ภาคผนวก

รหัสคำสั่งภาษาจาวาของควิกซอร์ต

รหัสคำสั่งภาษาจาวาของควิกซอร์ตจาก [14]

ควิกซอร์ต

```
private void Quicksort(int first,int last){
  recursive_call++;
  if(first < last){
    int a = DATA[first];
    int p = partition(a,first,last);
    DATA[p] = a;
    Quicksort(first,p-1);
    Quicksort(p+1,last);
  }
}
private int partition(int p_value,int first,int last){
  int low,high;
  low = first;
  high = last;
  while(low < high){
    int highVac = highRegion(p_value,low,high);
    int lowVac = lowRegion(p_value,low+1,highVac);
    low = lowVac;
    high = highVac - 1;
  }
  return low;
}
```

```
private int highRegion(int p_value,int low,int high){
    int highVac,curr;
   highVac = low;
   curr = high;
   while(curr > low){
     if(DATA[curr] < p_value){
       DATA[low] = DATA[curr];
       highVac = curr;
       break;
     }
     curr--;
   }
   return highVac;
}

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

ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องแบบรวมในขั้นตอนการแบ่งข้อมูล

```
private void Quicksort(int first,int last){
   recursive_call++;
   successive_value = 0;
   if(first < last){
      int a = DATA[first];
      int p = partition(a,first,last);
      DATA[p] = a;
      if(p > first && successive_value <= (DATA[last] - DATA[first]))
         successive_value += a - DATA[p-1];
      if (p < last && successive_value <= (DATA[last] - DATA[first]))
         successive_value += DATA[p+1] - a;
      if((DATA[last] - DATA[first]) == successive_value)  return;
      Quicksort(first,p-1);
      Quicksort(p+1,last);
   }
}
private int partition(int p_value,int first,int last){
   int low,high;
   low = first;
   high = last;
   while(low < high){
      int highVac = highRegion(p_value,low,high,first,last);
      int lowVac = lowRegion(p_value,low+1,highVac,first,last);
      low = lowVac;
      high = highVac - 1;
   }
   return low;
}
```

```
private int highRegion(int p_value,int low,int high,int first,int last){
  int highVac,curr;
  highVac = low;
  curr = high;
  while(curr > low){
    if(DATA[curr] < p_value){
      DATA[low] = DATA[curr];
      DATA[curr] = p_value;
      highVac = curr;
      if(low > first && successive_value <= (DATA[last] - DATA[first])){
        successive_value+=DATA[low]>DATA[low-1]?DATA[low]-DATA[low-
        1]:DATA[low-1]-DATA[low];
      }
      break;
    }
    if(curr < last && successive_value <= (DATA[last] - DATA[first])){
      successive_value+=DATA[curr]>DATA[curr+1]?DATA[curr]-
      DATA[curr+1]:DATA[curr+1]-DATA[curr];
    }
    curr--;
  }
  return highVac;
}

private int lowRegion(int p_value,int low,int highVac,int first,int last){
  int lowVac,curr;
  lowVac = highVac;
  curr = low;
  while(curr < highVac){
```

```
    if(DATA[curr] >= p_value){
      DATA[highVac] = DATA[curr];
      DATA[curr] = p_value;
      lowVac = curr;
      if(highVac < last && successive_value <= (DATA[last] - DATA[first])){
        successive_value+=DATA[highVac]>DATA[highVac+1]?
        DATA[highVac]-DATA[highVac+1]:DATA[highVac+1]-DATA[highVac];
      }
      break;
    }
    if(curr > first && successive_value <= (DATA[last] - DATA[first])){
      successive_value+=DATA[curr]>DATA[curr-1]?DATA[curr]-DATA[curr-
      1]:DATA[curr-1]-DATA[curr];
    }
    curr++;
  }
  return lowVac;
}
```

ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องแบบเรียกใช้ก่อนขั้นตอนการแบ่งข้อมูล

```
private void Quicksort(int first,int last){
  recursive_call++;
  if(first < last){
    if((DATA[last] - DATA[first]) == successive_difference(first,last)) return;
    int a = DATA[first];
    int p = partition(a,first,last);
    DATA[p] = a;
    Quicksort(first,p-1);
```

```
      Quicksort(p+1,last);
    }
  }

  private int partition(int p_value,int first,int last){
    .... เหมือนควิกซอร์ต
  }

  private int highRegion(int p_value,int low,int high){
    .... เหมือนควิกซอร์ต

  }

  private int lowRegion(int p_value,int low,int highVac){
    .... เหมือนควิกซอร์ต
  }


  public int successive_difference(int start , int stop){
    int SS = 0;
    int SD_min_max = DATA[stop] - DATA[start];
    for(int i = start; i < stop ; i++){
      if(SS > SD_min_max) break;
         SS += DATA[i] > DATA[i+1]?DATA[i]-DATA[i+1]:DATA[i+1]-DATA[i];
    }
    return SS;
  }
```

ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องแบบเรียกใช้หลังขั้นตอนการแบ่งข้อมูล

```
private void Quicksort(int first,int last){
   recursive_call++;
   if(first < last){
      int a = DATA[first];
     int p = partition(a,first,last);
      DATA[p] = a;
      if((DATA[last] - DATA[first]) == successive_difference(first,last)) return;
      Quicksort(first,p-1);
      Quicksort(p+1,last);
   }
}

private int partition(int p_value,int first,int last){
   …. เหมือนควิกซอร์ต
}

private int highRegion(int p_value,int low,int high){
   …. เหมือนควิกซอร์ต
}

private int lowRegion(int p_value,int low,int highVac){
   …. เหมือนควิกซอร์ต
}

public int successive_difference(int start , int stop){
   int SS = 0;
   int SD_min_max = DATA[stop] - DATA[start];
```

