180, 589, 647, 836, 0, 616, 588, 589, 150, 691, 649,
258, 244, 648, 874, 176, 431, 74, 642, 513, 247, 711, 856, 575, 686, 769, 994, 660,
447, 521, 359, 813, 608, 213, 494, 70, 667, 914, 923, 871, 867, 705, 568, 10, 984,
848, 174, 4, 994, 368, 205, 376, 101, 822, 864, 61, 565, 865, 900, 639, 698, 760,
718, 134, 768, 909, 814, 681, 985, 473, 862, 425, 37, 470, 364, 630, 285, 675, 966,
779, 52, 554, 272, 124, 649, 585, 797, 127, 159, 948, 959, 526, 376, 170, 148, 210,
147, 110, 343, 704, 921, 0, 180, 828, 412, 39, 371, 150, 40, 399, 862, 58, 41, 382,
21, 360, 94, 791, 882, 708, 493, 463, 55, 303, 320, 982, 998, 674, 594, 472, 316,
358, 446, 414, 464, 833, 743, 775, 577, 370, 224, 801, 725, 121, 183, 226, 521, 663,
291, 463, 21, 763, 194, 64, 517, 979, 774, 51, 604, 895, 268, 547, 168, 608, 577,
700, 213, 815, 251, 20, 580, 764, 671, 528, 890, 128, 966, 160, 69, 929, 524, 257,
292, 840, 718, 749, 564, 688, 604, 117, 798, 540, 879, 184, 433, 798, 95, 585, 339,
332, 154, 696, 153, 577, 555, 430, 208, 664, 15, 131, 936, 453, 766, 118, 615, 450,
408, 413, 498, 411, 28, 940, 475, 385, 80, 917, 260, 753, 135, 850, 118, 381, 852,
106, 655, 568, 285, 146, 995, 287, 280, 794, 987, 58, 969, 184, 661, 200, 65, 974,
734, 764, 373, 393, 195, 887, 574, 94, 20, 831, 303, 649, 660, 58, 307, 70, 104, 692,
542, 174, 377, 243, 689, 209, 769, 267, 291, 353, 288, 475, 782, 578, 986, 872, 98,
880, 354, 858, 386, 712, 747, 398, 601, 756, 16, 435, 296, 704, 745, 919, 384, 490,
633, 276, 26, 763, 778, 957, 112, 303, 894, 18, 349, 761, 497, 911, 80, 882, 92, 782,
811, 434, 425, 372, 395, 974, 364, 108, 334, 357, 473, 440, 444, 589, 694, 509, 61,
111, 345, 768, 480, 293, 782, 87, 590, 913, 718, 755, 496, 871, 949, 1, 951, 76, 147,
964, 212, 153, 210, 480, 890, 348, 780, 192, 347, 788, 773, 379, 814, 480, 625, 374,
897, 472, 928, 411, 534, 410, 410, 542, 524, 931, 883, 49, 596, 650, 206, 534, 87,
227, 137, 752, 136, 621, 49, 899, 628, 225, 679, 176, 855, 390, 517, 967, 526, 45,
416, 251, 582, 345, 505, 667, 178, 244, 687, 379, 836, 560, 880, 582, 352, 320, 877,
864, 305, 94, 218, 948, 741, 100, 761, 519, 121, 295, 962, 5, 212, 748, 787, 732,
808, 393, 789, 308, 952, 538, 71, 57, 835, 980, 219, 13, 595, 655, 419, 439, 826,
986, 434, 566, 659, 431, 550, 536, 782, 592, 613, 98, 263, 306, 156, 569, 911, 733,
705, 847, 378, 412, 492, 757, 236, 683

N	closest pair divide and conquer	closest pair brute force
50	0.00219588399999999	0.003813665
100	0.009244388	0.009461172
500	0.0416223479999999	0.212103333999999
1000	0.093488789	0.803883867

VI.ข้อมูลกราฟ 6.6 และ 6.7

i. Unique

N = 10

8.36E+02, 2.75E+02, 1.89E+02, 1.75E+02, 8.89E+02, 9.34E+02, 4.97E+02, 5.50E+02,
4.40E+01, 7.50E+01,

N = 50

1.58E+02, 9.27E+02, 6.57E+02, 3.50E+02, 2.29E+02, 2.56E+02, 1.92E+02, 6.35E+02,
7.41E+02, 2.80E+02, 7.37E+02, 8.64E+02, 4.50E+02, 3.33E+02, 4.31E+02, 1.30E+02,
7.71E+02, 3.51E+02, 2.85E+02, 2.69E+02, 8.82E+02, 3.80E+01, 3.00E+02, 4.53E+02,
7.16E+02, 7.00E+00, 1.28E+02, 9.45E+02, 3.28E+02, 2.98E+02, 6.63E+02, 5.45E+02,
7.08E+02, 2.31E+02, 8.79E+02, 3.54E+02, 6.81E+02, 1.77E+02, 8.44E+02, 6.50E+02,
7.76E+02, 4.82E+02, 8.06E+02, 1.85E+02, 8.62E+02, 8.07E+02, 7.99E+02, 5.48E+02,
6.88E+02, 4.10E+01,

N = 100

3.72E+02, 3.95E+02, 6.39E+02, 9.62E+02, 6.50E+01, 4.48E+02, 9.64E+02, 7.50E+01,
6.26E+02, 3.86E+02, 6.57E+02, 9.59E+02, 7.00E+00, 7.68E+02, 8.79E+02, 9.76E+02,
9.94E+02, 5.61E+02, 7.56E+02, 4.50E+02, 8.96E+02, 6.85E+02, 4.10E+02, 8.62E+02,
8.05E+02, 3.27E+02, 3.80E+02, 9.14E+02, 5.41E+02, 3.08E+02, 5.01E+02, 5.17E+02,
2.32E+02, 4.60E+01, 1.07E+02, 7.20E+01, 6.41E+02, 4.44E+02, 6.27E+02, 6.84E+02,
8.87E+02, 6.86E+02, 4.53E+02, 7.88E+02, 5.60E+01, 1.01E+02, 2.79E+02, 1.96E+02,
3.92E+02, 3.46E+02, 8.30E+02, 7.80E+02, 8.31E+02, 4.28E+02, 5.35E+02, 4.57E+02,
2.44E+02, 8.95E+02, 8.65E+02, 5.48E+02, 5.06E+02, 9.12E+02, 8.41E+02, 8.99E+02,
6.73E+02, 2.65E+02, 3.35E+02, 4.58E+02, 4.98E+02, 1.65E+02, 9.65E+02, 4.95E+02,
4.46E+02, 5.81E+02, 3.05E+02, 7.33E+02, 3.41E+02, 3.28E+02, 7.42E+02, 2.76E+02,

8.60E+01, 9.02E+02, 2.23E+02, 5.59E+02, 4.11E+02, 2.45E+02, 8.91E+02, 1.13E+02,
1.89E+02, 7.16E+02, 9.00E+02, 6.90E+01, 7.07E+02, 4.21E+02, 3.00E+00, 3.52E+02,
2.61E+02, 9.23E+02, 5.87E+02, 4.15E+02,

N = 250

2.60E+02, 3.01E+02, 5.28E+02, 3.37E+02, 5.84E+02, 9.09E+02, 9.72E+02, 3.28E+02,
4.82E+02, 6.70E+01, 6.18E+02, 1.74E+02, 6.82E+02, 2.14E+02, 2.00E+00, 5.64E+02,
7.46E+02, 4.35E+02, 1.37E+02, 7.66E+02, 4.02E+02, 3.06E+02, 7.98E+02, 2.20E+02,
2.83E+02, 6.58E+02, 6.99E+02, 2.71E+02, 2.31E+02, 8.45E+02, 3.22E+02, 3.78E+02,
3.60E+01, 9.02E+02, 5.36E+02, 7.74E+02, 5.94E+02, 4.05E+02, 6.15E+02, 5.09E+02,
3.82E+02, 7.29E+02, 4.42E+02, 7.56E+02, 8.74E+02, 4.85E+02, 8.71E+02, 6.05E+02,
2.36E+02, 3.91E+02, 3.65E+02, 9.85E+02, 2.53E+02, 7.70E+01, 6.38E+02, 2.68E+02,
7.86E+02, 6.91E+02, 4.07E+02, 9.82E+02, 2.78E+02, 1.62E+02, 9.65E+02, 3.30E+01,
8.68E+02, 2.10E+01, 9.42E+02, 3.40E+01, 9.59E+02, 8.87E+02, 6.96E+02, 7.07E+02,
9.79E+02, 9.63E+02, 5.37E+02, 4.52E+02, 2.97E+02, 2.96E+02, 2.34E+02, 6.27E+02,
8.37E+02, 8.49E+02, 8.23E+02, 9.88E+02, 2.42E+02, 4.74E+02, 5.82E+02, 3.39E+02,
4.34E+02, 6.47E+02, 2.48E+02, 6.81E+02, 5.08E+02, 6.78E+02, 9.87E+02, 1.88E+02,
3.30E+02, 7.38E+02, 3.90E+01, 8.05E+02, 3.33E+02, 8.62E+02, 2.00E+02, 1.11E+02,
2.60E+01, 1.80E+02, 9.43E+02, 3.84E+02, 2.50E+02, 1.10E+01, 6.60E+01, 4.80E+02,
1.67E+02, 2.46E+02, 7.14E+02, 8.96E+02, 4.99E+02, 8.93E+02, 1.10E+02, 4.00E+00,
5.75E+02, 2.72E+02, 4.97E+02, 6.51E+02, 8.91E+02, 9.89E+02, 3.72E+02, 7.67E+02,
6.66E+02, 6.30E+01, 7.13E+02, 6.87E+02, 2.17E+02, 6.71E+02, 1.93E+02, 1.12E+02,
1.29E+02, 1.95E+02, 1.75E+02, 1.34E+02, 4.49E+02, 1.73E+02, 4.26E+02, 2.56E+02,
5.20E+02, 7.61E+02, 6.02E+02, 8.51E+02, 2.93E+02, 6.65E+02, 3.63E+02, 8.22E+02,
9.50E+01, 2.03E+02, 8.80E+01, 5.25E+02, 3.58E+02, 5.13E+02, 5.42E+02, 7.50E+01,
1.35E+02, 5.63E+02, 8.26E+02, 3.81E+02, 4.28E+02, 3.97E+02, 2.86E+02, 1.65E+02,
5.10E+02, 7.72E+02, 8.39E+02, 4.50E+02, 5.20E+01, 4.50E+01, 8.75E+02, 3.26E+02,
8.57E+02, 2.09E+02, 4.89E+02, 4.79E+02, 2.26E+02, 2.51E+02, 4.46E+02, 9.56E+02,
2.07E+02, 2.57E+02, 7.70E+02, 6.00E+00, 9.16E+02, 4.76E+02, 5.96E+02, 4.61E+02,
5.67E+02, 4.22E+02, 3.32E+02, 3.43E+02, 9.61E+02, 4.03E+02, 6.12E+02, 4.27E+02,
4.92E+02, 6.31E+02, 9.03E+02, 9.08E+02, 5.48E+02, 5.73E+02, 5.14E+02, 3.62E+02,
7.84E+02, 6.25E+02, 7.64E+02, 5.61E+02, 6.43E+02, 1.40E+01, 2.69E+02, 5.93E+02,
5.60E+02, 6.62E+02, 1.54E+02, 4.24E+02, 2.28E+02, 6.41E+02, 5.46E+02, 5.01E+02,
1.90E+02, 4.39E+02, 3.93E+02, 2.33E+02, 2.59E+02, 1.30E+01, 9.05E+02, 9.07E+02,
3.80E+02, 3.00E+02, 7.11E+02, 1.57E+02, 2.91E+02, 8.70E+01, 5.74E+02, 9.81E+02,
4.80E+01, 8.10E+02, 1.33E+02, 4.23E+02, 2.08E+02, 9.41E+02, 9.12E+02, 4.88E+02,
1.09E+02, 9.60E+02,

N = 500

6.72E+02, 6.20E+02, 3.44E+02, 9.17E+02, 3.67E+02, 2.50E+02, 9.35E+02, 2.34E+02,
6.16E+02, 9.24E+02, 1.15E+02, 9.90E+01, 3.70E+01, 2.36E+02, 8.64E+02, 1.07E+02,
1.91E+02, 4.22E+02, 5.58E+02, 2.40E+01, 7.44E+02, 2.04E+02, 3.26E+02, 1.78E+02,
5.60E+02, 7.21E+02, 3.49E+02, 5.06E+02, 6.35E+02, 7.76E+02, 9.28E+02, 7.63E+02,
9.06E+02, 9.39E+02, 8.85E+02, 2.72E+02, 8.70E+02, 7.38E+02, 9.75E+02, 6.50E+02,
4.67E+02, 4.31E+02, 5.33E+02, 3.95E+02, 3.10E+02, 9.70E+02, 2.20E+01, 3.65E+02,
5.00E+01, 2.24E+02, 7.56E+02, 1.47E+02, 7.16E+02, 1.25E+02, 5.66E+02, 1.70E+01,
1.17E+02, 8.87E+02, 6.76E+02, 7.26E+02, 7.10E+02, 3.24E+02, 6.22E+02, 5.16E+02,
2.67E+02, 8.42E+02, 1.67E+02, 7.52E+02, 1.75E+02, 5.30E+01, 7.90E+02, 3.56E+02,
7.81E+02, 7.62E+02, 8.88E+02, 6.87E+02, 7.87E+02, 4.76E+02, 3.77E+02, 1.20E+02,
5.50E+01, 9.50E+01, 8.37E+02, 1.09E+02, 3.28E+02, 9.77E+02, 7.73E+02, 9.71E+02,
2.78E+02, 7.36E+02, 2.12E+02, 5.76E+02, 7.65E+02, 1.26E+02, 5.07E+02, 6.26E+02,
7.03E+02, 7.70E+01, 7.80E+01, 3.41E+02, 3.37E+02, 5.84E+02, 9.21E+02, 6.74E+02,
8.00E+01, 5.32E+02, 7.78E+02, 4.01E+02, 1.30E+01, 9.69E+02, 1.69E+02, 7.85E+02,
8.97E+02, 1.61E+02, 7.48E+02, 6.81E+02, 6.80E+01, 4.70E+01, 4.84E+02, 5.42E+02,
3.97E+02, 3.42E+02, 8.79E+02, 6.70E+01, 6.55E+02, 7.69E+02, 8.26E+02, 8.60E+01,
4.20E+02, 9.25E+02, 4.12E+02, 3.35E+02, 5.49E+02, 1.77E+02, 6.95E+02, 3.62E+02,
1.03E+02, 4.25E+02, 3.80E+02, 6.60E+01, 1.14E+02, 5.70E+01, 7.31E+02, 5.46E+02,
6.57E+02, 3.14E+02, 4.39E+02, 2.57E+02, 1.53E+02, 9.07E+02, 8.89E+02, 9.12E+02,
3.92E+02, 3.15E+02, 2.49E+02, 2.06E+02, 5.86E+02, 8.20E+02, 9.62E+02, 1.00E+02,
8.40E+01, 9.14E+02, 1.34E+02, 4.52E+02, 1.68E+02, 4.40E+02, 1.50E+02, 9.48E+02,
3.72E+02, 4.58E+02, 6.13E+02, 1.06E+02, 2.74E+02, 5.14E+02, 4.74E+02, 2.30E+01,
6.10E+01, 6.84E+02, 7.77E+02, 8.23E+02, 7.54E+02, 2.37E+02, 7.32E+02, 9.37E+02,
1.12E+02, 2.25E+02, 1.46E+02, 2.16E+02, 5.39E+02, 3.53E+02, 6.85E+02, 1.90E+01,
8.75E+02, 9.26E+02, 5.79E+02, 4.19E+02, 5.88E+02, 1.92E+02, 6.71E+02, 2.65E+02,
2.93E+02, 4.80E+01, 5.57E+02, 9.93E+02, 8.00E+02, 2.05E+02, 7.00E+01, 3.09E+02,
8.13E+02, 2.83E+02, 4.93E+02, 5.44E+02, 4.36E+02, 3.82E+02, 7.06E+02, 7.10E+01,
6.70E+02, 7.94E+02, 5.81E+02, 4.14E+02, 3.89E+02, 6.36E+02, 4.80E+02, 9.38E+02,
3.52E+02, 8.77E+02, 4.71E+02, 9.90E+02, 7.53E+02, 8.59E+02, 1.51E+02, 7.59E+02,

