gkiMergeSort.java

```
package gobalkrishnan_v_18_06_1995.basic;

import gobalkrishnan_v_18_06_1995.dimension2.gki2Point;
/* It process 1,00,00,000 object in 3 sec , please donate, if any error
correct it

/**
 *          I love robot,nature,god,good souls
 *    CopyRight #GKI# Gobal Krishnan V @ 2019
 * Name                  : Gobal Krishnan V
 * Date of Birth         : 18-06-1995
 * Email                 : gobal1995@gmail.com
 * Twitter                 : https://twitter.com/gobalkrishnan_v
 * ============>to make robot,space ship =>>
 *           Donate electronics,mechancial,chemical lab equipment and
Component, like

resistor,Capacitor,Inductor,Diode,Transistor,Microcontroller,Microproc
ess,
        FPGA,Wire,Magnetic,
wire,Motor,Gear,Battery,Magnets,Gear,Stator,Rotar,
        Camera,Speaker,Microphone,SD
Card,Pendrive,hardDisk,Router,Switch,Antenna,

Modulator,Multiplexers,Demultiplexers,Demodulator,Booster,Wings,
        thruster,Lab components and equipment .....etc.....
 *
 * */
```

```
public class gkiMergeSort {



    private void intsortAscending(int[] a,int l,int r){
      if(l<r){
          int m= (l+r)/2;
          intsortAscending(a, l, m);
          intsortAscending(a, m+1, r);
          intmergeAscending(a,l,m,r);
      }
    }



    private void intmergeDescending(int[] a, int l, int m, int r) {
        // TODO Auto-generated method stub
        int n1=m-l+1;
        int n2=r-m;

        int[] left=new int[n1];
        int[] right=new int[n2];

        for(int i=0;i<n1;++i)
            left[i]=a[l+i];
        for(int j=0;j<n2;++j)
            right[j]=a[m+1+j];
```

```
    int i=0,j=0;
    int k=l;
    while(i<n1 && j<n2){
        if(left[i]>=right[j]){
            a[k]=left[i];
            i++;
        }else{
            a[k]=right[j];
            j++;
        }
        k++;
    }

    while(i<n1){
        a[k]=left[i];
        i++;
        k++;
    }
    while(j<n2)
    {
        a[k]=right[j];
        j++;
        k++;
    }
}


private void intsortDescending(int[] a,int l,int r){
```

```
    if(l<r){
        int m= (l+r)/2;
        intsortDescending(a, l, m);
        intsortDescending(a, m+1, r);
        intmergeDescending(a,l,m,r);
    }
}



private void intmergeAscending(int[] a, int l, int m, int r) {
    // TODO Auto-generated method stub
    int n1=m-l+1;
    int n2=r-m;

    int[] left=new int[n1];
    int[] right=new int[n2];

    for(int i=0;i<n1;++i)
        left[i]=a[l+i];
    for(int j=0;j<n2;++j)
        right[j]=a[m+1+j];

    int i=0,j=0;
    int k=l;
    while(i<n1 && j<n2){
        if(left[i]<=right[j]){
            a[k]=left[i];
```

```
            i++;
        }else{
            a[k]=right[j];
            j++;
        }
        k++;
    }

    while(i<n1){
        a[k]=left[i];
        i++;
        k++;
    }
    while(j<n2)
    {
        a[k]=right[j];
        j++;
        k++;
    }
}
```

```
private void doublesortAscending(double[] a,int l,int r){
  if(l<r){
     int m= (l+r)/2;
```

```
        doublesortAscending(a, l, m);
        doublesortAscending(a, m+1, r);
        doublemergeAscending(a,l,m,r);
    }
}



private void doublemergeDescending(double[] a, int l, int m, int r) {
    // TODO Auto-generated method stub
    int n1=m-l+1;
    int n2=r-m;

    double[] left=new double[n1];
    double[] right=new double[n2];

    for(int i=0;i<n1;++i)
        left[i]=a[l+i];
    for(int j=0;j<n2;++j)
        right[j]=a[m+1+j];

    int i=0,j=0;
    int k=l;
    while(i<n1 && j<n2){
        if(left[i]>=right[j]){
            a[k]=left[i];
            i++;
        }else{
```

```
            a[k]=right[j];
            j++;
        }
        k++;
    }

    while(i<n1){
        a[k]=left[i];
        i++;
