บทที่ 3

วิธีการดำเนินงานวิจัย

งานวิจัยนี้เสนอการนำเทคนิควิธีการผลต่างสืบเนื่องประยุกต์ใช้กับควิกซอร์ต ใช้ผลต่างสืบเนื่องช่วยหยุดการทำงานควิกซอร์ตเมื่อตรวจพบว่าชุดข้อมูลมีการเรียงลำดับถูกต้อง ทำให้ลดเวลาการทำงานและลดจำนวนครั้งการเรียกซ้ำที่เกิดขึ้นหลังจากชุดข้อมูลเรียงลำดับ โดยไม่ต้องทำงานต่อจนกระทั่งแบ่งข้อมูลไม่ได้ และยังสามารถแก้ปัญหาการเกิดกรณีแย่ที่สุดเมื่อนำข้อมูลที่เรียงลำดับหรือเกือบเรียงลำดับมาจัดเรียงด้วยควิกซอร์ต ในบทนี้จะอธิบายถึงรายละเอียดขั้นตอนของการทดลอง รูปแบบของการประยุกต์ใช้ผลต่างสืบเนื่องกับควิกซอร์ต ลักษณะของชุดข้อมูลที่นำมาใช้ในการทดลอง รูปแบบการวัดผลและประเมินผล โดยอธิบายแยกเป็น 2 ส่วน คือ องค์ประกอบที่ใช้ในการทดลอง และขั้นตอนในการทดลอง โดยมีดังต่อไปนี้

3.1 องค์ประกอบที่ใช้ในการทดลอง

งานวิจัยเรื่องควิกซอร์ตที่ใช้ผลต่างสืบเนื่อง ผู้วิจัยแยกองค์ประกอบที่ใช้ในการทดลอง โดยแยกออกเป็น 5 ส่วนดังต่อไปนี้

3.1.1 ขั้นตอนวิธีควิกซอร์ต

3.1.2 ผลต่างสืบเนื่อง

3.1.3 ข้อมูลใช้ในการทดลอง

3.1.4 รูปแบบการประยุกต์วิธีผลต่างสืบเนื่องเข้ากับควิกซอร์ต

3.1.5 เครื่องมือและภาษาโปรแกรมที่ใช้ในการทดลอง

3.1.1 ขั้นตอนวิธีควิกซอร์ต

ในงานวิจัยนี้เลือกใช้ควิกซอร์ตเพื่อการนำมาทดลองในการประยุกต์วิธีผลต่างสืบเนื่องกับควิกซอร์ต 2 แบบ เนื่องจากควิกซอร์ตตั้งแต่เริ่มนำเสนอจนถึงปัจจุบันได้มีการปรับปรุงและแก้ไขเพื่อปรับปรุงประสิทธิภาพการจัดเรียงทำให้เกิดควิกซอร์ตหลายเวอร์ชันที่แตกต่างกัน ดังนั้นงานวิจัยนี้จึงเลือกวิธีควิกซอร์ต 2 แบบ ที่นำมาใช้ในการทดลอง ดังนี้

3.1.1.1 **ขั้นตอนวิธีควิกซอร์ตที่เสนอไว้ใน [14]** เป็นควิกซอร์ตที่ใช้ข้อมูลตัวแรกเป็นตัวหลัก และใช้วิธีการตรวจและสลับข้อมูลที่ละด้าน ขั้นตอนการทำงานของควิกซอร์ตคือ เริ่มต้นเลือกตัวหลักจากข้อมูลตำแหน่งแรกนำไปเก็บในหน่วยความจำชั่วคราว จากนั้นในขั้นตอนการแบ่งข้อมูลทำการตรวจสอบข้อมูลโดยเริ่มจากด้านท้ายของชุดข้อมูลโดยใช้ตัวชี้ *(j)* ตรวจสอบข้อมูลย้อนมาทางด้านซ้ายจนกระทั่งพบข้อมูลตำแหน่งที่ค่าน้อยกว่าค่าตัวหลัก สลับข้อมูลตำแหน่งของ *j* แทนตำแหน่งตัวหลัก จากนั้นเปลี่ยนการตรวจสอบข้อมูลมาทางด้านหน้าโดยใช้ตัวชี้ *(i)* ตรวจสอบไปทางด้านขวาจนกระทั่งพบข้อมูลตำแหน่งที่ค่ามากกว่าหรือเท่ากับตัวหลัก สลับข้อมูลตำแหน่ง *i* แทนตำแหน่งตัวหลัก จากนั้นเปลี่ยนการตรวจสอบไปด้านท้ายต่อตำแหน่ง *j* ตรวจสอบข้อมูลเมื่อพบให้สลับข้อมูลกับตำแหน่งตัวหลัก จากนั้นเปลี่ยนมาตรวจสอบจากทางด้านหน้าต่อจากตำแหน่ง *i* ตรวจสอบข้อมูลเมื่อพบให้สลับข้อมูลกับตำแหน่งตัวหลัก ทำการตรวจสอบสลับไปมาจนกระทั่งตำแหน่งทางด้านหน้า *(i)* มีค่ามากกว่าตำแหน่งจากทางด้านท้าย *(j)* เมื่อค่าตำแหน่งตัวชี้ *i* มากกว่าตำแหน่งตัวชี้ *j* ให้สลับค่าตำแหน่งตัวหลักกับตำแหน่งตัวชี้ *i* เมื่อเสร็จขั้นตอนการแบ่งข้อมูล ทำตามขั้นตอนการทำงานของควิกซอร์ตต่อไป รายละเอียดของรหัสคำสั่งของควิกซอร์ตมีดังนี้

```
void Quicksort(int[] E, int first, int last){
        if(first < last){
                int a = E[first];
                int p = partition(a,first,last);
                E[p] = a;
                Quicksort(E, first, p-1);
                Quicksort(E, p+1, last);
        }
}
```

```
int partition(int[] E, int p_value, int first, int last){
        int low,high;
        low = first;
        high = last;
        while(low < high){
                int highVac = extendLargeRegion(E, p_value, low, high);
                int lowVac = extendSmallRegion (E, p_value, low+1, highVac);
                low = lowVac;
                high = highVac - 1;
        }
        return low;
}

int extendLargeRegion(int[] E, int p_value, int lowVac, int high){
        int highVac,curr;
        highVac = lowVac;
        curr = high;
        while(curr > lowVac){
                if(E[curr] < p_value){
                        E[low] = E[curr];
                        highVac = curr;
                        break;
                }
                curr--;
        }
        return highVac;
}
```

```
int extendSmallRegion (int[] E, int p_value, int low, int highVac){
    int lowVac,curr;
    lowVac = highVac;
    curr = low;
    while(curr < highVac){
        if(E[curr] >= p_value){
            E[highVac] = E[curr];
            lowVac = curr;
            break;
        }
        curr++;
    }
    return lowVac;
}
```