7.90E+01, 6.21E+02, 7.60E+02, 3.79E+02, 6.77E+02, 3.96E+02, 5.29E+02, 6.45E+02,
9.13E+02, 1.87E+02, 5.78E+02, 6.90E+01, 7.92E+02, 6.75E+02, 8.15E+02, 9.43E+02,
8.09E+02, 1.29E+02, 6.18E+02, 6.04E+02, 4.38E+02, 1.37E+02, 2.80E+02, 8.33E+02,
3.27E+02, 9.95E+02, 7.08E+02, 9.30E+01, 4.85E+02, 6.09E+02, 7.70E+02, 5.95E+02,
7.14E+02, 4.63E+02, 2.11E+02, 8.18E+02, 3.98E+02, 4.99E+02, 1.48E+02, 9.10E+01,
2.61E+02, 9.36E+02, 4.44E+02, 5.10E+01, 8.53E+02, 8.66E+02, 5.67E+02, 6.40E+02,
1.89E+02, 3.07E+02, 6.23E+02, 4.41E+02, 9.16E+02, 9.64E+02, 8.51E+02, 3.87E+02,
7.89E+02, 2.68E+02, 4.34E+02, 8.55E+02, 8.49E+02, 4.17E+02, 5.08E+02, 5.56E+02,
2.40E+02, 9.15E+02, 8.68E+02, 4.07E+02, 9.79E+02, 4.37E+02, 3.70E+02, 1.36E+02,
3.94E+02, 6.07E+02, 4.09E+02, 5.72E+02, 7.47E+02, 5.63E+02, 8.30E+02, 7.35E+02,
9.44E+02, 2.18E+02, 3.00E+01, 8.02E+02, 6.58E+02, 7.20E+02, 4.18E+02, 6.73E+02,
6.24E+02, 9.67E+02, 1.98E+02, 2.55E+02, 4.90E+02, 6.42E+02, 8.38E+02, 4.47E+02,
9.33E+02, 2.60E+02, 8.90E+02, 2.26E+02, 9.86E+02, 5.77E+02, 2.15E+02, 9.65E+02,
7.15E+02, 8.62E+02, 1.59E+02, 4.46E+02, 8.78E+02, 6.25E+02, 2.63E+02, 7.01E+02,
2.23E+02, 7.67E+02, 9.99E+02, 6.82E+02, 9.55E+02, 3.00E+02, 6.64E+02, 2.27E+02,
4.45E+02, 7.17E+02, 5.74E+02, 1.13E+02, 7.20E+01, 3.88E+02, 5.51E+02, 5.54E+02,
1.23E+02, 8.50E+01, 1.66E+02, 4.65E+02, 7.30E+02, 9.46E+02, 2.38E+02, 9.83E+02,
5.89E+02, 3.69E+02, 7.86E+02, 4.04E+02, 8.95E+02, 2.00E+02, 3.84E+02, 9.11E+02,
4.32E+02, 7.39E+02, 5.97E+02, 3.34E+02, 2.10E+02, 1.54E+02, 4.03E+02, 3.81E+02,
6.37E+02, 1.45E+02, 2.14E+02, 1.56E+02, 2.19E+02, 9.85E+02, 2.39E+02, 8.35E+02,
8.31E+02, 5.00E+00, 5.11E+02, 8.50E+02, 3.54E+02, 1.86E+02, 2.90E+01, 3.21E+02,
9.74E+02, 3.32E+02, 2.98E+02, 1.64E+02, 3.66E+02, 5.05E+02, 7.19E+02, 5.43E+02,
7.11E+02, 1.60E+01, 1.83E+02, 2.69E+02, 9.76E+02, 7.50E+01, 5.15E+02, 1.35E+02,
8.21E+02, 3.68E+02, 2.64E+02, 5.25E+02, 4.69E+02, 6.88E+02, 3.55E+02, 7.58E+02,
5.96E+02, 4.60E+01, 4.08E+02, 1.27E+02, 1.10E+02, 6.19E+02, 2.62E+02, 1.30E+02,
2.22E+02, 6.65E+02, 7.25E+02, 6.94E+02, 9.82E+02, 8.36E+02, 7.29E+02, 4.21E+02,
2.86E+02, 9.03E+02, 5.91E+02, 8.44E+02, 3.00E+00, 5.10E+02, 8.39E+02, 7.00E+02,
7.60E+01, 9.98E+02, 1.76E+02, 1.55E+02, 4.00E+00, 5.47E+02, 1.63E+02, 5.19E+02,
3.46E+02, 1.50E+01, 8.46E+02, 4.35E+02, 9.52E+02, 2.33E+02, 2.60E+01, 4.50E+01,
8.67E+02, 3.19E+02, 6.91E+02, 5.48E+02, 9.08E+02, 5.02E+02, 7.57E+02, 9.20E+01,
8.56E+02, 3.40E+01, 4.77E+02, 6.00E+00, 3.99E+02, 6.39E+02, 6.00E+02, 8.20E+01,
7.64E+02, 1.00E+00, 5.62E+02, 8.45E+02, 6.43E+02, 6.96E+02, 1.39E+02, 8.10E+01,
9.32E+02, 6.41E+02, 8.01E+02, 8.30E+01, 3.01E+02, 8.48E+02, 9.00E+00, 4.40E+01,
1.05E+02, 4.30E+02, 3.45E+02, 5.65E+02,

N = 750

4.60E+01, 5.22E+02, 4.31E+02, 6.58E+02, 6.02E+02, 6.43E+02, 7.00E+01, 2.71E+02,
3.24E+02, 1.74E+02, 7.80E+01, 2.08E+02, 1.18E+02, 5.12E+02, 5.02E+02, 7.20E+02,
9.22E+02, 7.62E+02, 1.62E+02, 3.77E+02, 8.11E+02, 4.73E+02, 9.37E+02, 6.44E+02,
5.92E+02, 7.73E+02, 2.00E+02, 3.07E+02, 4.59E+02, 5.67E+02, 8.51E+02, 5.78E+02,
4.62E+02, 7.30E+01, 7.68E+02, 4.44E+02, 7.20E+01, 3.15E+02, 1.21E+02, 2.22E+02,
9.21E+02, 3.60E+01, 8.40E+01, 2.33E+02, 5.74E+02, 8.98E+02, 5.40E+02, 5.69E+02,
9.71E+02, 8.30E+01, 4.85E+02, 2.14E+02, 6.34E+02, 5.72E+02, 4.12E+02, 4.58E+02,
9.95E+02, 9.04E+02, 7.46E+02, 4.48E+02, 5.04E+02, 9.19E+02, 3.49E+02, 9.79E+02,
9.76E+02, 6.42E+02, 7.08E+02, 7.36E+02, 5.50E+01, 2.77E+02, 1.82E+02, 6.20E+02,
7.50E+01, 9.40E+01, 8.46E+02, 2.32E+02, 8.17E+02, 2.45E+02, 2.81E+02, 1.33E+02,
4.37E+02, 2.91E+02, 1.10E+01, 7.09E+02, 9.88E+02, 1.80E+01, 1.85E+02, 9.82E+02,
5.56E+02, 8.70E+01, 7.39E+02, 7.10E+02, 3.51E+02, 2.58E+02, 1.71E+02, 9.91E+02,
4.70E+02, 9.75E+02, 7.60E+01, 9.59E+02, 6.51E+02, 3.44E+02, 8.03E+02, 8.33E+02,
5.98E+02, 3.62E+02, 1.91E+02, 2.60E+01, 1.63E+02, 2.76E+02, 1.00E+00, 7.52E+02,
6.50E+02, 7.78E+02, 1.12E+02, 6.32E+02, 1.45E+02, 9.47E+02, 4.79E+02, 9.93E+02,
5.09E+02, 6.24E+02, 8.78E+02, 8.55E+02, 5.81E+02, 5.60E+02, 7.11E+02, 7.90E+02,
8.86E+02, 8.02E+02, 4.90E+01, 9.84E+02, 9.92E+02, 2.61E+02, 3.29E+02, 4.61E+02,
6.09E+02, 2.52E+02, 8.85E+02, 1.17E+02, 3.26E+02, 6.67E+02, 9.89E+02, 2.23E+02,
2.64E+02, 9.43E+02, 3.00E+02, 7.96E+02, 7.17E+02, 9.28E+02, 1.20E+01, 7.97E+02,
4.21E+02, 2.38E+02, 7.15E+02, 4.84E+02, 1.11E+02, 4.39E+02, 8.24E+02, 2.78E+02,
5.19E+02, 2.68E+02, 8.89E+02, 7.75E+02, 6.90E+02, 9.85E+02, 2.49E+02, 1.39E+02,
3.80E+02, 2.86E+02, 9.87E+02, 5.86E+02, 5.03E+02, 3.01E+02, 4.20E+01, 3.34E+02,
6.77E+02, 3.67E+02, 7.38E+02, 4.71E+02, 6.62E+02, 4.24E+02, 2.59E+02, 7.31E+02,
4.49E+02, 7.77E+02, 7.70E+01, 6.88E+02, 5.48E+02, 4.35E+02, 6.76E+02, 3.92E+02,
3.22E+02, 3.28E+02, 2.73E+02, 2.75E+02, 6.78E+02, 9.51E+02, 9.08E+02, 4.20E+02,
4.90E+02, 3.61E+02, 8.34E+02, 5.57E+02, 8.70E+02, 4.66E+02, 9.23E+02, 3.38E+02,
3.20E+01, 1.07E+02, 2.74E+02, 5.11E+02, 2.98E+02, 3.50E+01, 3.94E+02, 5.65E+02,
2.95E+02, 9.06E+02, 9.60E+02, 8.84E+02, 6.79E+02, 6.87E+02, 2.36E+02, 6.94E+02,
5.50E+02, 1.47E+02, 6.95E+02, 8.90E+02, 6.46E+02, 4.03E+02, 6.80E+01, 9.68E+02,
1.31E+02, 3.17E+02, 7.05E+02, 3.73E+02, 1.26E+02, 7.86E+02, 7.30E+02, 2.56E+02,