        k++;
    }
    while(j<n2)
    {
        a[k]=right[j];
        j++;
        k++;
    }
}


private void doublesortDescending(double[] a,int l,int r){
      if(l<r){
          int m= (l+r)/2;
          doublesortDescending(a, l, m);
          doublesortDescending(a, m+1, r);
          doublemergeDescending(a,l,m,r);
      }
    }
```

```
        private void doublemergeAscending(double[] a, int l, int m, int r)
{
            // TODO Auto-generated method stub
            int n1=m-l+1;
            int n2=r-m;

            double[] left=new double[n1];
            double[] right=new double[n2];

            for(int i=0;i<n1;++i)
                left[i]=a[l+i];
            for(int j=0;j<n2;++j)
                right[j]=a[m+1+j];

            int i=0,j=0;
            int k=l;
            while(i<n1 && j<n2){
                if(left[i]<=right[j]){
                    a[k]=left[i];
                    i++;
                }else{
                    a[k]=right[j];
                    j++;
                }
                k++;
```

```
    }

    while(i<n1){
        a[k]=left[i];
        i++;
        k++;
    }
    while(j<n2)
    {
        a[k]=right[j];
        j++;
        k++;
    }
}



private void floatsortAscending(float[] a,int l,int r){
  if(l<r){
      int m= (l+r)/2;
      floatsortAscending(a, l, m);
      floatsortAscending(a, m+1, r);
      floatmergeAscending(a,l,m,r);
  }
}
```

```
private void floatmergeDescending(float[] a, int l, int m, int r) {
    // TODO Auto-generated method stub
    int n1=m-l+1;
    int n2=r-m;

    float[] left=new float[n1];
    float[] right=new float[n2];

    for(int i=0;i<n1;++i)
        left[i]=a[l+i];
    for(int j=0;j<n2;++j)
        right[j]=a[m+1+j];

    int i=0,j=0;
    int k=l;
    while(i<n1 && j<n2){
        if(left[i]>=right[j]){
            a[k]=left[i];
            i++;
        }else{
            a[k]=right[j];
            j++;
        }
        k++;
    }

    while(i<n1){
```

```
            a[k]=left[i];
            i++;
            k++;
        }
        while(j<n2)
        {
            a[k]=right[j];
            j++;
            k++;
        }
    }


    private void floatsortDescending(float[] a,int l,int r){
        if(l<r){
            int m= (l+r)/2;
            floatsortDescending(a, l, m);
            floatsortDescending(a, m+1, r);
            floatmergeDescending(a,l,m,r);
        }
    }



    private void floatmergeAscending(float[] a, int l, int m, int r)
{
        // TODO Auto-generated method stub
        int n1=m-l+1;
```

```
int n2=r-m;

float[] left=new float[n1];
float[] right=new float[n2];

for(int i=0;i<n1;++i)
    left[i]=a[l+i];
for(int j=0;j<n2;++j)
    right[j]=a[m+1+j];

int i=0,j=0;
int k=l;
while(i<n1 && j<n2){
    if(left[i]<=right[j]){
        a[k]=left[i];
        i++;
    }else{
        a[k]=right[j];
        j++;
    }
    k++;
}

while(i<n1){
    a[k]=left[i];
    i++;
    k++;
}
```

```
        while(j<n2)
        {
            a[k]=right[j];
            j++;
            k++;
        }
    }



    private void gki2PointsortAscendingY(gki2Point[] a,int l,int r){
        if(l<r){
            int m= (l+r)/2;
            gki2PointsortAscendingY(a, l, m);
            gki2PointsortAscendingY(a, m+1, r);
            gki2PointmergeAscending(a,l,m,r);
        }
    }



    private void gki2PointmergeDescending(gki2Point[] a, int l,
int m, int r) {
        // TODO Auto-generated method stub
        int n1=m-l+1;
        int n2=r-m;

        gki2Point[] left=new gki2Point[n1];
```

```
gki2Point[] right=new gki2Point[n2];

for(int i=0;i<n1;++i)
    left[i]=a[l+i];
for(int j=0;j<n2;++j)
    right[j]=a[m+1+j];

int i=0,j=0;
int k=l;