3.1.1.2 **ขั้นตอนวิธีควิกซอร์ตของ** Roger L. Wainwright [13] เสนอควิกซอร์ตที่ปรับปรุงการทำงานให้เหมาะกับข้อมูลแบบเกือบเรียงลำดับ ควิกซอร์ตแบบนี้ใช้เทคนิคการเลือกตัวหลักด้วยข้อมูลตำแหน่งกลาง และใช้การสลับข้อมูลในขั้นตอนการแบ่งข้อมูลทีละคู่ระหว่างตำแหน่งของตัวชี้ที่ใช้ในการตรวจข้อมูล โดยมีขั้นตอนการทำงาน คือ เลือกตัวหลักโดยใช้ข้อมูลที่อยู่ตำแหน่งกลาง ในขั้นตอนการแบ่งข้อมูลโดยให้ *i* เป็น ตัวชี้ด้านหน้า และ *j* เป็น ตัวชี้ด้านท้าย เริ่มต้นจะตรวจสอบข้อมูลจากทางด้านหน้าของชุดข้อมูลโดยใช้ตัวชี้ (*i*) ไปทางขวาจนพบข้อมูลที่ค่ามากกว่าค่าของตัวหลัก หยุดการตรวจทางด้านหน้าสลับไปตรวจสอบทางด้านหลังของชุดข้อมูลโดยใช้ตัวชี้ (*j*) ไปทางซ้ายจนพบข้อมูลที่ค่าน้อยกว่าค่าตัวหลัก ทำการสลับค่าข้อมูลระหว่างข้อมูลตำแหน่ง *i* กับตำแหน่ง *j* หลังจากสลับข้อมูลเสร็จ จะกลับมาตรวจสอบข้อมูลต่อโดยเริ่มจากด้านหน้าก่อนเสมอ ทำการตรวจสอบไปจนกระทั่งตำแหน่งของตัวชี้ *i* มีค่ามากกว่าตัวชี้ *j* จึงหยุดการตรวจสอบ นำค่าข้อมูลตำแหน่งตัวหลักแทนที่ข้อมูลตำแหน่งตัวชี้ *i* เมื่อเสร็จขั้นตอนการแบ่งข้อมูล ทำตามขั้นตอนการทำงานของควิกซอร์ตต่อไป รายละเอียดของรหัสคำสั่งของควิกซอร์ตมีดังนี้

```
void Quicksort(int[] E, int first, int last, int pivot_loc){
        boolean flag;
        int t,pivot;
        int i,j,size;
        if(first < last){
                pivot = E[pivot_loc];
                i = first-1;
                j = last+1;
                flag = true;
                while(flag){
                        i = i+1;
                        while(E[i] < pivot){
                                if(i > first)
                                i = i+1;
                        }
                        j = j-1;
                        while((E[j] >= pivot) && (j > first)){
                                if(j < last)
                                j = j-1;
                        }
                        if(i < j){
                                t = E[j];
                                E[j] = E[i];
                                E[i] = t;
                                if(i == pivot_loc)
                                        pivot_loc = j;
                        }else{
                                flag = false;
```

```
                } // of if(i < j)
            } // end of while flag loop
            t = E[i];
            E[i] = E[pivot_loc];
            E[pivot_loc] = t;
            i = i+1;
            size = j-first+1;
            if(size > 2)
                Quicksort(E, first, j, (first+j)/2);
            else
                if(size == 2)
                    if(E[first] > E[first+1]){
                        t = E[first];
                        E[first] = E[first+1];
                        E[first+1] = t;
                    }
            size = last-i+1;
            if(size > 2)
                Quicksort(E, i, last, (i+last)/2);
            else
                if(size ==2)
                    if(E[last] < E[last-1]){
                        t = E[last];
                        E[last] = E[last-1];
                        E[last-1] = t;
                    }
            } // end while
        } // of if(first<last)
}
```

3.1.2 วิธีผลต่างสืบเนื่อง

การหาค่าผลต่างสืบเนื่องของชุดข้อมูลสามารถหาได้จากสมการ

$$d(k_1,k_2,...,k_n)=\sum_{i=1}^{n-1}\left|x_{k_{i+1}}-x_{k_i}\right| \quad (1)$$

เมื่อ k_1, k_2, ..., k_n คือ ดรรชนีของข้อมูล และ x_i คือ ค่าข้อมูลที่ดรรชนี i และ n คือ จำนวนชุดข้อมูล