2.30E+01, 5.39E+02, 1.95E+02, 5.87E+02, 3.52E+02, 5.15E+02, 7.07E+02, 9.61E+02,
8.19E+02, 3.93E+02, 3.88E+02, 2.24E+02, 8.10E+02, 8.39E+02, 9.58E+02, 1.02E+02,
2.80E+02, 7.88E+02, 2.10E+02, 4.54E+02, 6.01E+02, 3.25E+02, 1.28E+02, 1.13E+02,
3.39E+02, 5.91E+02, 9.50E+02, 6.59E+02, 1.00E+01, 3.35E+02, 1.87E+02, 7.00E+00,
1.88E+02, 7.48E+02, 9.73E+02, 7.24E+02, 9.86E+02, 5.71E+02, 4.53E+02, 2.20E+02,
2.62E+02, 5.93E+02, 8.37E+02, 1.10E+02, 5.47E+02, 2.53E+02, 8.76E+02, 4.01E+02,
4.13E+02, 2.63E+02, 8.27E+02, 9.60E+01, 3.83E+02, 9.81E+02, 6.33E+02, 1.32E+02,
4.55E+02, 7.50E+02, 4.80E+02, 8.45E+02, 5.29E+02, 3.79E+02, 9.30E+01, 7.27E+02,
7.33E+02, 5.42E+02, 2.96E+02, 7.22E+02, 3.19E+02, 8.30E+02, 8.81E+02, 1.46E+02,
3.70E+02, 8.61E+02, 6.08E+02, 3.81E+02, 2.06E+02, 2.40E+02, 1.20E+02, 9.66E+02,
8.21E+02, 4.40E+02, 2.27E+02, 3.23E+02, 8.38E+02, 1.98E+02, 4.05E+02, 2.65E+02,
9.78E+02, 6.89E+02, 4.36E+02, 9.55E+02, 1.83E+02, 8.13E+02, 9.69E+02, 5.46E+02,
8.43E+02, 4.80E+01, 4.72E+02, 2.51E+02, 6.99E+02, 7.92E+02, 5.30E+01, 6.13E+02,
1.16E+02, 7.32E+02, 9.00E+01, 9.42E+02, 4.78E+02, 3.76E+02, 1.56E+02, 8.50E+02,
5.79E+02, 2.93E+02, 5.34E+02, 3.64E+02, 3.90E+02, 3.58E+02, 6.17E+02, 9.54E+02,
5.40E+01, 8.96E+02, 9.49E+02, 5.97E+02, 9.24E+02, 7.69E+02, 9.03E+02, 4.00E+02,
8.00E+00, 5.70E+01, 3.21E+02, 3.66E+02, 7.67E+02, 1.75E+02, 8.40E+02, 9.25E+02,
7.19E+02, 5.60E+01, 9.00E+00, 4.98E+02, 1.29E+02, 8.57E+02, 6.30E+02, 2.94E+02,
6.93E+02, 6.29E+02, 4.63E+02, 1.30E+01, 3.50E+02, 6.56E+02, 7.01E+02, 6.10E+02,
2.10E+01, 5.89E+02, 6.69E+02, 7.37E+02, 5.35E+02, 8.04E+02, 4.14E+02, 1.70E+02,
6.47E+02, 5.31E+02, 9.80E+02, 8.50E+01, 8.52E+02, 6.85E+02, 5.28E+02, 9.27E+02,
9.33E+02, 4.74E+02, 6.15E+02, 1.59E+02, 3.00E+01, 1.77E+02, 4.86E+02, 2.48E+02,
7.12E+02, 9.29E+02, 8.99E+02, 7.25E+02, 2.44E+02, 7.13E+02, 8.01E+02, 3.97E+02,
2.50E+02, 2.46E+02, 7.16E+02, 8.56E+02, 7.53E+02, 5.99E+02, 6.86E+02, 9.20E+02,
1.06E+02, 2.85E+02, 9.44E+02, 1.72E+02, 6.06E+02, 2.12E+02, 4.26E+02, 4.22E+02,
7.54E+02, 2.37E+02, 6.73E+02, 2.60E+02, 6.36E+02, 1.81E+02, 4.28E+02, 3.00E+00,
9.36E+02, 9.38E+02, 6.12E+02, 3.96E+02, 8.91E+02, 8.67E+02, 8.77E+02, 2.15E+02,
6.21E+02, 1.64E+02, 3.87E+02, 4.67E+02, 8.20E+02, 8.74E+02, 6.04E+02, 3.71E+02,
6.70E+02, 2.18E+02, 1.08E+02, 8.82E+02, 8.10E+01, 3.12E+02, 3.69E+02, 2.00E+00,
1.99E+02, 6.74E+02, 8.68E+02, 5.23E+02, 5.95E+02, 1.73E+02, 1.69E+02, 6.90E+01,
9.02E+02, 6.71E+02, 8.00E+01, 1.90E+02, 5.18E+02, 2.42E+02, 8.80E+02, 4.25E+02,
2.19E+02, 7.21E+02, 2.13E+02, 3.33E+02, 1.54E+02, 2.92E+02, 6.55E+02, 4.30E+02,
6.23E+02, 4.76E+02, 7.42E+02, 3.04E+02, 4.50E+01, 7.83E+02, 7.04E+02, 2.26E+02,
4.06E+02, 8.95E+02, 2.00E+01, 1.30E+02, 1.90E+01, 3.30E+01, 4.47E+02, 3.82E+02,
8.58E+02, 4.75E+02, 7.74E+02, 7.84E+02, 2.80E+01, 5.38E+02, 3.09E+02, 3.98E+02,
8.26E+02, 5.55E+02, 1.41E+02, 3.91E+02, 8.69E+02, 1.79E+02, 7.60E+02, 1.40E+01,
6.50E+01, 4.51E+02, 1.01E+02, 7.80E+02, 3.41E+02, 9.17E+02, 8.71E+02, 3.95E+02,
7.72E+02, 2.97E+02, 4.52E+02, 5.17E+02, 1.52E+02, 1.76E+02, 7.49E+02, 6.41E+02,
5.41E+02, 7.82E+02, 4.42E+02, 9.67E+02, 1.53E+02, 4.77E+02, 7.91E+02, 4.10E+01,
1.51E+02, 2.47E+02, 8.54E+02, 7.61E+02, 2.34E+02, 1.70E+01, 9.99E+02, 6.25E+02,
4.38E+02, 7.59E+02, 3.40E+01, 1.92E+02, 5.10E+02, 3.32E+02, 5.54E+02, 1.97E+02,
5.21E+02, 2.90E+01, 5.70E+02, 2.17E+02, 7.64E+02, 7.00E+02, 5.30E+02, 9.90E+01,
5.84E+02, 5.07E+02, 1.43E+02, 4.70E+01, 4.82E+02, 7.99E+02, 4.02E+02, 8.23E+02,
7.66E+02, 5.33E+02, 6.20E+01, 6.10E+01, 5.96E+02, 8.18E+02, 8.49E+02, 5.80E+02,
7.23E+02, 1.93E+02, 3.68E+02, 8.48E+02, 6.31E+02, 9.74E+02, 8.53E+02, 6.49E+02,
9.18E+02, 6.26E+02, 7.29E+02, 4.68E+02, 1.37E+02, 6.54E+02, 9.41E+02, 1.48E+02,
6.37E+02, 9.26E+02, 5.06E+02, 5.26E+02, 4.00E+00, 4.29E+02, 5.49E+02, 2.82E+02,
3.74E+02, 4.89E+02, 1.23E+02, 6.22E+02, 1.34E+02, 8.97E+02, 6.70E+01, 4.81E+02,
4.18E+02, 1.38E+02, 3.54E+02, 9.39E+02, 5.24E+02, 0.00E+00, 8.60E+01, 5.61E+02,
7.57E+02, 1.15E+02, 8.87E+02, 4.04E+02, 3.48E+02, 6.97E+02, 9.20E+01, 4.32E+02,
2.66E+02, 7.94E+02, 8.64E+02, 5.10E+01, 9.32E+02, 5.75E+02, 6.63E+02, 2.79E+02,
2.84E+02, 6.07E+02, 8.05E+02, 5.08E+02, 6.45E+02, 9.40E+02, 9.65E+02, 5.20E+02,
6.38E+02, 5.43E+02, 3.90E+01, 7.98E+02, 1.58E+02, 1.50E+01, 3.06E+02, 2.25E+02,
3.05E+02, 3.03E+02, 9.15E+02, 4.97E+02, 3.53E+02, 4.07E+02, 1.44E+02, 7.93E+02,
4.34E+02, 8.92E+02, 7.35E+02, 7.81E+02, 3.11E+02, 1.19E+02, 4.69E+02, 2.29E+02,
8.60E+02, 3.75E+02, 7.56E+02, 4.43E+02, 2.03E+02, 2.57E+02, 7.85E+02, 1.57E+02,
3.85E+02, 8.15E+02, 5.77E+02, 6.48E+02, 7.40E+02, 4.11E+02, 2.83E+02, 6.57E+02,
5.64E+02, 4.99E+02, 6.75E+02, 6.14E+02, 3.43E+02, 3.42E+02, 9.57E+02, 6.82E+02,
7.28E+02, 8.29E+02, 2.41E+02, 1.25E+02, 7.44E+02, 9.30E+02, 2.55E+02, 6.40E+01,
1.09E+02, 2.05E+02, 6.11E+02, 5.00E+00, 2.89E+02, 4.17E+02, 6.68E+02, 4.46E+02,
2.16E+02, 1.05E+02, 1.60E+02, 4.09E+02, 8.65E+02, 2.09E+02, 2.70E+01, 5.76E+02,
9.70E+01, 6.16E+02, 3.10E+02, 3.56E+02, 7.26E+02, 3.55E+02, 7.58E+02, 2.67E+02,
9.70E+02, 3.40E+02, 9.52E+02, 9.11E+02, 8.80E+01, 6.92E+02, 4.08E+02, 8.36E+02,
7.95E+02, 5.68E+02, 9.16E+02, 6.03E+02, 4.65E+02, 6.19E+02,
N = 1000
8.34E+02, 1.60E+02, 5.00E+01, 3.32E+02, 6.21E+02, 3.00E+02, 3.45E+02, 3.68E+02,
5.07E+02, 8.40E+02, 9.47E+02, 2.89E+02, 4.54E+02, 1.20E+01, 5.29E+02, 4.50E+02,

4.36E+02, 5.90E+02, 8.29E+02, 3.58E+02, 5.94E+02, 6.67E+02, 9.83E+02, 1.81E+02,
2.50E+01, 5.89E+02, 7.10E+02, 2.24E+02, 3.47E+02, 8.89E+02, 6.15E+02, 4.40E+01,
5.59E+02, 7.40E+01, 7.35E+02, 4.00E+02, 5.52E+02, 5.80E+01, 7.86E+02, 5.46E+02,
1.10E+02, 6.62E+02, 5.44E+02, 8.43E+02, 2.71E+02, 3.62E+02, 2.34E+02, 1.37E+02,
3.93E+02, 6.88E+02, 5.01E+02, 6.87E+02, 1.11E+02, 9.33E+02, 1.90E+01, 7.20E+02,
4.29E+02, 5.21E+02, 5.15E+02, 3.54E+02, 5.20E+02, 2.98E+02, 2.07E+02, 7.48E+02,
4.10E+02, 7.45E+02, 5.79E+02, 2.59E+02, 7.17E+02, 4.70E+01, 1.26E+02, 9.90E+02,
5.60E+01, 5.40E+01, 6.66E+02, 9.86E+02, 9.05E+02, 8.22E+02, 5.58E+02, 4.76E+02,
7.31E+02, 6.80E+01, 1.20E+02, 3.40E+01, 4.74E+02, 3.46E+02, 7.02E+02, 3.82E+02,
4.65E+02, 8.00E+02, 1.82E+02, 6.44E+02, 6.46E+02, 2.48E+02, 7.80E+02, 8.51E+02,
4.42E+02, 4.50E+01, 2.40E+01, 7.05E+02, 7.25E+02, 8.45E+02, 3.85E+02, 8.25E+02,
6.02E+02, 8.53E+02, 8.00E+01, 8.37E+02, 6.07E+02, 9.81E+02, 9.87E+02, 3.02E+02,
2.96E+02, 2.57E+02, 4.47E+02, 9.68E+02, 7.18E+02, 7.12E+02, 2.06E+02, 8.32E+02,
2.09E+02, 6.52E+02, 1.12E+02, 1.86E+02, 7.57E+02, 9.31E+02, 2.58E+02, 9.51E+02,
4.95E+02, 4.43E+02, 7.73E+02, 9.76E+02, 1.39E+02, 5.22E+02, 9.52E+02, 1.40E+02,
7.37E+02, 8.80E+02, 9.71E+02, 9.03E+02, 6.17E+02, 7.84E+02, 2.92E+02, 4.62E+02,
6.86E+02, 7.47E+02, 3.17E+02, 6.39E+02, 2.77E+02, 5.45E+02, 9.38E+02, 8.57E+02,
4.90E+01, 7.04E+02, 5.30E+01, 5.67E+02, 5.41E+02, 9.90E+01, 5.12E+02, 2.43E+02,
8.50E+01, 1.92E+02, 7.83E+02, 1.87E+02, 1.62E+02, 7.51E+02, 9.93E+02, 3.44E+02,
9.70E+01, 2.12E+02, 7.64E+02, 7.42E+02, 4.35E+02, 7.03E+02, 7.23E+02, 3.04E+02,
2.95E+02, 3.43E+02, 1.38E+02, 2.05E+02, 9.26E+02, 8.07E+02, 2.23E+02, 6.09E+02,
3.65E+02, 6.79E+02, 8.17E+02, 1.45E+02, 5.23E+02, 7.32E+02, 5.76E+02, 8.26E+02,
1.29E+02, 6.11E+02, 3.52E+02, 4.66E+02, 9.80E+01, 1.57E+02, 9.25E+02, 1.72E+02,
9.11E+02, 8.08E+02, 4.86E+02, 6.12E+02, 5.49E+02, 6.60E+02, 5.31E+02, 7.68E+02,
3.64E+02, 5.24E+02, 2.85E+02, 2.94E+02, 5.73E+02, 6.22E+02, 1.10E+01, 3.09E+02,
9.62E+02, 9.72E+02, 5.99E+02, 4.03E+02, 9.36E+02, 1.09E+02, 1.70E+02, 4.40E+02,
1.89E+02, 4.49E+02, 5.36E+02, 7.92E+02, 2.90E+01, 3.31E+02, 9.20E+02, 6.10E+01,
6.96E+02, 9.09E+02, 5.91E+02, 1.80E+02, 4.72E+02, 9.24E+02, 1.88E+02, 3.53E+02,
2.00E+02, 9.60E+02, 9.15E+02, 7.07E+02, 3.90E+01, 3.60E+02, 8.20E+02, 5.08E+02,
2.42E+02, 1.55E+02, 8.09E+02, 2.91E+02, 7.98E+02, 2.29E+02, 7.43E+02, 6.41E+02,
2.21E+02, 6.98E+02, 8.82E+02, 4.91E+02, 2.52E+02, 6.10E+02, 4.82E+02, 3.11E+02,
6.74E+02, 8.28E+02, 5.98E+02, 1.67E+02, 7.90E+01, 8.11E+02, 2.50E+02, 8.81E+02,
6.00E+02, 6.20E+02, 8.38E+02, 6.43E+02, 6.77E+02, 3.00E+01, 7.52E+02, 2.02E+02,
2.76E+02, 8.61E+02, 1.77E+02, 5.06E+02, 2.39E+02, 2.88E+02, 1.54E+02, 5.18E+02,
7.14E+02, 8.50E+02, 3.07E+02, 6.56E+02, 8.00E+00, 1.50E+02, 5.35E+02, 9.79E+02,
6.19E+02, 9.65E+02, 2.20E+01, 5.30E+02, 3.30E+01, 9.46E+02, 6.58E+02, 5.16E+02,
8.74E+02, 7.33E+02, 7.66E+02, 2.30E+02, 1.83E+02, 2.82E+02, 4.13E+02, 8.56E+02,
5.50E+01, 5.86E+02, 9.95E+02, 1.35E+02, 8.60E+02, 9.63E+02, 8.63E+02, 7.72E+02,
9.48E+02, 5.03E+02, 2.25E+02, 5.72E+02, 3.87E+02, 4.00E+00, 1.85E+02, 1.70E+01,
8.72E+02, 6.26E+02, 8.78E+02, 2.04E+02, 8.77E+02, 2.74E+02, 7.15E+02, 3.84E+02,
8.86E+02, 6.30E+01, 8.21E+02, 7.75E+02, 7.70E+01, 2.67E+02, 8.83E+02, 2.20E+02,
2.27E+02, 4.28E+02, 3.76E+02, 9.04E+02, 5.00E+00, 1.65E+02, 8.76E+02, 4.67E+02,
6.35E+02, 7.36E+02, 4.78E+02, 1.24E+02, 9.99E+02, 7.95E+02, 4.23E+02, 8.55E+02,
7.58E+02, 3.69E+02, 4.73E+02, 2.46E+02, 9.98E+02, 8.16E+02, 6.69E+02, 2.41E+02,
9.85E+02, 4.48E+02, 2.00E+01, 2.53E+02, 4.79E+02, 7.97E+02, 3.33E+02, 1.16E+02,
8.79E+02, 4.80E+01, 8.95E+02, 5.66E+02, 3.66E+02, 8.88E+02, 8.98E+02, 3.37E+02,
3.56E+02, 3.91E+02, 3.21E+02, 8.54E+02, 8.47E+02, 1.42E+02, 2.60E+02, 5.90E+01,
1.94E+02, 5.25E+02, 1.30E+02, 3.41E+02, 7.85E+02, 1.52E+02, 8.64E+02, 5.19E+02,
3.96E+02, 1.02E+02, 8.94E+02, 6.63E+02, 4.04E+02, 8.65E+02, 4.61E+02, 3.72E+02,
2.54E+02, 7.71E+02, 7.55E+02, 4.63E+02, 2.49E+02, 9.96E+02, 6.20E+01, 2.65E+02,
7.80E+01, 1.28E+02, 4.16E+02, 8.31E+02, 6.31E+02, 3.89E+02, 6.76E+02, 3.20E+01,
9.23E+02, 6.50E+01, 5.05E+02, 2.62E+02, 9.40E+01, 4.55E+02, 3.40E+02, 5.77E+02,
9.32E+02, 8.10E+02, 6.75E+02, 1.95E+02, 2.70E+02, 5.02E+02, 5.54E+02, 1.03E+02,
3.13E+02, 6.38E+02, 6.24E+02, 3.29E+02, 1.90E+02, 1.73E+02, 6.83E+02, 4.94E+02,
6.73E+02, 1.00E+00, 3.78E+02, 8.70E+01, 4.70E+02, 8.87E+02, 7.41E+02, 5.63E+02,
6.81E+02, 5.04E+02, 9.69E+02, 8.60E+01, 2.64E+02, 5.60E+02, 4.60E+01, 2.26E+02,
6.60E+01, 9.28E+02, 7.21E+02, 8.06E+02, 8.14E+02, 4.33E+02, 8.49E+02, 3.50E+01,
9.54E+02, 1.32E+02, 5.42E+02, 7.67E+02, 4.22E+02, 9.20E+01, 4.64E+02, 5.11E+02,
8.93E+02, 5.26E+02, 8.73E+02, 6.89E+02, 5.61E+02, 1.04E+02, 6.28E+02, 9.39E+02,
9.75E+02, 3.83E+02, 2.68E+02, 4.00E+01, 4.58E+02, 6.53E+02, 9.82E+02, 9.91E+02,
4.71E+02, 9.06E+02, 9.10E+01, 9.41E+02, 2.66E+02, 2.79E+02, 9.57E+02, 3.34E+02,
4.57E+02, 1.47E+02, 7.44E+02, 1.51E+02, 9.18E+02, 1.22E+02, 1.80E+01, 1.60E+01,
4.53E+02, 3.94E+02, 3.48E+02, 7.53E+02, 3.24E+02, 1.43E+02, 3.71E+02, 5.71E+02,
1.78E+02, 4.56E+02, 7.06E+02, 8.70E+02, 7.26E+02, 9.53E+02, 5.84E+02, 6.23E+02,
6.32E+02, 7.99E+02, 8.58E+02, 6.72E+02, 7.08E+02, 6.99E+02, 4.52E+02, 4.34E+02,
2.31E+02, 3.73E+02, 6.40E+01, 7.65E+02, 8.92E+02, 4.93E+02, 1.08E+02, 3.81E+02,
9.70E+02, 7.16E+02, 9.00E+00, 3.08E+02, 5.37E+02, 1.41E+02, 6.91E+02, 3.42E+02,