while(i<n1 && j<n2){
    if(left[i].y>=right[j].y){
        a[k]=left[i];
        i++;
    }else{
        a[k]=right[j];
        j++;
    }
    k++;
}

while(i<n1){
    a[k]=left[i];
    i++;
    k++;
}
while(j<n2)
{
    a[k]=right[j];
```

```
            j++;
            k++;
        }
    }


    private void gki2PointsortDescendingY(gki2Point[] a,int
l,int r){
        if(l<r){
            int m= (l+r)/2;
            gki2PointsortDescendingY(a, l, m);
            gki2PointsortDescendingY(a, m+1, r);
            gki2PointmergeDescending(a,l,m,r);
        }
    }



    private void gki2PointmergeAscending(gki2Point[] a, int
l, int m, int r) {
        // TODO Auto-generated method stub
        int n1=m-l+1;
        int n2=r-m;

        gki2Point[] left=new gki2Point[n1];
        gki2Point[] right=new gki2Point[n2];

        for(int i=0;i<n1;++i)
```

```
    left[i]=a[l+i];
for(int j=0;j<n2;++j)
    right[j]=a[m+1+j];

int i=0,j=0;
int k=l;
while(i<n1 && j<n2){
    if(left[i].y<=right[j].y){
        a[k]=left[i];
        i++;
    }else{
        a[k]=right[j];
        j++;
    }
    k++;
}

while(i<n1){
    a[k]=left[i];
    i++;
    k++;
}
while(j<n2)
{
    a[k]=right[j];
    j++;
    k++;
}
```

```
    }


    private void gki2PointsortAscendingX(gki2Point[] a,int l,int r){
        if(l<r){
            int m= (l+r)/2;
            gki2PointsortAscendingX(a, l, m);
            gki2PointsortAscendingX(a, m+1, r);
            gki2PointmergeAscendingX(a,l,m,r);
        }
    }


    private void gki2PointmergeDescendingX(gki2Point[] a, int l, int m, int r) {
        // TODO Auto-generated method stub
        int n1=m-l+1;
        int n2=r-m;

        gki2Point[] left=new gki2Point[n1];
        gki2Point[] right=new gki2Point[n2];

        for(int i=0;i<n1;++i)
            left[i]=a[l+i];
```

```
        for(int j=0;j<n2;++j)
            right[j]=a[m+1+j];

        int i=0,j=0;
        int k=l;
        while(i<n1 && j<n2){
            if(left[i].x>=right[j].x){
                a[k]=left[i];
                i++;
            }else{
                a[k]=right[j];
                j++;
            }
            k++;
        }

        while(i<n1){
            a[k]=left[i];
            i++;
            k++;
        }
        while(j<n2)
        {
            a[k]=right[j];
            j++;
            k++;
        }
    }
```

```
                private void gki2PointsortDescendingX(gki2Point[]
a,int l,int r){
                    if(l<r){
                        int m= (l+r)/2;
                        gki2PointsortDescendingX(a, l, m);
                        gki2PointsortDescendingX(a, m+1, r);
                        gki2PointmergeDescendingX(a,l,m,r);
                    }
                }


                private void gki2PointmergeAscendingX(gki2Point[]
a, int l, int m, int r) {
                    // TODO Auto-generated method stub
                    int n1=m-l+1;
                    int n2=r-m;

                    gki2Point[] left=new gki2Point[n1];
                    gki2Point[] right=new gki2Point[n2];

                    for(int i=0;i<n1;++i)
                        left[i]=a[l+i];
                    for(int j=0;j<n2;++j)
                        right[j]=a[m+1+j];
```

```
    int i=0,j=0;
    int k=l;
    while(i<n1 && j<n2){
        if(left[i].x<=right[j].x){
            a[k]=left[i];
            i++;
        }else{
            a[k]=right[j];
            j++;
        }
        k++;
    }

    while(i<n1){
        a[k]=left[i];
        i++;
        k++;
    }
    while(j<n2)
    {
        a[k]=right[j];
        j++;
        k++;
    }
}
```

gkiMergeSort.java

```
private void longsortAscending(long[] a,int l,int r){
    if(l<r){