งานวิจัยนี้ทำการทดลองประยุกต์ใช้วิธีผลต่างสืบเนื่องกับควิกซอร์ต 2 แบบคือ แบบฟังก์ชันและแบบรวมกับขั้นตอนการแบ่งข้อมูล มีรายละเอียดดังนี้

3.1.2.1 แบบฟังก์ชัน เป็นการหาค่าผลต่างสืบเนื่องของชุดข้อมูลซึ่งสามารถคำนวณได้จากสมการที่ 1 โดยผู้วิจัยทำการสร้างฟังก์ชันการคำนวณค่าผลต่างสืบเนื่องแยกจากควิกซอร์ต เมื่อต้องการค่าผลต่างสืบเนื่องจึงทำการเรียกใช้ เพื่อลดความซับซ้อนในการใช้งาน โดยรายละเอียดของรหัสคำสั่งมีดังนี้

```
int successive_difference(int[] X, int first, int last){
    int sd_value = 0;
    for(int i = first; i < last ; i++){
            sd_value += X[i] > X[i+1]? X[i]-X[i+1]: X[i+1]-X[i];
    }
    return sd_value;
}
```

การทำงานของชุดคำสั่งจะทำการคำนวณค่าผลต่างสืบเนื่องในชุดข้อมูล X ตั้งแต่ลำดับที่ first ไปถึงตัวที่ last-1 และคืนค่าผลต่างสืบเนื่องที่คำนวณได้ เนื่องจากค่าผลต่างสืบเนื่องของชุดข้อมูลที่ไม่เรียงลำดับมากกว่าค่าผลต่างระหว่างค่าที่มากที่สุดกับค่าที่น้อยที่สุด ดังนั้นในการวิจัยนี้ได้ทำการปรับปรุงขั้นตอนการหาค่าผลต่างสืบเนื่องโดยเพิ่มการตรวจสอบผลต่างสืบเนื่องที่คำนวณได้ในแต่ละรอบเทียบกับค่าผลลบระหว่างค่าที่มากที่สุดกับค่าที่น้อยที่สุดของชุดข้อมูล โดยใช้สมมุติฐานที่ว่าถ้าในชุดข้อมูลที่เรียงลำดับแบบเรียงจากค่าน้อยไปหาค่ามาก ค่าที่มากที่สุดจะเป็นข้อมูลตัวสุดท้ายและค่าที่น้อยที่สุดจะเป็นข้อมูลตัวแรก ดังนั้นถ้าพบว่าเมื่อใดที่ผลต่างสืบเนื่องในระหว่างการคำนวณสูงกว่าค่าผลลบระหว่างข้อมูลตัวสุดท้ายกับข้อมูลตัวแรก

ให้หยุดการคำนวณค่าผลต่างสืบเนื่องในรอบนั้นโดยไม่ต้องคำนวณให้ครบจำนวนข้อมูล ตัวอย่างชุดข้อมูล k = 8, -3, 5, 0, 7

ข้อมูลตัวสุดท้าย – ข้อมูลตัวแรก ของข้อมูล k(1,2,3,4,5) = 7 – 8 = -1

ผลต่างสืบเนื่องของ k(1,2,3,4,5) รอบแรก คู่ลำดับที่ (1,2) = |8 - -3| = 11

จะเห็นว่าในรอบแรกของการคำนวณ ค่าผลต่างสืบเนื่องสูงกว่าผลต่างระหว่างข้อมูลตัวสุดท้ายกับข้อมูลตัวแรกซึ่งแสดงว่าชุดข้อมูลไม่เรียงลำดับ จึงทำให้สามารถหยุดการหาค่าผลต่างสืบเนื่องได้ทันทีโดยไม่ต้องเสียเวลาคำนวณจนครบทุกข้อมูล ดังนั้นรหัสคำสั่งการหาผลต่างสืบเนื่องใหม่ เป็นดังนี้

```
int successive_difference(int[] X, int first, int last){
    int sd_value = 0;
    int max_min = X[last] – X[first];
    for(int i = first; i < last ; i++){
        if(sd_value > max_min) break;
        sd_value += X[i] > X[i+1]? X[i]-X[i+1]: X[i+1]-X[i];
        ...
}
```

3.1.2.2 แบบรวมอยู่ในขั้นตอนการแบ่งข้อมูล เป็นการเพิ่มขั้นตอนการหาผลต่างสืบเนื่องเข้าไปในขั้นตอนการแบ่งข้อมูลเนื่องจากในขั้นตอนการแบ่งข้อมูลของควิกซอร์ตต้องมีการตรวจสอบข้อมูลทั้งหมดอยู่แล้ว ดังนั้นในขั้นตอนการแบ่งข้อมูลจึงสามารถเพิ่มขั้นตอนการคำนวณหาค่าผลต่างสืบเนื่องได้และเมื่อเสร็จสิ้นขั้นตอนการแบ่งข้อมูลจะได้ค่าผลต่างสืบเนื่องของชุดข้อมูล โดยรหัสคำสั่งที่ใช้กับควิกซอร์ตแบบนี้แสดงในภาคผนวก

3.1.3 ข้อมูลที่ใช้ในการทดลอง

ข้อมูลที่นำมาใช้ในการทดลองแบ่งเป็น 2 กลุ่ม คือ กลุ่มข้อมูลที่ใช้เป็นกรณีศึกษาตามลักษณะความแตกต่างกันของค่าข้อมูลภายในชุดข้อมูล ประกอบด้วยข้อมูลแบบสุ่มและข้อมูลแบบเรียงลำดับ และกลุ่มข้อมูลที่เป็นข้อมูลแบบกำหนดสัดส่วนสมบัติข้อมูล โดยข้อมูลแต่ละแบบมีรายละเอียดดังนี้