4.05E+02, 9.34E+02, 6.06E+02, 7.62E+02, 6.59E+02, 7.79E+02, 9.17E+02, 2.80E+02,
6.57E+02, 1.93E+02, 5.50E+02, 8.41E+02, 3.90E+02, 2.13E+02, 7.00E+01, 3.14E+02,
4.69E+02, 8.03E+02, 7.11E+02, 6.14E+02, 6.90E+01, 8.67E+02, 9.30E+01, 9.02E+02,
9.44E+02, 6.78E+02, 6.45E+02, 6.85E+02, 5.10E+02, 7.28E+02, 8.27E+02, 8.15E+02,
6.54E+02, 8.48E+02, 1.40E+01, 7.49E+02, 9.21E+02, 1.27E+02, 4.02E+02, 9.89E+02,
1.99E+02, 8.84E+02, 7.96E+02, 1.61E+02, 9.45E+02, 8.66E+02, 4.41E+02, 4.30E+01,
1.00E+01, 9.92E+02, 1.56E+02, 1.58E+02, 9.59E+02, 3.15E+02, 6.93E+02, 5.69E+02,
4.45E+02, 4.60E+02, 3.25E+02, 8.44E+02, 6.33E+02, 3.67E+02, 2.73E+02, 8.71E+02,
9.74E+02, 3.80E+01, 1.15E+02, 3.38E+02, 7.90E+02, 9.30E+02, 9.94E+02, 1.19E+02,
4.08E+02, 5.40E+02, 8.13E+02, 5.09E+02, 1.68E+02, 9.66E+02, 6.70E+01, 9.35E+02,
6.00E+01, 4.27E+02, 3.92E+02, 2.36E+02, 3.79E+02, 4.96E+02, 9.67E+02, 6.27E+02,
2.69E+02, 2.17E+02, 9.16E+02, 4.15E+02, 5.74E+02, 4.97E+02, 5.28E+02, 3.39E+02,
5.70E+02, 1.06E+02, 2.93E+02, 2.81E+02, 9.97E+02, 4.84E+02, 9.08E+02, 9.00E+01,
4.20E+01, 7.20E+01, 5.82E+02, 1.33E+02, 2.47E+02, 3.49E+02, 4.17E+02, 3.95E+02,
4.10E+01, 1.74E+02, 6.70E+02, 7.22E+02, 8.99E+02, 9.80E+02, 3.01E+02, 4.90E+02,
1.31E+02, 2.08E+02, 1.76E+02, 2.60E+01, 6.71E+02, 9.58E+02, 4.68E+02, 2.80E+01,
1.79E+02, 2.83E+02, 7.46E+02, 1.44E+02, 7.34E+02, 2.28E+02, 8.01E+02, 8.40E+01,
5.38E+02, 1.30E+01, 5.93E+02, 3.27E+02, 6.61E+02, 8.19E+02, 3.70E+02, 1.46E+02,
8.90E+01, 2.10E+02, 9.84E+02, 9.19E+02, 4.39E+02, 6.47E+02, 8.59E+02, 7.88E+02,
8.10E+01, 4.31E+02, 2.22E+02, 6.13E+02, 1.59E+02, 5.81E+02, 8.91E+02, 2.10E+01,
7.27E+02, 3.61E+02, 4.89E+02, 7.81E+02, 3.06E+02, 7.00E+02, 2.56E+02, 7.60E+01,
6.29E+02, 5.97E+02, 4.80E+02, 6.18E+02, 4.83E+02, 7.54E+02, 9.00E+02, 6.37E+02,
4.75E+02, 5.55E+02, 8.46E+02, 1.13E+02, 3.00E+00, 4.32E+02, 5.39E+02, 2.75E+02,
4.77E+02, 7.74E+02, 9.40E+02, 5.57E+02, 3.10E+01, 6.30E+02, 5.13E+02, 1.53E+02,
4.07E+02, 3.74E+02, 1.00E+02, 4.81E+02, 4.18E+02, 9.50E+02, 6.95E+02, 2.99E+02,
7.40E+02, 6.65E+02, 1.36E+02, 8.18E+02, 8.80E+01, 6.68E+02, 3.97E+02, 2.87E+02,
3.70E+01, 6.03E+02, 5.62E+02, 8.02E+02, 1.96E+02, 6.34E+02, 7.89E+02, 2.97E+02,
2.44E+02, 8.36E+02, 6.64E+02, 8.04E+02, 1.84E+02, 3.75E+02, 1.69E+02, 4.51E+02,
3.28E+02, 1.14E+02, 2.15E+02, 8.35E+02, 7.91E+02, 4.24E+02, 4.44E+02, 3.60E+01,
5.64E+02, 2.45E+02, 7.29E+02, 4.87E+02, 8.96E+02, 0.00E+00, 1.97E+02, 6.50E+02,
5.14E+02, 1.63E+02, 4.38E+02, 4.59E+02, 7.70E+02, 5.43E+02, 3.35E+02, 7.56E+02,
8.30E+01, 2.86E+02, 5.34E+02, 9.12E+02, 1.18E+02, 5.17E+02, 2.30E+01, 2.00E+00,
8.20E+01, 7.38E+02, 2.78E+02, 9.56E+02, 5.20E+01, 4.92E+02, 6.92E+02, 1.25E+02,
6.48E+02, 7.19E+02, 4.85E+02, 6.08E+02, 3.26E+02, 2.01E+02, 4.19E+02, 5.48E+02,
2.61E+02, 5.33E+02, 9.49E+02, 5.27E+02, 4.26E+02, 5.78E+02, 6.04E+02, 3.80E+02,
8.75E+02, 9.27E+02, 1.48E+02, 5.75E+02, 3.57E+02, 5.51E+02, 9.73E+02, 5.92E+02,
4.12E+02, 9.43E+02, 4.09E+02, 1.21E+02, 2.14E+02, 3.10E+02, 2.51E+02, 9.88E+02,
2.16E+02, 6.05E+02, 5.96E+02, 3.22E+02, 7.94E+02, 9.37E+02, 4.37E+02, 7.09E+02,
2.03E+02, 1.71E+02, 5.80E+02, 6.25E+02, 1.49E+02, 1.17E+02, 4.98E+02, 9.01E+02,
3.20E+02, 4.11E+02, 7.30E+02, 6.51E+02, 7.93E+02, 9.10E+02, 7.69E+02, 2.40E+02,
9.78E+02, 9.77E+02, 2.84E+02, 9.07E+02, 9.60E+01, 4.30E+02, 3.18E+02, 5.10E+01,
1.01E+02, 2.19E+02, 9.61E+02, 7.87E+02, 5.88E+02, 8.68E+02, 7.30E+01, 5.00E+02,
7.82E+02, 5.87E+02, 1.75E+02, 7.50E+01, 8.42E+02, 2.72E+02, 2.33E+02, 3.88E+02,
8.97E+02, 4.06E+02, 3.30E+02, 8.24E+02, 2.11E+02, 2.63E+02, 5.32E+02, 6.01E+02,
2.55E+02, 3.77E+02, 1.34E+02, 8.12E+02, 5.95E+02, 8.85E+02, 9.55E+02, 6.94E+02,
3.19E+02, 1.23E+02, 6.82E+02, 4.21E+02, 2.38E+02, 4.88E+02, 3.86E+02, 8.05E+02,
1.50E+01, 3.99E+02, 7.13E+02, 4.99E+02, 8.23E+02, 3.36E+02, 8.33E+02, 7.77E+02,
8.39E+02, 6.36E+02, 3.51E+02, 1.98E+02, 4.25E+02, 2.18E+02, 5.70E+01, 1.05E+02,
9.64E+02, 3.12E+02, 5.68E+02, 6.55E+02, 5.65E+02, 2.32E+02, 7.50E+02, 9.14E+02,
6.80E+02, 9.22E+02, 7.00E+00, 6.90E+02, 2.70E+01, 3.23E+02, 3.98E+02, 8.90E+02,
5.83E+02, 8.62E+02, 5.56E+02, 5.85E+02, 2.37E+02, 7.10E+01, 9.42E+02, 3.63E+02,
7.01E+02, 8.69E+02, 3.03E+02, 6.16E+02, 6.49E+02, 3.05E+02, 7.60E+02, 4.20E+02,
2.90E+02, 6.84E+02, 9.50E+01, 7.76E+02, 1.07E+02, 7.24E+02, 9.29E+02, 7.78E+02,
6.40E+02, 5.53E+02, 7.39E+02, 3.55E+02, 1.66E+02, 6.42E+02, 8.52E+02, 5.47E+02,
7.59E+02, 2.35E+02, 3.16E+02, 4.46E+02, 1.64E+02, 3.59E+02, 7.61E+02, 8.30E+02,
7.63E+02, 1.91E+02, 3.50E+02, 6.97E+02, 4.14E+02, 9.13E+02, 6.00E+00, 4.01E+02,

ii. Duplicate

N = 10

499.00, 755.00, 564.00, 814.00, 299.00, 911.00, 299.00, 235.00, 263.00, 272.00,

N = 50

673, 608, 73, 211, 726, 83, 599, 143, 911, 4.00, 86.00, 933.00, 288.00, 927.00,
737.00, 872.00, 662.00, 592.00, 413.00, 840.00, 193.00, 389.00, 813.00, 102.00,
61.00, 233.00, 540.00, 463.00, 808.00, 6.00, 513.00, 245.00, 840.00, 951.00, 764.00,

462.00, 959.00, 868.00, 564.00, 11.00, 938.00, 626.00, 587.00, 687.00, 343.00, 339.00, 243.00, 897.00, 790.00, 213.00,

N = 100

3.25E+02, 9.92E+02, 3.17E+02, 4.22E+02, 1.37E+02, 7.42E+02, 3.34E+02, 7.72E+02, 6.84E+02, 6.37E+02, 1.30E+02, 4.89E+02, 6.27E+02, 4.77E+02, 8.93E+02, 5.04E+02, 9.56E+02, 4.52E+02, 3.23E+02, 8.31E+02, 6.63E+02, 8.62E+02, 7.50E+01, 2.75E+02, 4.92E+02, 5.71E+02, 3.37E+02, 4.91E+02, 3.85E+02, 1.42E+02, 6.92E+02, 4.34E+02, 2.95E+02, 6.32E+02, 4.90E+01, 8.40E+02, 5.51E+02, 4.99E+02, 8.01E+02, 4.12E+02, 4.86E+02, 8.35E+02, 9.40E+01, 5.72E+02, 4.00E+02, 9.41E+02, 7.66E+02, 9.05E+02, 2.93E+02, 3.70E+01, 1.70E+01, 4.58E+02, 8.09E+02, 1.93E+02, 6.93E+02, 3.20E+02, 6.13E+02, 5.33E+02, 7.95E+02, 4.64E+02, 7.64E+02, 1.53E+02, 7.44E+02, 3.45E+02, 7.63E+02, 9.64E+02, 5.72E+02, 6.94E+02, 9.29E+02, 1.00E+01, 7.56E+02, 2.77E+02, 6.58E+02, 7.71E+02, 8.24E+02, 9.51E+02, 5.85E+02, 3.42E+02, 3.74E+02, 8.69E+02, 5.60E+01, 9.14E+02, 7.91E+02, 7.30E+01, 1.66E+02, 5.10E+02, 5.19E+02, 9.69E+02, 6.22E+02, 4.63E+02, 7.14E+02, 7.12E+02, 2.47E+02, 2.19E+02, 5.47E+02, 3.88E+02, 1.57E+02, 3.54E+02, 5.57E+02, 4.09E+02,