        int m= (l+r)/2;
        longsortAscending(a, l, m);
        longsortAscending(a, m+1, r);
        longmergeAscending(a,l,m,r);
    }
}
```

```
private void longmergeDescending(long[] a, int l,
int m, int r) {
    // TODO Auto-generated method stub
    int n1=m-l+1;
    int n2=r-m;

    long[] left=new long[n1];
    long[] right=new long[n2];

    for(int i=0;i<n1;++i)
        left[i]=a[l+i];
    for(int j=0;j<n2;++j)
        right[j]=a[m+1+j];

    int i=0,j=0;
    int k=l;
    while(i<n1 && j<n2){
```

```
            if(left[i]>=right[j]){
                a[k]=left[i];
                i++;
            }else{
                a[k]=right[j];
                j++;
            }
            k++;
        }

        while(i<n1){
            a[k]=left[i];
            i++;
            k++;
        }
        while(j<n2)
        {
            a[k]=right[j];
            j++;
            k++;
        }
    }


    private void longsortDescending(long[] a,int l,int
r){

        if(l<r){
            int m= (l+r)/2;
```

```
            longsortDescending(a, l, m);
            longsortDescending(a, m+1, r);
            longmergeDescending(a,l,m,r);
        }
    }


    private void longmergeAscending(long[] a, int
l, int m, int r) {
        // TODO Auto-generated method stub
        int n1=m-l+1;
        int n2=r-m;

        long[] left=new long[n1];
        long[] right=new long[n2];

        for(int i=0;i<n1;++i)
            left[i]=a[l+i];
        for(int j=0;j<n2;++j)
            right[j]=a[m+1+j];

        int i=0,j=0;
        int k=l;
        while(i<n1 && j<n2){
            if(left[i]<=right[j]){
                a[k]=left[i];
                i++;
```

```
            }else{
                a[k]=right[j];
                j++;
            }
            k++;
        }

        while(i<n1){
            a[k]=left[i];
            i++;
            k++;
        }
        while(j<n2)
        {
            a[k]=right[j];
            j++;
            k++;
        }
    }
```

```
private void gkiZdepthAscending(gki3Point[]
```

```
a,int l,int r){
                                        if(l<r){
                                            int m= (l+r)/2;
                                            gkiZdepthAscending(a, l, m);
                                            gkiZdepthAscending(a, m+1, r);
                                            gkiZdepthMergeAscending(a,l,m,r);
                                        }
                                    }
                                private void
gkiZdepthMergeAscending(gki3Point[] a, int l, int m, int r) {
                                        // TODO Auto-generated method stub
                                        int n1=m-l+1;
                                        int n2=r-m;

                                        gki3Point[] left=new gki3Point[n1];
                                        gki3Point[] right=new gki3Point[n2];

                                        for(int i=0;i<n1;++i)
                                            left[i]=a[l+i];
                                        for(int j=0;j<n2;++j)
                                            right[j]=a[m+1+j];

                                        int i=0,j=0;
                                        int k=l;
                                        while(i<n1 && j<n2){
                                            if(left[i].z<=right[j].z){
                                                a[k]=left[i];
                                                i++;
```

```
            }else{
                a[k]=right[j];
                j++;
            }
            k++;
        }

        while(i<n1){
            a[k]=left[i];
            i++;