3.1.3.1 ข้อมูลสุ่มและข้อมูลเรียงลำดับ กลุ่มข้อมูลนี้เป็นข้อมูลที่จำลองขึ้นเพื่อเป็นกรณีศึกษาการทำงานของ ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องตามลักษณะความแตกต่างกันของค่าข้อมูลภายในชุดข้อมูล ในการวิจัยนี้แบ่งออกเป็น 2 ส่วน โดยข้อมูลทั้งหมดจะเป็นข้อมูลที่สร้างขึ้นโดยใช้ฟังก์ชัน Random() ของภาษาจาวาในการสร้างข้อมูล โดยข้อมูลในกลุ่มนี้แบ่งเป็น 2 ประเภท ดังนี้

1. ข้อมูลสุ่ม เป็นข้อมูลที่สร้างขึ้นโดยใช้ฟังก์ชัน Random() เพื่อสร้างข้อมูลตามลักษณะโดเมนที่นำมาทดลองแต่ในการวิจัยนี้จำกัดชนิดข้อมูลเป็นจำนวนเต็มเท่านั้น ดังนั้นก่อนนำข้อมูลไปใช้งานจึงต้องทำการแปลงให้เป็นจำนวนเต็ม ข้อมูลในกลุ่มนี้เป็นข้อมูลที่มุ่งสนใจในโดเมนของข้อมูลที่จะใช้ทดสอบการทำงานของควิกซอร์ตที่ใช้ผลต่างสืบเนื่อง ในงานวิจัยนี้จึงทำการสร้างข้อมูลสุ่มขึ้นมา 4 แบบ ดังนี้

- ข้อมูลสุ่ม เป็นตัวอย่างของข้อมูลที่มีจำนวนของค่าที่ต่างกันสูงโอกาสมีข้อมูลค่าซ้ำกันน้อย ช่วงข้อมูลนี้ คือ 0 – 100000 ซึ่งมีสัดส่วนการซ้ำ > 1
- ข้อมูลสุ่มจำกัดช่วงขนาดข้อมูล เป็นตัวอย่างข้อมูลที่มีจำนวนค่าที่ต่างกันสูงแต่มีโอกาสที่มีค่าซ้ำกันในชุดข้อมูลสูงกว่าข้อมูลแบบแรก โดยช่วงข้อมูลของแบบนี้จะแปรผันตามขนาดของชุดข้อมูลที่ใช้ในการทดลอง ซึ่งมีสัดส่วนการซ้ำ = 1
- ข้อมูลสุ่มจำกัดช่วงเกรด เป็นตัวอย่างของข้อมูลที่มีค่าที่ต่างกันในชุดข้อมูลต่ำ โดเมนที่สนใจ คือ คะแนนเกรดของนักเรียน นักศึกษา ซึ่งข้อมูลแบบนี้มีโอกาสที่ต้องใช้การจัดเรียงข้อมูลเพื่อการนำไปใช้ประโยชน์ โดยช่วงข้อมูลของข้อมูลนี้ คือ 0-4 ซึ่งมีสัดส่วนการซ้ำระหว่าง 0.005 ถึง 0.05
- ข้อมูลสุ่มจำกัดช่วงอายุ เป็นตัวอย่างของข้อมูลที่มีค่าต่างกันในชุดข้อมูลต่ำแต่โอกาสที่มีค่าซ้ำกันต่ำกว่าข้อมูลสุ่มจำกัดช่วงเกรด โดยข้อมูลในโดเมน คือ อายุของคน ซึ่งจะใช้ในงานที่เกี่ยวกับสถิติที่เกี่ยวกับประชากร โดยช่วงข้อมูลของข้อมูลนี้ คือ 0-99 ซึ่งมีสัดส่วนการซ้ำระหว่าง 0.1 ถึง 1

2. ข้อมูลเรียงลำดับ เป็นข้อมูลเพื่อใช้ทดลองกรณีที่แย่ที่สุดกับควิกซอร์ตที่ใช้ผลต่างสืบเนื่อง โดยใช้ข้อมูลสุ่มมาทำการจัดเรียงก่อนใช้ทดลอง เพื่อดูประสิทธิภาพการทำงานเทียบกันระหว่างควิกซอร์ตกับควิกซอร์ตที่ใช้ผลต่างสืบเนื่อง

3.1.3.2 ข้อมูลกำหนดสัดส่วนสมบัติข้อมูล ข้อมูลกลุ่มนี้เป็นข้อมูลที่สร้างขึ้นเพื่อศึกษาลักษณะข้อมูลที่มีผลต่อประสิทธิภาพการทำงานของควิกซอร์ต โดยใช้การกำหนดสัดส่วนลักษณะสมบัติภายในของชุดข้อมูล ในงานวิจัยนี้กำหนดสมบัติของข้อมูล 2 แบบ คือ แบบกำหนดสัดส่วนการเรียงลำดับ และแบบกำหนดสัดส่วนการซ้ำ