N = 250

1.36E+02, 3.94E+02, 8.23E+02, 3.01E+02, 6.55E+02, 5.38E+02, 5.97E+02, 5.20E+02, 5.43E+02, 6.39E+02, 9.98E+02, 5.80E+01, 3.42E+02, 9.55E+02, 9.34E+02, 5.47E+02, 5.11E+02, 6.82E+02, 2.90E+01, 2.58E+02, 3.10E+01, 7.74E+02, 4.40E+02, 5.30E+02, 1.93E+02, 5.03E+02, 9.77E+02, 4.50E+01, 1.10E+02, 6.38E+02, 6.41E+02, 1.92E+02, 7.48E+02, 2.15E+02, 5.54E+02, 1.21E+02, 4.93E+02, 9.50E+01, 1.81E+02, 9.11E+02, 7.30E+01, 6.59E+02, 1.78E+02, 8.66E+02, 3.15E+02, 2.68E+02, 5.36E+02, 8.85E+02, 4.62E+02, 5.12E+02, 5.99E+02, 2.51E+02, 4.14E+02, 5.35E+02, 6.80E+01, 6.75E+02, 6.77E+02, 1.20E+02, 2.91E+02, 9.41E+02, 8.54E+02, 5.10E+02, 5.22E+02, 8.39E+02, 2.86E+02, 9.25E+02, 8.69E+02, 8.86E+02, 1.30E+01, 9.24E+02, 1.39E+02, 5.26E+02, 3.62E+02, 8.50E+01, 1.60E+02, 9.92E+02, 6.74E+02, 4.74E+02, 9.96E+02, 2.35E+02, 9.30E+02, 9.80E+02, 6.06E+02, 2.19E+02, 4.06E+02, 3.00E+02, 3.60E+01, 3.78E+02, 9.00E+02, 5.67E+02, 2.36E+02, 1.50E+01, 3.51E+02, 7.50E+01, 4.10E+02, 5.00E+00, 4.80E+01, 3.48E+02, 1.09E+02, 8.18E+02, 6.00E+02, 4.02E+02, 7.97E+02, 3.02E+02, 9.27E+02, 7.33E+02, 1.70E+02, 7.38E+02, 8.97E+02, 7.25E+02, 4.41E+02, 8.21E+02, 4.67E+02, 4.57E+02, 9.48E+02, 5.65E+02, 4.98E+02, 9.43E+02, 2.07E+02, 3.41E+02, 9.76E+02, 1.60E+01, 8.81E+02, 3.40E+02, 7.98E+02, 8.29E+02, 9.20E+02, 1.30E+02, 1.29E+02, 9.50E+02, 4.47E+02, 3.99E+02, 5.80E+02, 2.94E+02, 6.18E+02, 8.01E+02, 7.94E+02, 1.65E+02, 9.74E+02, 5.32E+02, 4.75E+02, 2.28E+02, 7.24E+02, 5.28E+02, 2.23E+02, 3.20E+01, 5.15E+02, 8.43E+02, 1.62E+02, 8.91E+02, 6.54E+02, 7.29E+02, 8.40E+02, 4.09E+02, 8.06E+02, 5.59E+02, 1.22E+02, 8.78E+02, 4.71E+02, 8.19E+02, 9.56E+02, 6.12E+02, 3.31E+02, 4.51E+02, 2.05E+02, 2.32E+02, 7.21E+02, 6.37E+02, 7.57E+02, 2.49E+02, 9.63E+02, 3.95E+02, 3.30E+01, 6.04E+02, 5.66E+02, 6.17E+02, 7.87E+02, 2.99E+02, 7.13E+02, 2.79E+02, 2.04E+02, 7.67E+02, 7.77E+02, 9.42E+02, 1.45E+02, 6.84E+02, 3.05E+02, 8.10E+01, 1.85E+02, 5.91E+02, 3.55E+02, 7.47E+02, 4.01E+02, 8.04E+02, 5.83E+02, 7.20E+02, 3.47E+02, 2.22E+02, 3.08E+02, 5.85E+02, 7.04E+02, 6.24E+02, 9.94E+02, 2.71E+02, 6.57E+02, 5.78E+02, 6.09E+02, 3.68E+02, 8.65E+02, 1.30E+02, 2.64E+02, 4.99E+02, 8.17E+02, 3.74E+02, 0.00E+00, 2.73E+02, 1.03E+02, 9.84E+02, 9.65E+02, 6.16E+02, 3.37E+02, 5.86E+02, 4.11E+02, 9.47E+02, 7.84E+02, 9.14E+02, 1.02E+02, 9.99E+02, 2.77E+02, 7.40E+02, 4.55E+02, 5.20E+01, 2.10E+02, 8.24E+02, 8.25E+02, 4.19E+02, 6.23E+02, 8.49E+02, 9.29E+02, 4.33E+02, 9.18E+02, 2.38E+02, 5.72E+02, 1.12E+02, 6.20E+02, 5.73E+02, 4.76E+02, 2.24E+02, 6.45E+02, 1.74E+02,

N = 500

4.87E+02, 5.38E+02, 2.51E+02, 5.71E+02, 9.88E+02, 4.50E+01, 9.29E+02, 8.13E+02, 5.35E+02, 9.50E+02, 7.26E+02, 4.17E+02, 5.49E+02, 3.68E+02, 2.61E+02, 5.75E+02, 7.58E+02, 3.15E+02, 6.38E+02, 4.02E+02, 2.54E+02, 1.67E+02, 6.54E+02, 4.90E+02, 3.66E+02, 8.40E+02, 1.82E+02, 7.53E+02, 2.63E+02, 4.69E+02, 4.10E+02, 2.90E+02, 2.95E+02, 3.27E+02, 1.90E+02, 4.57E+02, 2.36E+02, 3.49E+02, 3.17E+02, 7.15E+02, 5.23E+02, 1.94E+02, 8.03E+02, 6.90E+01, 2.74E+02, 3.96E+02, 7.00E+01, 1.05E+02, 2.44E+02, 8.42E+02, 4.89E+02, 2.09E+02, 8.41E+02, 6.71E+02, 7.11E+02, 8.86E+02, 9.20E+02, 7.46E+02, 3.94E+02, 7.55E+02, 7.83E+02, 9.23E+02, 4.80E+01, 6.69E+02, 1.86E+02, 9.76E+02, 7.21E+02, 9.96E+02, 3.46E+02, 3.54E+02, 3.37E+02, 1.70E+01, 9.28E+02, 7.67E+02, 6.82E+02, 1.65E+02, 2.32E+02, 4.01E+02, 3.42E+02, 8.50E+02, 5.51E+02, 9.68E+02, 5.87E+02, 9.00E+00, 4.62E+02, 4.13E+02, 7.00E+00, 2.70E+02, 8.27E+02, 2.00E+01, 6.40E+01, 8.63E+02, 8.45E+02, 2.29E+02, 7.01E+02, 6.10E+01, 8.16E+02, 4.83E+02, 8.11E+02, 9.81E+02, 2.90E+01, 5.79E+02, 8.72E+02, 6.03E+02, 3.70E+02, 3.53E+02, 4.59E+02, 6.93E+02, 9.01E+02, 6.14E+02, 9.95E+02, 3.47E+02, 1.99E+02, 5.12E+02, 3.13E+02, 3.99E+02, 8.89E+02, 4.05E+02, 1.38E+02, 8.10E+02, 8.57E+02, 3.33E+02, 4.58E+02, 3.03E+02, 9.35E+02, 7.30E+01, 2.22E+02, 5.18E+02,

8.88E+02, 2.89E+02, 6.86E+02, 5.96E+02, 3.11E+02, 6.79E+02, 5.08E+02, 6.00E+01,
8.12E+02, 2.98E+02, 5.04E+02, 7.16E+02, 1.50E+02, 8.83E+02, 5.40E+02, 7.44E+02,
5.20E+02, 2.60E+01, 1.21E+02, 7.00E+02, 7.86E+02, 7.80E+01, 1.75E+02, 8.30E+02,
5.00E+01, 4.85E+02, 5.88E+02, 8.80E+02, 6.46E+02, 2.99E+02, 3.77E+02, 1.96E+02,
9.54E+02, 1.71E+02, 3.04E+02, 7.29E+02, 1.31E+02, 7.28E+02, 6.75E+02, 1.66E+02,
3.92E+02, 2.43E+02, 7.09E+02, 7.50E+01, 8.44E+02, 7.40E+02, 1.88E+02, 2.30E+01,
7.56E+02, 3.59E+02, 7.95E+02, 6.43E+02, 6.13E+02, 9.40E+02, 6.76E+02, 3.24E+02,
8.64E+02, 2.79E+02, 2.80E+02, 5.77E+02, 6.80E+01, 7.32E+02, 5.16E+02, 2.45E+02,
7.24E+02, 1.40E+01, 9.93E+02, 7.65E+02, 5.53E+02, 8.01E+02, 2.18E+02, 5.94E+02,
6.05E+02, 2.42E+02, 4.32E+02, 2.00E+00, 7.12E+02, 4.48E+02, 4.78E+02, 9.43E+02,
7.59E+02, 5.37E+02, 9.60E+02, 9.30E+01, 3.71E+02, 1.17E+02, 7.94E+02, 1.39E+02,
6.28E+02, 1.61E+02, 8.33E+02, 2.12E+02, 6.91E+02, 7.64E+02, 3.93E+02, 7.14E+02,
1.11E+02, 1.20E+02, 4.92E+02, 2.97E+02, 8.61E+02, 3.80E+01, 9.57E+02, 5.91E+02,
9.63E+02, 1.98E+02, 5.44E+02, 7.42E+02, 5.46E+02, 4.28E+02, 3.76E+02, 1.83E+02,
6.08E+02, 2.96E+02, 2.77E+02, 7.71E+02, 5.10E+02, 5.15E+02, 1.37E+02, 5.54E+02,
7.30E+02, 4.95E+02, 3.89E+02, 9.52E+02, 2.92E+02, 1.42E+02, 7.13E+02, 7.10E+01,
6.85E+02, 6.98E+02, 7.38E+02, 7.33E+02, 6.42E+02, 9.46E+02, 4.22E+02, 7.08E+02,
5.00E+00, 5.72E+02, 4.47E+02, 1.43E+02, 1.29E+02, 4.94E+02, 6.02E+02, 5.74E+02,
1.16E+02, 6.39E+02, 3.75E+02, 4.81E+02, 1.90E+01, 4.54E+02, 3.62E+02, 9.25E+02,
1.23E+02, 4.14E+02, 2.86E+02, 4.04E+02, 6.20E+01, 2.60E+02, 4.00E+02, 9.80E+02,
9.24E+02, 1.80E+02, 1.25E+02, 9.83E+02, 8.94E+02, 1.18E+02, 7.49E+02, 1.30E+02,
2.03E+02, 9.12E+02, 5.36E+02, 3.74E+02, 2.07E+02, 2.37E+02, 6.94E+02, 7.20E+02,
9.82E+02, 5.27E+02, 8.43E+02, 5.63E+02, 4.31E+02, 9.50E+01, 4.34E+02, 4.84E+02,
8.90E+01, 1.33E+02, 5.19E+02, 8.49E+02, 9.21E+02, 3.83E+02, 5.10E+01, 4.72E+02,
6.31E+02, 2.52E+02, 9.55E+02, 2.27E+02, 4.24E+02, 1.47E+02, 5.89E+02, 4.26E+02,
6.18E+02, 7.80E+02, 2.00E+02, 8.80E+01, 7.19E+02, 6.49E+02, 6.36E+02, 1.54E+02,
3.30E+02, 1.46E+02, 5.24E+02, 9.92E+02, 2.05E+02, 8.07E+02, 3.60E+01, 2.88E+02,
4.86E+02, 3.12E+02, 6.15E+02, 6.90E+02, 3.07E+02, 6.58E+02, 9.10E+02, 1.04E+02,
5.17E+02, 5.67E+02, 7.72E+02, 1.02E+02, 8.35E+02, 6.26E+02, 9.04E+02, 6.53E+02,
8.21E+02, 4.63E+02, 3.72E+02, 4.37E+02, 1.35E+02, 2.59E+02, 5.85E+02, 3.19E+02,
4.20E+01, 8.95E+02, 5.06E+02, 4.40E+02, 5.95E+02, 1.49E+02, 8.28E+02, 8.91E+02,
7.74E+02, 4.50E+02, 5.13E+02, 1.22E+02, 2.55E+02, 4.88E+02, 4.36E+02, 1.41E+02,
4.52E+02, 1.50E+01, 8.05E+02, 7.37E+02, 5.56E+02, 1.89E+02, 3.78E+02, 9.90E+02,
4.07E+02, 8.02E+02, 1.26E+02, 6.66E+02, 4.18E+02, 6.60E+02, 7.78E+02, 3.31E+02,
2.67E+02, 9.91E+02, 7.45E+02, 4.91E+02, 9.11E+02, 9.08E+02, 5.62E+02, 4.73E+02,
9.66E+02, 1.64E+02, 9.36E+02, 7.48E+02, 9.59E+02, 7.50E+02, 2.62E+02, 4.42E+02,
8.08E+02, 2.66E+02, 2.58E+02, 1.10E+01, 2.11E+02, 7.20E+01, 5.60E+02, 9.03E+02,
9.32E+02, 3.00E+00, 9.02E+02, 2.40E+02, 8.10E+01, 4.82E+02, 3.48E+02, 7.39E+02,
3.64E+02, 6.95E+02, 1.12E+02, 9.39E+02, 8.17E+02, 8.87E+02, 8.98E+02, 3.57E+02,
6.09E+02, 6.87E+02, 1.95E+02, 7.47E+02, 1.00E+01, 4.75E+02, 7.17E+02, 2.98E+02,
9.70E+01, 6.19E+02, 2.28E+02, 5.29E+02, 4.55E+02, 4.96E+02, 6.60E+01, 4.00E+01,
5.42E+02, 3.85E+02, 2.64E+02, 8.56E+02, 5.68E+02, 5.99E+02, 6.57E+02, 1.40E+02,
3.69E+02, 3.91E+02, 3.44E+02, 4.97E+02, 4.90E+01, 6.23E+02, 5.20E+01, 4.44E+02,
5.33E+02, 9.80E+01, 7.93E+02, 5.52E+02, 1.48E+02, 6.35E+02, 8.55E+02, 4.66E+02,
8.77E+02, 9.26E+02, 7.63E+02, 3.58E+02, 1.73E+02, 3.90E+02, 5.01E+02, 6.48E+02,
1.93E+02, 8.65E+02, 4.40E+01, 9.94E+02, 2.82E+02, 9.40E+01, 1.92E+02, 5.64E+02,
7.66E+02, 2.35E+02, 6.10E+02, 3.86E+02,