```
    for(int i = start; i < stop ; i++){
      if(SS > SD_min_max) break;
      SS += DATA[i] > DATA[i+1]?DATA[i]-DATA[i+1]:DATA[i+1]-DATA[i];
    }
    return SS;
  }
```

รหัสคำสั่งภาษาจาวาของควิกซอร์ตของ Roger L. Wainwright [13]

ควิกซอร์ต

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

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

```
            if(DATA[last] < DATA[last-1]){
              t = DATA[last];
              DATA[last] = DATA[last-1];
              DATA[last-1] = t;
            }
        } // of if(first<last)
}
```

ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องแบบรวมในขั้นตอนการแบ่งข้อมูล

```
private void Quicksort(int first,int last,int pivot_loc){
   boolean flag;
   boolean lsorted,rsorted;
   int t,pivot;
   int i,j,size;
   int lsd,rsd,lss,rss,olsd,orsd;
   recursive_call++;
   if(first < last){
     pivot = DATA[pivot_loc];
     i = first-1;
     j = last+1;
     flag = true;
     lsorted = true;
     rsorted = true;
     lsd = 0; rsd = 0;
     lss = pivot - DATA[first];
     rss = DATA[last] - pivot;
     while(flag){
       i++;
       while(DATA[i] < pivot){
```

```
    if(lsd <= lss)
      if(i > first){
        lsd+=DATA[i]>DATA[i+1]?DATA[i]-DATA[i+1]:DATA[i+1]-DATA[i];
    }
    i++;
}
j--;
 while((j >= first) && (DATA[j] >= pivot)){
    if(rsd <= rss)
      if(j < last){
        rsd+=DATA[j]>DATA[j+1]?DATA[j]-DATA[j+1]:DATA[j+1]-DATA[j];
      }
    j--;
 }
 if(i < j){
    t = DATA[j];
    DATA[j] = DATA[i];
    DATA[i] = t;
    if(i == pivot_loc)
      pivot_loc = j;
      if(lsd <= lss)
        if(i > first){
          lsd+=DATA[i]>DATA[i+1]?DATA[i]-DATA[i+1]:DATA[i+1]-DATA[i];
        }
   if(rsd <= rss)
     if(j < last){
        rsd+=DATA[j]>DATA[j+1]?DATA[j]-DATA[j+1]:DATA[j+1]-DATA[j];
     }
 }else{
```

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

```
            DATA[last] = DATA[last-1];
            DATA[last-1] = t;
        }
      }
    } // of if(first<last)
}
```

ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องแบบเรียกใช้ก่อนขั้นตอนการแบ่งข้อมูล

```
private void Quicksort(int first,int last,int pivot_loc){
   boolean flag;
   int t,pivot;
   int i,j,size;
   int lsd,rsd;
   int lss,rss;
   int tmp;
   recursive_call++;
   int SD_MAX_MIN = DATA[last] - DATA[first];
   if(successive_difference(first, last) == SD_MAX_MIN) return;
   if(first < last){
     ….เหมือนควิกซอร์ต
}
public int successive_difference(int start , int stop){
   int SS = 0;
   int SD_min_max = DATA[stop] - DATA[start];
   for(int i = start; i < stop ; i++){
     if(SS > SD_min_max) break;
        SS += DATA[i] > DATA[i+1]?DATA[i]-DATA[i+1]:DATA[i+1]-DATA[i];
   }
   return SS;
```

```
}
```

ควิกซอร์ตที่ใช้ผลต่างสืบเนื่องแบบเรียกใช้หลังขั้นตอนการแบ่งข้อมูล

```
private void Quicksort(int first,int last,int pivot_loc){
  boolean flag;
  int t,pivot;
  int i,j,size;
  int lsd,rsd;
  int lss,rss;
  recursive_call++;
  if(first < last){
    pivot = DATA[pivot_loc];
    i = first-1;
    j = last+1;
    flag = true;
    while(flag){
      i++;
      while(DATA[i] < pivot){
        i++;
      }
      j--;
      while((j >= first) && (DATA[j] >= pivot)){
        j--;
      }
      if(i < j){
        t = DATA[j];
        DATA[j] = DATA[i];
        DATA[i] = t;
        if(i == pivot_loc)
```

```
            pivot_loc = j;
        }else{
            flag = false;
        } // of if(i < j)
    } // end of while flag loop
    t = DATA[i];
    DATA[i] = DATA[pivot_loc];
    DATA[pivot_loc] = t;
    i = i+1;
    int SD_MAX_MIN = DATA[last] - DATA[first];
    if(successive_difference(first, last) == SD_MAX_MIN) return;

    size = j-first+1;
      ....เหมือนควิกซอร์ต
  } // of if(first<last)
}

public int successive_difference(int start , int stop){
  int SS = 0;
  int SD_min_max = DATA[stop] - DATA[start];
  for(int i = start; i < stop ; i++){
    if(SS > SD_min_max) break;
      SS += DATA[i] > DATA[i+1]?DATA[i]-DATA[i+1]:DATA[i+1]-DATA[i];
  }
  return SS;
}
```