1. แบบกำหนดสัดส่วนการเรียงลำดับ เป็นข้อมูลที่สร้างเพื่อให้สอดคล้องกับงานวิจัย [3, 12, 13] ซึ่งเป็นกลุ่มที่ศึกษาควิกซอร์ตกับข้อมูลที่เกือบเรียงลำดับ โดยนำข้อมูลที่เรียงลำดับแล้วมาสุ่มสลับตำแหน่งข้อมูลพื่อให้เกิดชุดข้อมูลที่เกือบเรียงลำดับ โดยจำนวนข้อมูลที่สลับจะเป็นไปตามสัดส่วนที่ต้องการ สัดส่วนการเรียงลำดับสามารถหาได้จากสมการ Sortedness ratio = *k/N* เมื่อ *k* คือ จำนวนข้อมูลที่เรียงลำดับไม่ถูกต้อง *N* คือ จำนวนข้อมูลทั้งหมด เช่น ข้อมูลขนาด 1000 มีข้อมูลที่มีลำดับไม่ถูกต้องจำนวน 10 สัดส่วนของการเรียงลำดับ คือ 10/1000 = 0.01 หรือ 1% ในทางกลับกันถ้าต้องการสร้างข้อมูลที่มีสัดส่วนการเรียง 0.1 หรือ 10% จำนวนข้อมูลที่ต้องสลับเพื่อทำให้ข้อมูลอยู่ผิดตำแหน่งสามารถหาได้จากสมการ *k* = Sortedness ratio * *N* ดังนั้นจำนวน *k* ที่ต้องการคือ (0.1)*1000 = 10 ดังนั้นต้องทำการสลับข้อมูล 10 ตัว แล้วจึงนำข้อมูลที่เหลือใส่คืนในส่วนที่ว่างอยู่ สัดส่วนการเรียงข้อมูลที่ใช้ในการทดลองในงานวิจัยนี้ คือ 2%, 4%, 6%, 8%, 10%, 15%, 20%, 25% และ 30%

2. แบบกำหนดสัดส่วนการซ้ำ เป็นข้อมูลสร้างขึ้นเพื่อทดสอบหาสัดส่วนของจำนวนค่าที่มีในชุดข้อมูลที่หมาะสมสำหรับการนำควิกซอร์ตที่ใช้ผลต่างสืบเนื่องไปใช้งาน โดยข้อมูลที่สร้างขึ้นจะกำหนดจำนวนค่าข้อมูลที่เกิดขึ้นในชุดข้อมูลตามสัดส่วนของการซ้ำที่ใช้ทดลองเพื่อทดสอบหาค่าความหนาแน่นข้อมูลที่ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องทำงานได้ดีกว่า ควิกซอร์ตปกติ สัดส่วนการซ้ำข้อมูลหาได้จากสมการ Repeatedness ratio = *d/N* เมื่อ *d* คือจำนวนค่าข้อมูลที่ต่างกันในชุดข้อมูล(distinct) *N* คือ ข้อมูลทั้งหมด เช่น ข้อมูลขนาด 1000 มีจำนวนข้อมูลที่ต่างกัน 100 ค่า สัดส่วนการซ้ำเท่ากับ 100/1000 = 0.1 หรือการหาค่าสัดส่วนการซ้ำข้อมูลเกรดนักเรียน จำนวนค่าที่ต่างกันในข้อมูลคะแนนเกรดที่เป็นตัวเลข คือ 5 (0, 1, 2, 3, 4) จำนวนข้อมูล 100 สัดส่วนการซ้ำเท่ากับ 5/100 = 0.05 หรือ 5% ในทางกลับกันถ้าต้องการสร้างข้อมูลที่มีสัดส่วนการซ้ำ 0.01 หรือ 1% ในชุดข้อมูล 1000 สามารถหาจำนวนค่าข้อมูลที่ต่างกันที่สามารถมีได้ในชุดข้อมูล จำนวนข้อมูลเท่ากับ 0.01 * 1000 = 10 ค่า ดังนั้นเมื่อสร้างข้อมูลจะจำกัดค่าข้อมูลที่จะสุ่มขึ้นมาเพียง 10 ค่าเท่านั้น สัดส่วนการซ้ำที่ใช้ในการทดลองของงานวิจัยนี้ คือ 0.001 - 1 โดยช่วง 0.001 - 0.01 มีระยะห่างช่วงละ 0.001, ช่วง 0.01 - 0.1 มีระยะห่างช่วงละ 0.01, ช่วง 0.1 - 0.6 มีระยะห่างช่วงละ 0.05 และช่วง 0.6 - 1 มีระยะห่างช่วงละ 0.1

3.1.4 รูปแบบการประยุกต์วิธีผลต่างสืบเนื่องกับควิกซอร์ต

รูปแบบของการประยุกต์วิธีการผลต่างสืบเนื่องเข้ากับควิกซอร์ตในระหว่างขั้นตอนต่างๆ เพื่อหาความแตกต่างของควิกซอร์ตเมื่อเพิ่มผลต่างสืบเนื่องเข้าไปในการทำงาน ในงานวิจัยนี้ทดลองประยุกต์ใช้วิธีผลต่างสืบเนื่องกับควิกซอร์ต 3 แบบ คือ

1. เรียกใช้ฟังก์ชันผลต่างสืบเนื่องก่อนขั้นตอนการแบ่งข้อมูล โดยจะเป็นการเรียกใช้ฟังก์ชันผลต่างสืบเนื่องเพื่อตรวจสอบการเรียงลำดับข้อมูลก่อนการทำงานของควิกซอร์ต ฟังก์ชันจะคืนค่าผลต่างของชุดข้อมูล นำค่าที่ได้ตรวจสอบผลลบระหว่างค่าสุดท้าย (last) กับค่าแรก (first) ถ้าข้อมูลเรียงลำดับให้จบการทำงาน แต่ถ้าข้อมูลยังไม่เรียงลำดับ จะเข้ากระบวนการทำงานของควิกซอร์ต รหัสคำสั่งดังนี้

```
void Quicksort(int[] E, int first, int last){
    if(first < last){
        if((E[last] – E[first]) == SuccessiveDifference(E, first, last)) return;
        int a = E[first];
        int p = partition(a,first,last);
        E[p] = a;
        Quicksort(E, first, p-1);
        Quicksort(E, p+1, last);
    }
}
```