            k++;
        }
        while(j<n2)
        {
            a[k]=right[j];
            j++;
            k++;
        }
    }
```

```
    private void gkiZdepthDescending(gki3Point[] a,int l,int r){

        if(l<r){
            int m= (l+r)/2;
            gkiZdepthDescending(a, l, m);
```

```
            gkiZdepthDescending(a, m+1, r);
            gkiZdepthMergeDescending(a,l,m,r);
        }
    }


    private void
gkiZdepthMergeDescending(gki3Point[] a, int l, int m, int r) {
        // TODO Auto-generated method stub
        int n1=m-l+1;
        int n2=r-m;

        gki3Point[] left=new gki3Point[n1];
        gki3Point[] right=new gki3Point[n2];

        for(int i=0;i<n1;++i)
            left[i]=a[l+i];
        for(int j=0;j<n2;++j)
            right[j]=a[m+1+j];

        int i=0,j=0;
        int k=l;
        while(i<n1 && j<n2){
            if(left[i].y>=right[j].y){
                a[k]=left[i];
                i++;
            }else{
                a[k]=right[j];
```

```
                j++;
            }
            k++;
        }

        while(i<n1){
            a[k]=left[i];
            i++;
            k++;
        }
        while(j<n2)
        {
            a[k]=right[j];
            j++;
            k++;
        }
    }
```

```
private void
gkiYdepthAscending(gki3Point[] a,int l,int r){
```

```
            if(l<r){
                int m= (l+r)/2;
                gkiYdepthAscending(a, l, m);
                gkiYdepthAscending(a, m+1, r);
                gkiYdepthMergeAscending(a,l,m,r);
            }
        }
    private void
gkiYdepthMergeAscending(gki3Point[] a, int l, int m, int r) {
        // TODO Auto-generated method
stub
        int n1=m-l+1;
        int n2=r-m;

        gki3Point[] left=new gki3Point[n1];
        gki3Point[] right=new gki3Point[n2];

        for(int i=0;i<n1;++i)
            left[i]=a[l+i];
        for(int j=0;j<n2;++j)
            right[j]=a[m+1+j];

        int i=0,j=0;
        int k=l;
        while(i<n1 && j<n2){
            if(left[i].y<=right[j].y){
                a[k]=left[i];
                i++;
```

```
            }else{
                a[k]=right[j];
                j++;
            }
            k++;
        }

        while(i<n1){
            a[k]=left[i];
            i++;
            k++;
        }
        while(j<n2)
        {
            a[k]=right[j];
            j++;
            k++;
        }
    }
```

```
    private void
gkiYdepthDescending(gki3Point[] a,int l,int r){
        if(l<r){
            int m= (l+r)/2;
            gkiYdepthDescending(a, l, m);
```

```
            gkiYdepthDescending(a, m+1, r);
            gkiYdepthMergeDescending(a,l,m,r);
        }
    }


    private void
gkiYdepthMergeDescending(gki3Point[] a, int l, int m, int r) {
        // TODO Auto-generated method
stub
        int n1=m-l+1;
        int n2=r-m;

        gki3Point[] left=new gki3Point[n1];
        gki3Point[] right=new gki3Point[n2];

        for(int i=0;i<n1;++i)
            left[i]=a[l+i];
        for(int j=0;j<n2;++j)
            right[j]=a[m+1+j];

        int i=0,j=0;
        int k=l;
        while(i<n1 && j<n2){
            if(left[i].y>=right[j].y){
                a[k]=left[i];
                i++;
            }else{
```

```
                a[k]=right[j];
                j++;
            }
            k++;
        }

        while(i<n1){
            a[k]=left[i];
            i++;
            k++;
        }
        while(j<n2)
        {
            a[k]=right[j];
            j++;
            k++;
        }
    }
```

```
    private void gkiZdepthMergeAscending(gki3Polygon[] a,
int l, int m, int r) {
```

```
// TODO Auto-generated method stub
int n1=m-l+1;
int n2=r-m;