N = 750

3.52E+02, 7.58E+02, 4.66E+02, 1.60E+01, 6.45E+02, 9.10E+02, 1.00E+01, 8.12E+02,
2.23E+02, 1.10E+01, 5.46E+02, 7.24E+02, 4.54E+02, 9.91E+02, 3.85E+02, 9.64E+02,
3.31E+02, 9.50E+01, 9.31E+02, 5.58E+02, 9.18E+02, 9.09E+02, 4.86E+02, 5.90E+01,
3.47E+02, 7.60E+02, 3.48E+02, 5.99E+02, 4.98E+02, 6.13E+02, 2.13E+02, 2.68E+02,
8.04E+02, 2.22E+02, 2.20E+01, 7.95E+02, 1.08E+02, 6.24E+02, 2.00E+02, 4.47E+02,
5.36E+02, 3.84E+02, 0.00E+00, 8.88E+02, 6.33E+02, 4.06E+02, 9.34E+02, 7.23E+02,
7.29E+02, 6.06E+02, 2.42E+02, 4.20E+02, 4.46E+02, 5.21E+02, 7.36E+02, 2.85E+02,
8.27E+02, 6.93E+02, 9.60E+01, 7.25E+02, 7.66E+02, 3.91E+02, 7.69E+02, 7.80E+01,
8.73E+02, 5.00E+01, 4.41E+02, 1.77E+02, 3.11E+02, 8.80E+01, 3.13E+02, 5.53E+02,
9.46E+02, 5.35E+02, 8.41E+02, 7.28E+02, 9.13E+02, 6.61E+02, 3.45E+02, 6.90E+01,
3.99E+02, 7.86E+02, 4.99E+02, 1.02E+02, 6.54E+02, 4.02E+02, 1.13E+02, 3.97E+02,
3.90E+01, 4.01E+02, 9.51E+02, 3.24E+02, 7.30E+02, 6.12E+02, 2.17E+02, 3.27E+02,
4.60E+02, 4.61E+02, 8.35E+02, 5.42E+02, 2.64E+02, 8.61E+02, 7.92E+02, 5.94E+02,
9.06E+02, 8.56E+02, 8.50E+01, 4.95E+02, 8.60E+02, 7.14E+02, 4.14E+02, 6.03E+02,
7.84E+02, 4.19E+02, 7.59E+02, 7.99E+02, 2.49E+02, 2.67E+02, 2.32E+02, 7.04E+02,
9.78E+02, 9.10E+01, 4.20E+01, 9.50E+02, 1.07E+02, 2.12E+02, 1.46E+02, 4.10E+02,
5.61E+02, 5.88E+02, 2.56E+02, 9.70E+02, 1.94E+02, 6.86E+02, 1.78E+02, 9.95E+02,
8.20E+01, 6.50E+01, 8.02E+02, 8.28E+02, 9.20E+02, 1.72E+02, 7.00E+02, 1.66E+02,
3.50E+01, 8.37E+02, 9.12E+02, 5.66E+02, 3.28E+02, 8.23E+02, 9.55E+02, 6.83E+02,

```
4.44E+02,  5.60E+01,  3.75E+02,  5.30E+01,  3.32E+02,  7.70E+02,  1.92E+02,  5.22E+02,
7.64E+02,  8.87E+02,  9.17E+02,  4.80E+01,  1.99E+02,  8.34E+02,  3.18E+02,  1.68E+02,
9.56E+02,  3.89E+02,  8.42E+02,  7.12E+02,  5.83E+02,  4.76E+02,  5.12E+02,  5.14E+02,
2.80E+02,  8.52E+02,  1.87E+02,  7.62E+02,  6.21E+02,  9.85E+02,  6.11E+02,  1.12E+02,
7.00E+00,  7.51E+02,  2.36E+02,  6.58E+02,  2.55E+02,  1.70E+01,  5.23E+02,  6.00E+02,
4.23E+02,  2.18E+02,  5.37E+02,  8.05E+02,  3.15E+02,  8.50E+02,  5.92E+02,  2.89E+02,
5.09E+02,  9.38E+02,  3.73E+02,  2.60E+01,  9.47E+02,  4.40E+01,  5.68E+02,  6.28E+02,
5.67E+02,  4.78E+02,  2.90E+01,  3.60E+02,  1.49E+02,  7.02E+02,  1.10E+02,  1.27E+02,
5.60E+02,  5.69E+02,  3.80E+02,  5.24E+02,  8.31E+02,  8.64E+02,  8.92E+02,  5.96E+02,
1.11E+02,  6.14E+02,  1.50E+01,  1.67E+02,  4.90E+02,  8.91E+02,  1.17E+02,  7.26E+02,
4.36E+02,  1.82E+02,  6.65E+02,  2.20E+02,  3.02E+02,  1.40E+01,  9.75E+02,  9.53E+02,
9.71E+02,  8.08E+02,  3.42E+02,  2.00E+00,  6.92E+02,  3.93E+02,  9.86E+02,  8.24E+02,
2.11E+02,  4.18E+02,  8.99E+02,  3.05E+02,  8.66E+02,  3.61E+02,  6.10E+02,  1.58E+02,
2.44E+02,  5.05E+02,  1.14E+02,  2.97E+02,  6.49E+02,  5.48E+02,  3.12E+02,  2.27E+02,
2.69E+02,  8.16E+02,  3.25E+02,  7.34E+02,  3.83E+02,  6.78E+02,  3.94E+02,  7.21E+02,
6.43E+02,  6.68E+02,  2.38E+02,  8.89E+02,  3.35E+02,  1.26E+02,  4.04E+02,  4.33E+02,
1.48E+02,  4.92E+02,  6.67E+02,  5.32E+02,  2.72E+02,  8.45E+02,  6.57E+02,  7.94E+02,
1.40E+02,  2.83E+02,  2.50E+02,  5.75E+02,  9.44E+02,  4.13E+02,  5.80E+01,  8.67E+02,
9.87E+02,  3.46E+02,  4.03E+02,  9.36E+02,  1.81E+02,  8.57E+02,  4.26E+02,  1.36E+02,
5.20E+02,  4.30E+02,  3.44E+02,  4.52E+02,  2.77E+02,  2.10E+02,  9.82E+02,  7.81E+02,
8.55E+02,  1.35E+02,  9.26E+02,  1.95E+02,  1.38E+02,  3.08E+02,  2.29E+02,  4.00E+02,
5.98E+02,  6.23E+02,  8.01E+02,  9.57E+02,  3.30E+02,  8.77E+02,  6.16E+02,  9.94E+02,
1.01E+02,  8.85E+02,  8.30E+02,  6.36E+02,  8.47E+02,  6.80E+02,  1.89E+02,  5.11E+02,
2.73E+02,  7.19E+02,  4.81E+02,  9.02E+02,  5.76E+02,  6.15E+02,  1.97E+02,  7.57E+02,
5.77E+02,  1.75E+02,  4.51E+02,  7.63E+02,  2.30E+01,  3.87E+02,  2.10E+01,  8.93E+02,
2.88E+02,  9.52E+02,  2.91E+02,  5.50E+02,  4.28E+02,  7.06E+02,  5.52E+02,  9.81E+02,
1.00E+00,  4.08E+02,  2.79E+02,  2.31E+02,  3.96E+02,  1.24E+02,  3.09E+02,  8.69E+02,
2.84E+02,  6.17E+02,  3.76E+02,  2.93E+02,  8.39E+02,  1.59E+02,  2.96E+02,  7.40E+02,
7.60E+01,  4.16E+02,  5.49E+02,  8.26E+02,  4.43E+02,  7.16E+02,  1.22E+02,  3.38E+02,
5.27E+02,  6.40E+02,  7.18E+02,  1.43E+02,  2.87E+02,  3.22E+02,  6.75E+02,  9.08E+02,
3.34E+02,  4.96E+02,  9.00E+01,  4.77E+02,  9.01E+02,  8.86E+02,  4.63E+02,  3.26E+02,
3.95E+02,  9.74E+02,  8.90E+01,  5.26E+02,  4.80E+02,  6.26E+02,  9.29E+02,  5.57E+02,
6.63E+02,  7.73E+02,  6.53E+02,  8.00E+02,  2.71E+02,  9.93E+02,  9.84E+02,  1.54E+02,
1.20E+02,  2.66E+02,  9.27E+02,  6.22E+02,  2.03E+02,  3.51E+02,  6.29E+02,  5.41E+02,
2.19E+02,  8.53E+02,  2.82E+02,  7.31E+02,  6.72E+02,  9.49E+02,  3.00E+02,  8.32E+02,
4.89E+02,  6.01E+02,  5.33E+02,  9.54E+02,  8.46E+02,  9.72E+02,  4.29E+02,  9.33E+02,
6.10E+01,  5.65E+02,  1.03E+02,  1.39E+02,  9.73E+02,  8.14E+02,  1.55E+02,  5.72E+02,
6.08E+02,  1.76E+02,  7.70E+01,  3.14E+02,  5.89E+02,  1.09E+02,  6.09E+02,  4.71E+02,
4.40E+02,  4.10E+01,  6.90E+02,  9.69E+02,  9.92E+02,  2.08E+02,  8.40E+02,  3.86E+02,
6.27E+02,  9.62E+02,  2.40E+02,  4.88E+02,  9.04E+02,  1.60E+02,  8.11E+02,  5.29E+02,
6.37E+02,  3.43E+02,  3.78E+02,  6.38E+02,  5.91E+02,  4.50E+02,  8.10E+01,  3.23E+02,
1.06E+02,  9.40E+01,  6.25E+02,  7.08E+02,  1.80E+01,  9.25E+02,  6.96E+02,  9.63E+02,
1.74E+02,  9.32E+02,  4.85E+02,  8.95E+02,  3.36E+02,  4.82E+02,  9.28E+02,  4.74E+02,
2.43E+02,  6.64E+02,  4.87E+02,  5.74E+02,  3.00E+01,  7.47E+02,  5.62E+02,  2.92E+02,
8.60E+01,  8.54E+02,  2.70E+01,  9.77E+02,  2.41E+02,  9.67E+02,  2.26E+02,  9.68E+02,
3.71E+02,  8.94E+02,  2.81E+02,  5.63E+02,  2.06E+02,  2.34E+02,  7.44E+02,  1.70E+02,
9.19E+02,  1.18E+02,  4.17E+02,  8.29E+02,  2.53E+02,  9.15E+02,  2.99E+02,  8.71E+02,
9.23E+02,  4.72E+02,  9.99E+02,  4.59E+02,  6.89E+02,  3.39E+02,  9.90E+02,  6.46E+02,
2.48E+02,  5.64E+02,  2.51E+02,  2.76E+02,  3.30E+01,  9.16E+02,  5.79E+02,  2.16E+02,
8.00E+01,  8.75E+02,  7.03E+02,  3.17E+02,  5.82E+02,  1.15E+02,  9.00E+00,  4.42E+02,
2.86E+02,  7.32E+02,  1.00E+02,  6.77E+02,  9.96E+02,  6.94E+02,  8.36E+02,  5.03E+02,
1.93E+02,  7.68E+02,  5.18E+02,  5.15E+02,  6.66E+02,  3.19E+02,  2.58E+02,  2.95E+02,
9.35E+02,  4.15E+02,  7.22E+02,  7.38E+02,  8.80E+02,  6.98E+02,  2.02E+02,  3.29E+02,
3.10E+02,  3.37E+02,  1.90E+01,  9.80E+02,  3.60E+01,  4.45E+02,  3.70E+02,  4.07E+02,
9.11E+02,  6.04E+02,  6.48E+02,  7.82E+02,  2.65E+02,  8.40E+01,  8.30E+01,  2.75E+02,
1.80E+02,  8.17E+02,  4.62E+02,  2.63E+02,  6.80E+01,  7.78E+02,  7.79E+02,  8.06E+02,
7.42E+02,  6.20E+02,  3.56E+02,  9.83E+02,  7.45E+02,  1.62E+02,  7.75E+02,  6.76E+02,
3.64E+02,  8.84E+02,  8.44E+02,  1.41E+02,  5.16E+02,  6.40E+01,  3.16E+02,  7.39E+02,
5.04E+02,  4.60E+01,  7.11E+02,  9.88E+02,  5.50E+01,  1.44E+02,  8.74E+02,  4.24E+02,
2.30E+02,  4.25E+02,  9.37E+02,  1.20E+01,  6.31E+02,  4.49E+02,  7.87E+02,  6.47E+02,
8.38E+02,  6.56E+02,  3.50E+02,  4.05E+02,  1.32E+02,  6.95E+02,  5.02E+02,  9.30E+01,
7.65E+02,  8.72E+02,  8.15E+02,  5.71E+02,  2.47E+02,  9.58E+02,  4.30E+01,  6.99E+02,
8.03E+02,  3.79E+02,  9.24E+02,  3.67E+02,  9.66E+02,  1.96E+02,  9.43E+02,  4.93E+02,
6.60E+01,  4.38E+02,  5.30E+02,  1.83E+02,  1.79E+02,  2.70E+02,  1.71E+02,  9.39E+02,
6.81E+02,  1.29E+02,  8.25E+02,  1.84E+02,  2.90E+02,  6.50E+02,  2.39E+02,  4.09E+02,
7.98E+02,  9.89E+02,  1.91E+02,  5.70E+01,  8.20E+02,  3.65E+02,  4.55E+02,  7.72E+02,
7.76E+02,  8.07E+02,  1.88E+02,  6.07E+02,  7.90E+01,  6.42E+02,  4.73E+02,  8.76E+02,
```