2. เรียกใช้ฟังก์ชันผลต่างสืบเนื่องหลังการแบ่งข้อมูล คล้ายกับแบบแรก คือ ใช้การเรียกฟังก์ชันผลต่างสืบเนื่องตรวจสอบข้อมูลหลังขั้นตอนการแบ่งข้อมูลของควิกซอร์ต เนื่องจากในขั้นตอนการแบ่งข้อมูลมีการสลับตำแหน่งข้อมูล ทำให้มีโอกาสเกิดข้อมูลเรียงลำดับหลังการแบ่งกลุ่มได้ แล้วทำการตรวจค่าผลต่างสืบเนื่องกับผลลบระหว่างค่าสุดท้ายกับค่าแรก รหัสคำสั่งดังนี้

```
void Quicksort(int[] E, int first, int last){
    if(first < last){
        int a = E[first];
        int p = partition(a,first,last);
        E[p] = a;
        if((E[last] – E[first]) == SuccessiveDifference(E, first, last)) return;
```

```
            Quicksort(E, first, p-1);
            Quicksort(E, p+1, last);
        }
    }
```

3. รวมวิธีการผลต่างสืบเนื่องรวมในขั้นตอนการแบ่งข้อมูล เนื่องจากในขั้นตอนการแบ่งข้อมูล ควิกซอร์ตจะมีการตรวจสอบข้อมูลทั้งหมดดังนั้นจึงสามารถเพิ่มขั้นตอนการคำนวณค่าผลต่างสืบเนื่อง เมื่อเสร็จขั้นตอนการแบ่งข้อมูลทำการตรวจค่าผลต่างสืบเนื่องกับผลลบระหว่างค่าสุดท้ายกับค่าแรก รหัสคำสั่งดังนี้

```
int successive_difference_value = 0;
void Quicksort(int[] E, int first, int last){
    if(first < last){
            int a = E[first];
            int p = partition(a,first,last);
            E[p] = a;
            if((E[last] – E[first]) == successive_difference_value) return;
            Quicksort(E, first, p-1);
            Quicksort(E, p+1, last);
     }
}
int extendLargeRegion(int[] E, int p_value, int lowVac, int high){
    int highVac,curr;
    highVac = lowVac;
    curr = high;
    while(curr > lowVac){
            if(E[curr] < p_value){
                    E[low] = E[curr];
                    highVac = curr;
                    break;
            }
            Successive_difference_value += Math.abs(E[curr] – E[curr+1]);
```

```
                curr--;
        }
        return highVac;
    }
    int extendSmallRegion (int[] E, int p_value, int low, int highVac){
        int lowVac,curr;
        lowVac = highVac;
        curr = low;
        while(curr < highVac){
                if(E[curr] >= p_value){
                        E[highVac] = E[curr];
                        lowVac = curr;
                        break;
                }
                Successive_difference_value += Math.abs(E[curr] – E[curr-1]);
                curr++;
        }
        return lowVac;
    }
```

3.1.5 เครื่องมือและภาษาโปรแกรมที่ใช้ในการทดลอง

ในส่วนของเครื่องมือและภาษาโปรแกรมที่ใช้ในการพัฒนาและวัดผล การทำงานของควิกซอร์ตแบบต่างๆ มีดังนี้

ตารางที่ 3.1

แสดงรายละเอียดของเครื่องมือและซอฟต์แวร์ที่ใช้ในการทดลอง

ฮาร์ดแวร์	PC AMD Turion64 2.0GHz, RAM 1 Gbytes
ระบบปฏิบัติการ	Windows XP Professional
ซอฟต์แวร์ภาษาที่ใช้ในการเขียนโปรแกรม	Java 1.5

3.2 ขั้นตอนการทดลอง

งานวิจัยนี้เสนอการประยุกต์วิธีการผลต่างสืบเนื่องกับควิกซอร์ต ในหัวข้อนี้เป็นการอธิบายรายละเอียดของขั้นตอนการทดลอง ในงานวิจัยนี้แบ่งขั้นตอนการทดลองเป็น 2 ส่วนหลัก คือ ขั้นตอนการเตรียมข้อมูลและควิกซอร์ต และขั้นตอนการทดลองและบันทึกผล ซึ่งรายละเอียดของแต่ละขั้นตอนมีดังต่อไปนี้

3.2.1 ขั้นตอนการเตรียมข้อมูลและควิกซอร์ต

3.2.1.1 ขั้นตอนการเตรียมข้อมูล ดังที่ได้กล่าวไว้ในหัวข้อที่ 3.1.3 ว่าลักษณะข้อมูลที่นำมาใช้ประกอบการทดลองประกอบด้วยข้อมูล 2 แบบ คือ แบบแรกเป็นข้อมูลแบบสุ่มและข้อมูลเรียงลำดับ แบบที่สองเป็นข้อมูลแบบกำหนดสัดส่วนข้อมูล ซึ่งสามารถอธิบายลักษณะของข้อมูลได้ดังนี้