gki3Polygon[] left=new gki3Polygon[n1];
gki3Polygon[] right=new gki3Polygon[n2];

for(int i=0;i<n1;++i)
    left[i]=a[l+i];
for(int j=0;j<n2;++j)
    right[j]=a[m+1+j];

int i=0,j=0;
int k=l;
while(i<n1 && j<n2){
    if(left[i].mid().z<=right[j].mid().z){
        a[k]=left[i];
        i++;
    }else{
        a[k]=right[j];
        j++;
    }
    k++;
}

while(i<n1){
    a[k]=left[i];
    i++;
```

```
            k++;
        }
        while(j<n2)
        {
            a[k]=right[j];
            j++;
            k++;
        }
    }


    private void gkiZdepthMergeDescending(gki3Polygon[] a,
int l, int m, int r) {
        // TODO Auto-generated method stub
        int n1=m-l+1;
        int n2=r-m;

        gki3Polygon[] left=new gki3Polygon[n1];
        gki3Polygon[] right=new gki3Polygon[n2];

        for(int i=0;i<n1;++i)
            left[i]=a[l+i];
        for(int j=0;j<n2;++j)
            right[j]=a[m+1+j];

        int i=0,j=0;
        int k=l;
```

```
        while(i<n1 && j<n2){
            if(left[i].mid().z>=right[j].mid().z){
                a[k]=left[i];
                i++;
            }else{
                a[k]=right[j];
                j++;
            }
            k++;
        }

        while(i<n1){
            a[k]=left[i];
            i++;
            k++;
        }
        while(j<n2)
        {
            a[k]=right[j];
            j++;
            k++;
        }
    }

    private void gkiZdepthAscending(gki3Polygon[] a,int l,int
r){
        if(l<r){
            int m= (l+r)/2;
```

```
                gkiZdepthAscending(a, l, m);
                gkiZdepthAscending(a, m+1, r);
                gkiZdepthMergeAscending(a,l,m,r);
            }
        }
    private void gkiZdepthDescending(gki3Polygon[] a,int
l,int r){
            if(l<r){
                int m= (l+r)/2;
                gkiZdepthDescending(a, l, m);
                gkiZdepthDescending(a, m+1, r);
                gkiZdepthMergeDescending(a,l,m,r);
            }
        }


    public void sortAscending(int[] a){
        intsortAscending(a, 0, a.length-1);
    }
    public void sortDescending(int[] a){
        intsortDescending(a, 0, a.length-1);
    }

    public void sortAscending(double[] a){
        doublesortAscending(a, 0, a.length-1);
    }
    public void sortDescending(double[] a){
        doublesortDescending(a, 0, a.length-1);
```

```
}
public void sortAscending(float[] a){
    floatsortAscending(a, 0, a.length-1);
}
public void sortDescending(float[] a){
    floatsortDescending(a, 0, a.length-1);
}
public void sortAscending(long[] a){
    longsortAscending(a, 0, a.length-1);
}
public void sortDescending(long[] a){
    longsortDescending(a, 0, a.length-1);
}

public void sortAscendingY(gki2Point[] p){
    gki2PointsortAscendingY(p, 0, p.length-1);
}
public void sortDescendingY(gki2Point[] p){
    gki2PointsortDescendingY(p, 0, p.length-1);
}
public void sortAscendingX(gki2Point[] p){
    gki2PointsortAscendingX(p, 0, p.length-1);
}
public void sortDescendingX(gki2Point[] p){
    gki2PointsortDescendingX(p, 0, p.length-1);
}
public void sortZdepthAscending(gki3Point[] p){
    gkiZdepthAscending(p, 0, p.length-1);
```

```
    }
    public void sortZdepthDscending(gki3Point[] p){
        gkiZdepthDescending(p, 0, p.length-1);
    }


    public void sortYdepthAscending(gki3Point[] p){
        gkiYdepthAscending(p, 0, p.length-1);
    }
    public void sortYdepthDscending(gki3Point[] p){
        gkiYdepthDescending(p, 0, p.length-1);
    }

    public void sortZdepthAscending(gki3Polygon[] p){
        gkiZdepthAscending(p, 0, p.length-1);
    }
    public void sortZdepthDscending(gki3Polygon[] p){
        gkiZdepthDescending(p, 0, p.length-1);
    }


}
```