1.05E+02, 3.68E+02, 3.77E+02, 2.80E+01, 3.58E+02, 1.63E+02, 1.86E+02, 6.02E+02,
7.10E+01, 4.32E+02, 1.61E+02, 3.66E+02, 5.34E+02, 2.62E+02, 6.70E+01, 7.05E+02,
8.82E+02, 9.42E+02, 7.49E+02, 5.55E+02, 5.80E+02, 9.79E+02, 5.87E+02, 4.56E+02,
1.33E+02, 5.45E+02, 1.34E+02, 7.90E+02, 9.80E+01, 1.65E+02, 8.97E+02, 3.92E+02,
8.22E+02, 2.54E+02, 5.90E+02, 8.78E+02, 2.59E+02, 8.83E+02, 1.30E+01, 5.28E+02,
7.93E+02, 7.01E+02, 6.44E+02, 2.28E+02, 9.76E+02, 7.27E+02, 7.00E+01, 3.72E+02,
1.23E+02, 9.65E+02, 4.70E+01, 1.51E+02, 5.40E+01, 5.44E+02, 4.97E+02, 8.14E+02,
4.69E+02, 2.60E+02, 2.98E+02, 2.21E+02, 7.35E+02, 6.82E+02,
N = 1000
2.70E+02, 9.69E+02, 4.84E+02, 4.25E+02, 8.93E+02, 2.07E+02, 2.60E+01, 1.39E+02,
5.66E+02, 1.20E+01, 1.99E+02, 9.79E+02, 1.28E+02, 9.90E+01, 4.51E+02, 6.92E+02,
1.80E+01, 3.43E+02, 1.14E+02, 4.04E+02, 5.92E+02, 9.36E+02, 2.69E+02, 8.46E+02,
7.01E+02, 3.23E+02, 2.00E+02, 5.83E+02, 5.37E+02, 8.77E+02, 8.74E+02, 9.83E+02,
3.97E+02, 2.16E+02, 2.98E+02, 6.70E+01, 4.61E+02, 9.05E+02, 9.48E+02, 1.46E+02,
3.53E+02, 4.50E+01, 4.66E+02, 8.86E+02, 5.91E+02, 3.62E+02, 6.56E+02, 2.23E+02,
3.22E+02, 8.35E+02, 5.52E+02, 4.05E+02, 2.31E+02, 2.73E+02, 3.67E+02, 9.39E+02,
8.96E+02, 4.72E+02, 4.56E+02, 9.90E+02, 1.07E+02, 9.93E+02, 8.33E+02, 7.00E+02,
3.84E+02, 7.91E+02, 4.90E+02, 2.50E+01, 8.05E+02, 4.98E+02, 6.76E+02, 6.17E+02,
8.16E+02, 8.09E+02, 7.69E+02, 3.31E+02, 5.29E+02, 4.81E+02, 7.51E+02, 1.67E+02,
7.17E+02, 9.71E+02, 1.19E+02, 7.97E+02, 1.66E+02, 3.54E+02, 8.72E+02, 6.20E+02,
8.24E+02, 4.67E+02, 1.30E+01, 6.95E+02, 6.99E+02, 6.03E+02, 3.55E+02, 3.96E+02,
8.90E+01, 2.66E+02, 2.00E+00, 1.08E+02, 8.44E+02, 2.40E+02, 5.89E+02, 4.12E+02,
9.01E+02, 2.22E+02, 3.36E+02, 6.87E+02, 7.30E+01, 5.74E+02, 8.75E+02, 2.75E+02,
6.31E+02, 4.40E+02, 3.35E+02, 9.60E+02, 8.98E+02, 4.60E+02, 9.06E+02, 3.00E+01,
7.63E+02, 8.26E+02, 3.83E+02, 1.17E+02, 7.40E+01, 3.06E+02, 7.30E+02, 7.56E+02,
8.97E+02, 3.28E+02, 9.78E+02, 8.08E+02, 9.58E+02, 9.47E+02, 3.50E+01, 3.76E+02,
1.23E+02, 7.54E+02, 6.04E+02, 4.59E+02, 7.76E+02, 8.62E+02, 1.95E+02, 1.81E+02,
4.93E+02, 4.10E+01, 7.38E+02, 6.98E+02, 7.79E+02, 8.56E+02, 1.64E+02, 1.45E+02,
3.04E+02, 5.41E+02, 8.10E+01, 4.43E+02, 3.51E+02, 7.39E+02, 8.22E+02, 0.00E+00,
9.98E+02, 8.68E+02, 1.50E+02, 7.24E+02, 5.58E+02, 7.50E+02, 7.65E+02, 2.30E+02,
9.65E+02, 4.22E+02, 6.77E+02, 3.45E+02, 2.13E+02, 3.79E+02, 7.36E+02, 7.98E+02,
3.30E+01, 2.97E+02, 7.37E+02, 4.80E+02, 5.42E+02, 4.60E+01, 9.50E+02, 9.51E+02,
2.54E+02, 4.02E+02, 9.34E+02, 9.07E+02, 9.31E+02, 4.71E+02, 9.96E+02, 7.34E+02,
1.70E+02, 4.89E+02, 3.21E+02, 2.91E+02, 3.11E+02, 4.20E+01, 3.89E+02, 2.09E+02,
4.80E+01, 7.50E+01, 3.19E+02, 8.89E+02, 2.77E+02, 4.82E+02, 1.26E+02, 1.44E+02,
7.47E+02, 2.10E+01, 2.64E+02, 5.50E+01, 2.38E+02, 8.64E+02, 5.08E+02, 6.02E+02,
9.38E+02, 8.18E+02, 3.98E+02, 9.88E+02, 9.86E+02, 8.82E+02, 2.85E+02, 3.46E+02,
7.59E+02, 4.62E+02, 9.26E+02, 1.22E+02, 9.40E+01, 5.63E+02, 4.55E+02, 9.72E+02,
6.49E+02, 2.30E+01, 7.05E+02, 1.16E+02, 7.71E+02, 8.04E+02, 8.40E+01, 3.44E+02,
1.31E+02, 6.59E+02, 3.93E+02, 2.63E+02, 6.00E+00, 9.85E+02, 1.90E+01, 6.06E+02,
3.07E+02, 7.95E+02, 9.70E+01, 6.22E+02, 6.32E+02, 7.80E+01, 5.07E+02, 6.53E+02,
8.70E+02, 7.22E+02, 3.15E+02, 6.52E+02, 4.00E+02, 1.65E+02, 5.21E+02, 8.21E+02,
7.15E+02, 4.30E+02, 3.64E+02, 5.60E+01, 7.60E+01, 6.34E+02, 8.50E+01, 6.26E+02,
4.74E+02, 9.56E+02, 8.61E+02, 9.73E+02, 8.87E+02, 8.73E+02, 6.60E+02, 1.41E+02,
6.44E+02, 5.14E+02, 7.75E+02, 8.53E+02, 5.46E+02, 1.86E+02, 3.66E+02, 6.24E+02,
1.48E+02, 4.69E+02, 9.09E+02, 1.90E+02, 8.66E+02, 3.88E+02, 5.03E+02, 5.02E+02,
7.41E+02, 1.84E+02, 7.44E+02, 2.33E+02, 5.60E+02, 8.07E+02, 8.27E+02, 5.50E+02,
2.00E+01, 2.88E+02, 8.58E+02, 2.25E+02, 2.95E+02, 5.16E+02, 6.79E+02, 1.15E+02,
7.06E+02, 6.36E+02, 2.68E+02, 1.00E+00, 3.59E+02, 5.57E+02, 8.94E+02, 1.89E+02,
3.87E+02, 4.94E+02, 4.15E+02, 9.75E+02, 9.76E+02, 1.59E+02, 4.77E+02, 9.29E+02,
5.01E+02, 7.73E+02, 2.26E+02, 5.84E+02, 1.09E+02, 2.32E+02, 8.06E+02, 2.78E+02,
8.00E+01, 5.25E+02, 4.86E+02, 8.12E+02, 4.79E+02, 2.42E+02, 7.48E+02, 4.07E+02,
3.99E+02, 8.92E+02, 1.88E+02, 5.93E+02, 7.35E+02, 9.70E+02, 4.57E+02, 9.55E+02,
3.60E+01, 7.87E+02, 4.20E+02, 4.16E+02, 4.76E+02, 9.81E+02, 7.46E+02, 9.50E+01,
9.08E+02, 9.04E+02, 1.53E+02, 3.42E+02, 9.17E+02, 3.38E+02, 1.00E+01, 5.09E+02,
1.30E+02, 4.46E+02, 8.14E+02, 8.41E+02, 7.57E+02, 4.97E+02, 1.79E+02, 3.20E+02,
8.91E+02, 6.21E+02, 9.02E+02, 8.79E+02, 4.42E+02, 3.78E+02, 8.80E+02, 8.03E+02,
7.03E+02, 4.00E+00, 6.28E+02, 3.20E+01, 4.10E+02, 1.60E+01, 1.10E+02, 4.18E+02,
1.76E+02, 4.49E+02, 5.90E+02, 4.29E+02, 7.60E+02, 8.38E+02, 5.90E+01, 6.61E+02,
5.64E+02, 8.55E+02, 5.82E+02, 2.48E+02, 7.62E+02, 5.88E+02, 3.12E+02, 7.78E+02,
3.17E+02, 5.26E+02, 4.06E+02, 3.74E+02, 3.27E+02, 4.11E+02, 9.54E+02, 5.87E+02,
7.02E+02, 7.49E+02, 9.80E+02, 2.21E+02, 9.19E+02, 1.83E+02, 1.24E+02, 7.11E+02,
1.92E+02, 2.59E+02, 6.69E+02, 1.87E+02, 9.87E+02, 8.48E+02, 3.73E+02, 2.20E+01,
3.58E+02, 8.63E+02, 1.73E+02, 8.69E+02, 6.96E+02, 2.99E+02, 3.70E+02, 4.13E+02,
2.36E+02, 9.44E+02, 1.11E+02, 5.18E+02, 9.12E+02, 5.54E+02, 8.60E+02, 9.20E+02,
3.25E+02, 6.55E+02, 6.86E+02, 2.84E+02, 8.52E+02, 7.72E+02, 6.10E+02, 1.61E+02,
2.96E+02, 3.71E+02, 6.19E+02, 7.12E+02, 2.44E+02, 9.35E+02, 9.57E+02, 3.57E+02,

```
2.08E+02,  7.84E+02,  3.47E+02,  5.28E+02,  1.13E+02,  4.39E+02,  2.81E+02,  8.51E+02,
4.68E+02,  3.95E+02,  6.81E+02,  6.25E+02,  5.48E+02,  8.30E+02,  2.20E+02,  5.23E+02,
3.80E+01,  2.82E+02,  9.13E+02,  4.75E+02,  9.77E+02,  8.34E+02,  3.10E+02,  2.19E+02,
8.84E+02,  1.58E+02,  7.27E+02,  2.39E+02,  4.54E+02,  9.63E+02,  6.75E+02,  1.12E+02,
9.62E+02,  9.28E+02,  3.90E+02,  3.92E+02,  2.14E+02,  1.77E+02,  9.22E+02,  2.53E+02,
8.25E+02,  7.40E+02,  2.71E+02,  7.00E+01,  3.91E+02,  4.27E+02,  6.91E+02,  1.51E+02,
3.40E+01,  9.80E+01,  9.41E+02,  8.10E+02,  9.97E+02,  5.96E+02,  7.55E+02,  3.82E+02,
5.62E+02,  3.00E+02,  6.80E+01,  6.63E+02,  5.15E+02,  6.13E+02,  3.65E+02,  5.05E+02,
5.30E+02,  1.78E+02,  7.61E+02,  1.04E+02,  7.13E+02,  6.43E+02,  2.49E+02,  9.43E+02,
4.03E+02,  2.89E+02,  2.10E+02,  6.90E+01,  7.68E+02,  1.40E+01,  1.37E+02,  6.15E+02,
4.30E+01,  3.72E+02,  4.53E+02,  8.57E+02,  8.47E+02,  6.58E+02,  2.67E+02,  2.46E+02,
7.00E+00,  7.07E+02,  8.70E+01,  3.13E+02,  7.28E+02,  8.17E+02,  5.10E+02,  8.83E+02,
5.51E+02,  4.34E+02,  2.29E+02,  7.83E+02,  8.29E+02,  9.42E+02,  6.16E+02,  5.10E+01,
8.00E+02,  3.61E+02,  2.52E+02,  5.47E+02,  2.87E+02,  9.30E+02,  2.90E+02,  3.32E+02,
2.37E+02,  3.09E+02,  4.95E+02,  8.19E+02,  7.80E+02,  5.00E+02,  6.78E+02,  2.12E+02,
8.80E+01,  6.50E+01,  9.03E+02,  7.81E+02,  1.60E+02,  6.74E+02,  7.67E+02,  1.27E+02,
4.26E+02,  5.35E+02,  2.80E+01,  3.08E+02,  5.32E+02,  9.99E+02,  2.74E+02,  5.55E+02,
5.12E+02,  7.52E+02,  6.08E+02,  1.72E+02,  9.49E+02,  7.31E+02,  4.01E+02,  4.50E+02,
3.50E+02,  5.00E+00,  8.31E+02,  4.36E+02,  8.11E+02,  9.21E+02,  4.85E+02,  6.54E+02,
3.85E+02,  6.57E+02,  4.87E+02,  3.26E+02,  6.05E+02,  8.42E+02,  9.94E+02,  3.16E+02,
1.18E+02,  9.23E+02,  8.85E+02,  6.07E+02,  4.96E+02,  4.32E+02,  3.81E+02,  8.99E+02,
9.18E+02,  1.75E+02,  6.01E+02,  7.53E+02,  4.48E+02,  2.43E+02,  2.02E+02,  4.14E+02,
6.40E+01,  8.40E+02,  7.20E+01,  2.94E+02,  4.31E+02,  1.05E+02,  9.24E+02,  5.00E+01,
6.42E+02,  4.73E+02,  3.41E+02,  4.17E+02,  4.40E+01,  5.38E+02,  9.25E+02,  5.78E+02,
1.47E+02,  9.53E+02,  9.10E+02,  2.57E+02,  6.80E+02,  5.34E+02,  2.72E+02,  9.74E+02,
9.37E+02,  5.40E+01,  3.01E+02,  6.83E+02,  2.41E+02,  1.56E+02,  5.98E+02,  1.54E+02,
1.38E+02,  5.86E+02,  6.47E+02,  1.00E+02,  5.76E+02,  2.27E+02,  7.21E+02,  6.93E+02,
6.60E+01,  8.15E+02,  6.00E+02,  4.47E+02,  5.19E+02,  4.41E+02,  5.44E+02,  9.30E+01,
4.90E+01,  2.61E+02,  7.96E+02,  1.55E+02,  7.42E+02,  5.65E+02,  7.25E+02,  9.11E+02,
2.83E+02,  7.82E+02,  1.33E+02,  5.20E+02,  8.32E+02,  3.14E+02,  3.86E+02,  1.80E+02,
7.70E+01,  5.24E+02,  6.82E+02,  7.10E+01,  7.09E+02,  5.17E+02,  3.69E+02,  9.92E+02,
7.90E+01,  9.91E+02,  1.70E+01,  3.40E+02,  9.84E+02,  6.65E+02,  1.42E+02,  3.05E+02,
5.11E+02,  1.40E+02,  5.97E+02,  9.46E+02,  4.91E+02,  4.44E+02,  3.34E+02,  1.32E+02,
9.66E+02,  6.40E+02,  5.31E+02,  6.30E+01,  1.35E+02,  4.33E+02,  3.33E+02,  3.03E+02,
7.08E+02,  9.00E+00,  6.85E+02,  6.88E+02,  6.09E+02,  6.35E+02,  6.38E+02,  8.23E+02,
5.33E+02,  8.81E+02,  5.22E+02,  7.16E+02,  2.24E+02,  8.02E+02,  8.54E+02,  9.14E+02,
4.28E+02,  4.21E+02,  7.26E+02,  6.94E+02,  5.69E+02,  2.40E+01,  6.29E+02,  4.70E+01,
5.53E+02,  2.03E+02,  8.13E+02,  1.57E+02,  1.03E+02,  9.60E+01,  3.90E+01,  8.90E+02,
4.99E+02,  7.86E+02,  1.01E+02,  2.62E+02,  1.63E+02,  5.73E+02,  2.04E+02,  9.00E+02,
9.40E+02,  5.13E+02,  9.52E+02,  5.20E+01,  7.77E+02,  8.01E+02,  1.62E+02,  7.66E+02,
3.63E+02,  7.32E+02,  7.43E+02,  7.85E+02,  4.08E+02,  8.36E+02,  8.71E+02,  2.34E+02,
3.80E+02,  2.70E+01,  6.67E+02,  9.45E+02,  6.90E+02,  6.73E+02,  5.80E+02,  4.52E+02,
7.10E+02,  7.33E+02,  7.74E+02,  3.24E+02,  5.27E+02,  2.60E+02,  4.19E+02,  4.45E+02,
2.18E+02,  6.20E+01,  5.70E+01,  8.00E+00,  7.93E+02,  3.68E+02,  7.18E+02,  4.88E+02,
1.34E+02,  5.99E+02,  8.45E+02,  2.90E+01,  7.70E+02,  7.64E+02,  6.30E+02,  5.71E+02,
1.93E+02,  8.67E+02,  2.01E+02,  3.94E+02,  3.30E+02,  5.56E+02,  2.93E+02,  1.52E+02,
4.92E+02,  3.48E+02,  3.75E+02,  8.88E+02,  2.47E+02,  8.50E+02,  2.06E+02,  4.37E+02,
8.95E+02,  5.94E+02,  9.10E+01,  6.51E+02,  6.89E+02,  8.78E+02,  5.59E+02,  2.92E+02,
2.79E+02,  3.60E+02,  4.00E+01,  8.30E+01,  6.66E+02,  7.29E+02,  9.64E+02,  1.06E+02,
7.04E+02,  7.14E+02,  4.64E+02,  6.10E+01,  2.76E+02,  8.43E+02,  3.56E+02,  2.80E+02,
4.38E+02,  1.25E+02,  6.48E+02,  5.68E+02,  1.97E+02,  2.58E+02,  5.95E+02,  1.29E+02,
8.65E+02,  2.51E+02,  1.36E+02,  7.90E+02,  4.65E+02,  4.70E+02,  4.23E+02,  7.88E+02,
1.21E+02,  2.55E+02,  7.89E+02,  5.36E+02,  7.58E+02,  2.05E+02,  3.10E+01,  3.70E+01,
3.02E+02,  3.77E+02,  3.52E+02,  5.45E+02,  6.64E+02,  5.75E+02,  5.39E+02,  1.68E+02,
1.94E+02,  5.79E+02,  6.41E+02,  7.23E+02,  4.63E+02,  7.92E+02,  5.72E+02,  8.49E+02,
9.61E+02,  6.71E+02,  2.56E+02,  5.70E+02,  7.94E+02,  6.70E+02,  2.35E+02,  7.20E+02,
8.60E+01,  9.68E+02,  8.20E+01,  5.04E+02,  2.65E+02,  8.59E+02,  9.95E+02,  6.46E+02,
5.40E+02,  4.83E+02,  1.96E+02,  3.39E+02,  9.82E+02,  1.91E+02,  8.20E+02,  7.45E+02,
6.18E+02,  6.62E+02,  3.37E+02,  4.09E+02,  1.02E+02,  6.11E+02,  9.00E+01,  5.77E+02,
1.98E+02,  6.27E+02,  2.50E+02,  3.18E+02,  2.28E+02,  6.12E+02,  9.33E+02,  6.72E+02,
1.74E+02,  2.15E+02,  5.06E+02,  6.50E+02,  6.00E+01,  3.29E+02,  5.30E+01,  6.68E+02,
8.37E+02,  9.15E+02,  2.45E+02,  8.39E+02,  9.89E+02,  6.84E+02,  1.71E+02,  3.49E+02,
2.86E+02,  6.39E+02,  2.11E+02,  6.14E+02,  1.43E+02,  1.50E+01,  9.20E+01,  6.23E+02,
7.99E+02,  1.69E+02,  1.20E+02,  5.49E+02,  5.43E+02,  7.40E+02,  4.58E+02,  5.67E+02,
7.19E+02,  2.17E+02,  6.45E+02,  9.59E+02,  8.76E+02,  4.78E+02,  9.27E+02,  5.61E+02,
5.85E+02,  6.97E+02,  6.33E+02,  9.16E+02,  4.35E+02,  9.67E+02,  4.24E+02,  5.81E+02,
1.10E+01, 1.85E+02, 6.37E+02, 1.49E+02, 9.32E+02, 3.00E+00, 1.82E+02, 8.28E+02,
```