1. ข้อมูลแบบสุ่ม ข้อมูลที่สร้างขึ้นใช้ในการทดลองแบ่งได้ 2 แบบ คือ ข้อมูลแบบสุ่ม และข้อมูลสุ่มจำกัดช่วงตามโดเมนของข้อมูลที่ใช้เป็นกรณีศึกษา เนื่องด้วยงานวิจัยนี้จำกัดลักษณะชนิดข้อมูลเป็นจำนวนเต็มเท่านั้น แต่ผลที่ได้จากฟังก์ชัน Random() เป็นค่าจำนวนจริง ดังนั้นจึงต้องทำการแปลงให้เป็นจำนวนเต็มก่อนนำไปใช้งาน การสร้างข้อมูลสุ่มที่ใช้ในการทดลองใช้ผลที่ได้จากฟังก์ชัน Random() ในภาษาจาวา และใช้เทคนิคการเปลี่ยนค่า seed ของการสุ่มตามรอบการทำงานเพื่อให้ทุกรอบการทดลองได้ข้อมูลไม่ซ้ำกัน สำหรับข้อมูลสุ่มเป็นการสร้างข้อมูลที่มีลักษณะการซ้ำกันของข้อมูลในชุดข้อมูลต่ำ ดังนั้นจึงทำการคูณด้วย 100,000 เพื่อให้ข้อมูลที่ได้กระจายตัวและมีการซ้ำของข้อมูลต่ำ ในส่วนของข้อมูลแบบสุ่มจำกัดช่วง เป็นข้อมูลที่สร้างขึ้นโดยใช้ฟังก์ชัน Random() และกำหนดค่า seed ตามรอบการทดลองเช่นกัน แต่ปรับลดตัวคูณเพื่อสร้างข้อมูลที่มีลักษณะการซ้ำในชุดข้อมูลที่ต่างๆ กัน โดยในการทดลองนี้ใช้ช่วงข้อมูล 3 แบบ คือ แบบแรกช่วงคะแนนเกรด(0-4) เป็นกรณีศึกษาของข้อมูลที่มีลักษณะความต่างกันของค่าภายในชุดข้อมูลต่ำคือมีค่าที่แตกต่างกันแค่ 5 ค่า ทำให้ข้อมูลที่มีโอกาสมีข้อมูลซ้ำๆ สูงในข้อมูลปริมาณมาก แบบที่สองช่วงอายุประชากร(0-100) เป็นกรณีศึกษาอีกแบบของข้อมูลที่มีโอกาสมีข้อมูลซ้ำในชุดข้อมูลสูงแต่จะน้อยกว่าแบบแรกเพราะมีค่าข้อมูลที่เป็นไปได้ 100 ค่า และแบบที่สามกำหนดช่วงขนาดข้อมูล (0-N) เป็นข้อมูลที่สร้างขึ้นเพื่อเป็นกรณีศึกษาของข้อมูลที่มีลักษณะข้อมูลซ้ำน้อยๆ เช่น ข้อมูลขนาด 500 จะสุ่มค่าช่วงข้อมูลไม่เกิน 500 เป็นต้น

ข้อมูลแบบเรียงลำดับ เป็นข้อมูลที่นำข้อมูลสุ่มมาจัดเรียงก่อน จึงนำมาใช้ในการทดลองเพื่อทดลองนำข้อมูลกรณีแย่ที่สุดมาจัดเรียง

2. ข้อมูลแบบกำหนดสัดส่วนการเรียงและข้อมูลแบบกำหนดสัดส่วนการซ้ำ

ข้อมูลแบบกำหนดสัดส่วนการเรียงลำดับ เป็นข้อมูลที่ใช้ในการทดลองของงานวิจัย [3, 12, 13] ดังนั้นในงานวิจัยนี้จึงนำข้อมูลแบบนี้มาใช้ทดลองกับควิกซอร์ตที่ใช้ผลต่างสืบเนื่องเพื่อเทียบกับ Qsort การสร้างข้อมูลแบบกำหนดสัดส่วนการจัดเรียงลำดับทำได้โดย นำข้อมูลที่เรียงลำดับมาทำการสุ่มสลับตำแหน่งข้อมูลจำนวน *k* โดยจำนวนการสลับค่าคำนวณได้จาก *k* = Sortedness ratio * *N* เมื่อหาจำนวนข้อมูลที่ต้องสลับตำแหน่งได้แล้ว ทำการสร้างชุดข้อมูลเปล่าขนาดเท่ากับชุดข้อมูลเดิม จากนั้นทำการสุ่มตำแหน่งที่ต้องการย้ายและทำการสุ่มตำแหน่งที่ต้องย้ายไป ทำการย้ายข้อมูลจากชุดข้อมูลเดิมไปยังชุดข้อมูลเปล่าตามตำแหน่งที่สุ่มได้ทำการย้ายไปจนกระทั่งครบจำนวน จากนั้นให้ทำการย้ายข้อมูลที่เหลือในชุดข้อมูลเดิมไปยังชุดข้อมูลใหม่ตั้งแต่ต้นจนจบชุดข้อมูลแทนส่วนที่ยังว่างตามลำดับจนหมด ชุดข้อมูลใหม่ที่ได้เป็นข้อมูลที่มีสัดส่วนการเรียงตามที่เราต้องการ เช่น ต้องการข้อมูลที่มีสัดส่วนการเรียง 0.02 โดยชุดข้อมูลมีขนาด 1000 ดังนั้นจำนวนข้อมูลที่ต้องทำการสลับตำแหน่งเท่ากับ 0.02 * 1000 = 20 ตัว โดยข้อมูลแบบกำหนดสัดส่วนการเรียงที่ใช้ในการทดลองในงานวิจัยมีดังนี้ 0.02, 0.04, 0.06, 0.08, 0.1, 0.15, 0.2, 0.25 และ 0.3 ตามลำดับ

ข้อมูลแบบกำหนดสัดส่วนการซ้ำ เป็นข้อมูลที่สร้างทดลองหาลักษณะข้อมูลที่ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องทำงานได้ดีกว่าควิกซอร์ตปกติ เนื่องจากในระหว่างการทดลองผู้วิจัยพบว่าข้อมูลที่มีสัดส่วนการซ้ำต่ำ(มีข้อมูลซ้ำมากๆ) ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องทำงานได้ดีกว่าควิกซอร์ตปกติ ข้อมูลแบบนี้เป็นข้อมูลที่กำหนดจำนวนค่าข้อมูลที่มีได้ในชุดข้อมูล เพื่อทดลองหาความหนาแน่นข้อมูลที่เหมาะสมการนำควิกซอร์ตที่ใช้วิธีผลต่างสืบเนื่องไปใช้งาน ขั้นตอนการสร้างข้อมูลแบบกำหนดสัดส่วนการซ้ำ มีดังนี้ เริ่มต้นให้ทำการหาจำนวนความต่างของค่าข้อมูลที่จะมีได้ในชุดข้อมูล โดยหาจากสมการ *d* = Repeatedness ratio * *N* เช่นต้องการได้สัดส่วนของการซ้ำที่ 0.01 หรือ 1% ของข้อมูลขนาด 1000 จำนวนความแตกต่างของจำนวนข้อมูลที่มีได้เท่ากับ *d* = 0.01 * 1000 = 10 จากนั้นในทำการสุ่มข้อมูลขึ้นมาและเก็บค่าข้อมูลที่สุ่มขึ้นมาเพื่อการตรวจสอบค่าที่สุ่มได้ว่าครบจำนวนหรือยัง ถ้ายังไม่ครบให้สุ่มใหม่ไปจนกว่าจะได้ข้อมูลที่ค่าต่างกันครบจำนวน โดยไม่นับรวมค่าที่ซ้ำกับค่าที่เคยสุ่มขึ้นมาแล้ว ถ้าได้ค่าข้อมูลครบแล้ว ข้อมูลที่สุ่มใหม่ต้องทำการเปรียบเทียบกับระเบียนของข้อมูลเดิม ถ้าซ้ำจึงจะนำมาใช้ ถ้าไม่ซ้ำให้โยนค่านั้นทิ้งแล้วสุ่มใหม่ สุ่มจนกว่าจะได้ข้อมูลครบจำนวนข้อมูลในชุดข้อมูล ข้อมูลที่ได้จะเป็นข้อมูลที่มีค่าข้อมูลที่แตกต่างกันตามที่ต้องการ เช่น ต้องการข้อมูลที่มีสัดส่วนการซ้ำ 0.01 ชุดข้อมูลขนาด 1000 ดังนั้นชุดข้อมูลนี้จะมีข้อมูล 1000 ตัว แต่มีข้อมูลที่มีค่าที่ต่างกัน 10 ค่า ในงานวิจัยนี้สร้าง

ชุดข้อมูลที่กำหนดสัดส่วนการซ้ำ ดังนี้ 0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.7, 0.8, 0.9 , 1 ตามลำดับ

3.2.1.2 **ขั้นตอนการสร้างควิกซอร์ต** ควิกซอร์ตที่นำมาใช้ในงานวิจัย แบ่งเป็น 2 แบบคือ ควิกซอร์ตจาก [14] และควิกซอร์ตของ Roger L. Wainwright งานวิจัย [13] รายละเอียดของรูปแบบการสร้างควิกซอร์ตเพื่อการทดลองแต่ละแบบ มีดังนี้

1. ควิกซอร์ตจาก [14] งานวิจัยนี้ทดลองสร้างควิกซอร์ตแบบนี้ 4 ลักษณะ คือ ควิกซอร์ตปกติ(Quicksort) ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องแบบรวมในขั้นตอนการแบ่งข้อมูล (SDQuicksort) ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องแบบฟังก์ชันเรียกใช้ก่อนการแบ่งข้อมูล (SDQuicksort(BP)) ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องแบบฟังก์ชันเรียกใช้หลังการแบ่งข้อมูล (SDQuicksort(AP))

2. ควิกซอร์ตจากงานวิจัย[13] งานวิจัยนี้ทดลองสร้างควิกซอร์ตแบบนี้ 5 ลักษณะ คือ ควิกซอร์ตปกติ(Quicksort) ควิกซอร์ตแบบใช้การเปรียบเทียบ [13](Qsort) ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องแบบรวมในขั้นตอนการแบ่งข้อมูล(SDQuicksort) ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องแบบฟังก์ชันเรียกใช้ก่อนการแบ่งข้อมูล(SDQuicksort(BP)) ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องแบบฟังก์ชันเรียกใช้หลังการแบ่งข้อมูล(SDQuicksort(AP))

ควิกซอร์ตทุกแบบเขียนด้วยภาษาจาวา โดยรหัสคำสั่งอยู่ในภาคผนวก

3.2.2 ขั้นตอนการทดลองและบันทึกผล

ขั้นตอนการทดลองเป็นการนำควิกซอร์ตทำการจัดเรียงข้อมูลแบบต่างๆ ในงานวิจัยนี้ใช้การบันทึกเวลาการทำงานและจำนวนครั้งการเรียกซ้ำเป็นตัววัดประสิทธิภาพการทำงานของควิกซอร์ตแต่ละแบบ การทดลองจัดเรียงข้อมูลแต่ละแบบจะทำการจัดเรียงข้อมูลจำนวน 50 ครั้ง จากนั้นนำเวลาและจำนวนการเรียกซ้ำ มาหาค่าเฉลี่ยของเวลาการทำงานและจำนวนการเรียกซ้ำของแต่ละข้อมูล ผลการทดลองบันทึกลงในตารางเก็บข้อมูลและแสดงผลในรูปแบบกราฟเส้นแสดงเวลาการทำงานและจำนวนครั้ง โดยเทียบกับขนาดข้อมูล