iii.Time

data	Duplicate, Ordinary	Duplicate, Presort
10	0.000228292942047119	0.000360133647918701
50	0.000416738986968994	0.000224626064300537
100	0.000409743785858154	0.000339040756225585
250	0.000387382507324218	0.000309855937957763
500	0.00255593538284301	0.000342123508453369
750	0.0101350498199462	0.000397365093231201
1000	0.0173480463027954	0.00028738260269165

data	Unique, Ordinary	Unique, Presort
10	0.000267322063446044	0.000329000949859619
50	0.000310072898864746	0.000382783412933349
100	0.000416445732116699	0.000229182243347167
250	0.00126860618591308	0.00030754804611206
500	0.00129709482192993	0.000328497886657714
750	0.0106589555740356	0.000346462726593017
1000	0.0190326261520385	0.00027144432067871

VII.ข้อมูลกราฟ 6.8 และ 6.9

N = 10

377, 405, 652, 466, 430, 807, 121, 596, 489, 5

N = 50

532, 33, 502, 266, 872, 210, 931, 370, 686, 524, 168, 303, 290, 855, 976, 416, 418, 25, 77, 88, 562, 985, 893, 319, 729, 561, 839, 808, 332, 938, 349, 685, 554, 700, 899, 903, 83, 356, 214, 177, 24, 550, 334, 346, 103, 917, 193, 190, 508, 980

N = 100

753, 508, 968, 718, 760, 75, 550, 426, 259, 916, 639, 30, 454, 402, 154, 313, 292, 701, 525, 398, 516, 272, 849, 530, 942, 582, 751, 159, 139, 912, 633, 612, 754, 704, 94, 104, 858, 679, 621, 520, 213, 568, 127, 230, 791, 791, 112, 424, 266, 797, 76, 38, 176, 940, 119, 751, 847, 324, 666, 369, 621, 804, 704, 58, 6, 454, 336, 263, 102, 471, 510, 158, 133, 787, 52, 519, 292, 78, 374, 907, 170, 901, 897, 167, 624, 873, 868, 630, 510, 837, 42, 458, 620, 516, 988, 180, 253, 317, 581, 910

N = 1000

891, 184, 593, 915, 402, 926, 681, 479, 931, 613, 403, 18, 613, 75, 561, 942, 152, 291, 764, 216, 368, 339, 327, 626, 709, 488, 934, 580, 997, 514, 789, 892, 354, 650, 151, 478, 622, 676, 69, 915, 676, 661, 14, 69, 311, 803, 980, 210, 409, 501, 136, 55, 50, 150, 834, 95, 899, 579, 940, 436, 984, 139, 416, 734, 392, 485, 912, 954, 520, 428, 690, 113, 498, 697, 502, 541, 425, 47, 157, 593, 413, 493, 255, 735, 461, 461, 942, 280, 797, 43, 356, 174, 143, 362, 578, 383, 985, 533, 710, 317, 546, 722, 924, 998, 376, 79, 25, 687, 927, 460, 688, 234, 796, 800, 231, 227, 851, 289, 704, 576, 589, 224, 905, 989, 266, 758, 452, 364, 249, 665, 397, 562, 785, 35, 721, 722, 227, 386, 153, 731, 604, 86, 221, 593, 378, 890, 789, 100, 229, 716, 111, 762, 15, 820, 54, 282, 25, 839, 711, 928, 546, 510, 884, 842, 430, 85, 282, 546, 413, 526, 817, 874, 616, 383, 738, 442, 21, 496, 428, 265, 176, 699, 631, 869, 700, 854, 500, 310, 655, 773, 161, 868, 301, 290, 215, 627, 357, 523, 569, 145, 836, 649, 894, 937, 611, 579, 750, 570, 814, 445, 480, 321, 933, 880, 927, 40, 377, 991, 10, 671, 733, 420, 71, 404, 691, 300, 256, 201, 127, 645, 200, 999, 357, 607, 256, 314, 629, 170, 462, 459, 704, 917, 611, 972, 534, 926, 768, 972, 951, 180, 719, 64, 780, 61, 903, 954, 80, 733, 232, 848, 107, 630, 930, 54, 976, 768, 730, 126, 302, 591, 543, 592, 891, 558, 359, 16, 426, 265, 734, 843, 590, 292, 501, 5, 806, 917, 430, 669, 286, 761, 27, 143, 259, 762, 280, 39, 312, 725, 905, 870, 555, 448, 773, 378, 16, 701, 495, 400, 659, 393, 321, 107, 3, 453, 250, 763, 193, 906, 381, 642, 927, 977, 346, 138, 594, 351, 419, 78, 756, 116, 106, 93, 275, 261, 714, 65, 91, 932, 135, 57, 164, 974, 923, 97, 896, 519, 127, 976, 648, 373, 901, 894, 68, 768, 991, 288, 469, 114, 155, 9, 546, 212, 240, 184, 243, 531, 769, 664, 689, 79, 703, 24, 376, 225, 891, 852, 717, 205, 313, 650, 21, 462, 36, 44, 965, 706, 487, 630, 781, 690, 126, 207, 133, 768, 183,

593, 110, 724, 556, 994, 759, 303, 868, 450, 527, 765, 238, 273, 17, 974, 230, 520, 242, 55, 929, 109, 337, 184, 267, 369, 196, 138, 550, 824, 749, 816, 649, 574, 243, 343, 877, 548, 1000, 613, 482, 936, 121, 875, 192, 959, 534, 822, 833, 755, 213, 184, 656, 895, 488, 946, 89, 563, 66, 986, 427, 660, 518, 782, 339, 277, 682, 631, 329, 638, 858, 139, 475, 134, 710, 770, 276, 651, 888, 598, 570, 480, 78, 521, 706, 535, 91, 337, 381, 186, 999, 120, 404, 822, 64, 599, 757, 891, 225, 340, 160, 431, 852, 643, 346, 364, 620, 147, 94, 569, 116, 697, 953, 646, 556, 131, 582, 398, 967, 533, 856, 572, 514, 288, 239, 389, 253, 835, 99, 131, 674, 486, 182, 209, 768, 261, 460, 193, 701, 201, 548, 426, 436, 166, 236, 557, 724, 29, 233, 50, 74, 482, 129, 660, 662, 415, 927, 917, 246, 729, 695, 676, 492, 62, 841, 304, 321, 151, 689, 909, 581, 664, 441, 12, 992, 217, 961, 505, 871, 980, 98, 563, 605, 498, 878, 270, 608, 433, 716, 318, 196, 971, 898, 368, 391, 242, 127, 746, 631, 464, 103, 969, 41, 254, 571, 426, 764, 634, 484, 141, 266, 334, 307, 741, 578, 832, 176, 907, 331, 748, 500, 425, 389, 155, 222, 223, 956, 523, 369, 372, 169, 8, 608, 887, 319, 973, 381, 352, 522, 538, 448, 934, 717, 190, 605, 602, 879, 364, 115, 835, 364, 29, 357, 900, 239, 319, 193, 843, 335, 752, 441, 702, 395, 954, 158, 357, 322, 317, 580, 467, 418, 736, 771, 244, 378, 735, 444, 822, 378, 17, 730, 193, 16, 217, 935, 763, 64, 328, 33, 94, 161, 44, 54, 378, 280, 141, 678, 674, 99, 893, 430, 735, 30, 640, 326, 953, 979, 961, 408, 57, 701, 436, 713, 58, 437, 528, 301, 513, 284, 813, 195, 590, 341, 673, 357, 859, 396, 736, 317, 398, 24, 793, 246, 382, 199, 425, 137, 115, 546, 786, 262, 124, 517, 184, 598, 665, 85, 935, 21, 859, 525, 162, 831, 610, 691, 334, 645, 481, 521, 342, 942, 953, 327, 74, 1, 485, 409, 171, 244, 167, 388, 315, 998, 119, 142, 768, 944, 519, 708, 653, 560, 159, 780, 436, 90, 39, 683, 475, 317, 280, 730, 190, 734, 660, 487, 465, 190, 165, 334, 829, 892, 514, 304, 17, 874, 667, 562, 926, 38, 172, 738, 386, 91, 761, 688, 592, 44, 975, 765, 527, 926, 470, 931, 682, 688, 745, 798, 283, 135, 366, 674, 443, 317, 300, 342, 159, 166, 670, 981, 888, 100, 291, 462, 337, 826, 32, 369, 983, 434, 520, 381, 808, 850, 694, 821, 143, 462, 956, 409, 932, 128, 742, 268, 745, 315, 327, 283, 937, 289, 750, 965, 672, 220, 682, 294, 372, 819, 15, 539, 390, 389, 838, 819, 576, 204, 70, 798, 70, 479, 205, 538, 75, 714, 383, 392, 45, 215, 971, 238, 349, 493, 274, 372, 960, 146, 817, 83, 800, 959, 593, 897, 795, 732, 241, 220, 628, 351, 983, 56, 51, 550, 179, 838, 177, 287, 60, 104, 335, 718, 522, 437, 966, 92, 646, 11, 696, 455, 167, 962, 583, 297, 992, 807, 433, 916, 703, 853, 804, 156, 258, 634, 347, 574, 317, 485, 714, 15, 896, 611, 364, 373, 44, 817, 485, 956, 578, 236, 998, 553, 74, 288, 201, 712, 349, 727, 934, 773, 180, 604, 370, 558, 596, 477, 761, 133, 342, 667, 956, 174, 485, 290, 59, 417, 223, 209, 7, 50, 565, 41, 746, 921, 51, 373, 244, 889, 432

Max Heapify

N	Running time(ns)
10	3.20035003E+03
50	6.50065006E+03
100	9.10093009E+03
1000	8.2300953009E+04

Heap sort

N	Running time(ns)
10	5.20066005E+03
50	2.4500263003E+04
100	8.750082900871E+04
1000	2.78500261400254E+05