

```

package _datos;

import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;

import us.lsi.common.Files2;

public class DatosCafes {

    public static List<Integer> tipos;
    public static List<Variedad> variedades;

    public record Variedad(int id, Integer beneficio, List<Double> composicion) {
        public static int cont;
        public static Variedad create(String linea) {
            List<Double> compo = new ArrayList<>();
            for (int j=0; j<tipos.size(); j++) {
                compo.add(0.);
            }
            String[] vari = linea.split(";");
            Integer benef = Integer.parseInt(vari[0].split("=")[1].replace(";", "").trim());
            String[] composiciones = vari[1].split("=")[1].trim().split(",");
            for (int j=0; j<composiciones.length; j++) {
                String[] percent = composiciones[j].replace(" (C", "").replace(")", "").split(":");
                Integer tipo = Integer.parseInt(percent[0].trim())-1;
                Double porcentaje = Double.parseDouble(percent[1].trim());
                compo.set(tipo, porcentaje);
            }
            return new Variedad(cont++, benef, new ArrayList<>(compo));
        }
    }

    public static void iniDatos(String fichero) {

        Variedad.cont=0;

        List<String> lineas = Files2.linesFromFile(fichero);
        int pos = lineas.indexOf("// VARIEDADES");
        List<String> tipoCafe = lineas.subList(1, pos);
        List<String> varCafe = lineas.subList(pos+1, lineas.size());

        List<Integer> aux = new ArrayList<>();
        for (int i=0; i<tipoCafe.size(); i++) {
            Integer valor = Integer.parseInt(tipoCafe.get(i).split("=")[1].replace(";", "").trim());
            aux.add(valor);
        }
        tipos = new ArrayList<>(aux);

        variedades = new ArrayList<>();
        for (int i=0; i<varCafe.size(); i++) {
            variedades.add(Variedad.create(varCafe.get(i)));
        }
        toConsole();
    }

    // Double getKgTipoVariedad(Integer i, Integer j)

    public static Integer getNumTipos() {
        return tipos.size();
    }

    public static Integer getNumVariedades() {
        return variedades.size();
    }

    public static Integer getKgTipo(Integer j) {

```

```

        return tipos.get(j);
    }

    public static Integer getBeneficioVariedad(Integer i) {
        return variedades.get(i).beneficio();
    }

    public static Double getKgTipoVariedad(Integer j, Integer i) {
        return variedades.get(i).composicion().get(j);
    }

    public static List<Variedad> getVariedades() {
        return new ArrayList<>(variedades);
    }

    public static Integer getMaxKgVariedad(Integer i) {
        List<Double> listaMax = new ArrayList<>();

        for (int j=0; j<tipos.size(); j++) {
            listaMax.add(getKgTipo(j) / getKgTipoVariedad(j, i));
        }
        listaMax.sort(Comparator.naturalOrder());
        return listaMax.get(0).intValue();
    }

    private static void toConsole() {
        System.out.println(
            "Kgs disponibles de cada tipo: "+tipos+
            "\nVariedades disponibles: "+variedades
        );
    }

    public static void main(String[] args) {
        for (int i=0; i<3; i++) {
            System.out.println("===== DATOS DE ENTRADA "+(i+1)+" =====");
            String fichero = "ficheros/Ejercicio1DatosEntrada"+String.valueOf(i+1)+".txt";
            iniDatos(fichero);
            System.out.println("\n\n");
        }
    }
}

```

```

package ejercicio1;

import java.io.IOException;
import java.util.List;
import java.util.Locale;

import _datos.DatosCafes;
import _datos.DatosCafes.Variedad;
import us.lsi.gurobi.GurobiLp;
import us.lsi.gurobi.GurobiSolution;
import us.lsi.solve.AuxGrammar;

public class Ejercicio1PLE {

    public static List<Integer> tipos;
    public static List<Variedad> variedades;

    public static Integer getNumTipos() {
        return tipos.size();
    }

    public static Integer getNumVariedades() {
        return variedades.size();
    }

    public static Integer getKgTipo(Integer j) {
        return tipos.get(j);
    }

    public static Integer getBeneficioVariedad(Integer i) {
        return variedades.get(i).beneficio();
    }

    public static Double getKgTipoVariedad(Integer j, Integer i) {
        return variedades.get(i).composicion().get(j);
    }

    public static void Ejercicio1_model() throws IOException {
        for (int i=0; i<3; i++) {

            DatosCafes.iniDatos("ficheros/Ejercicio1DatosEntrada"+(i+1)+".txt");

            tipos = DatosCafes.tipos;
            variedades = DatosCafes.variedades;

            AuxGrammar.generate(Ejercicio1PLE.class, "lsi_models/Ejercicio1.lsi",
"gurobi_models/Ejercicio1-"+(i+1)+".lp");
            GurobiSolution solucion = GurobiLp.gurobi("gurobi_models/Ejercicio1-"+(i+1)+".lp");
            Locale.setDefault(new Locale("en", "US"));
            System.out.println(solucion.toString((s,d)->d>0.)+"\n\n");

        }
    }

    public static void main(String[] args) throws IOException {
        Ejercicio1_model();
    }
}

```

# Ejercicio1.lsi

head section

```
Integer getNumTipos()
Integer getNumVariedades()
Integer getKgTipo(Integer j)
Integer getBeneficioVariedad(Integer i)
Double getKgTipoVariedad(Integer i, Integer j)
```

```
Integer n = getNumTipos()
Integer m = getNumVariedades()
```

goal section

```
// Objetivo: maximizar el beneficio obtenido
max sum(getBeneficioVariedad(i) x[i], i in 0 .. m)
```

constraints section

```
// Para cada tipo de café, no se puede superar la cantidad de kilos
disponibles
sum(getKgTipoVariedad(j,i) x[i], i in 0 .. m) <= getKgTipo(j), j in 0 .. n
```

int

```
x[i], i in 0 .. m
```

```

package _soluciones;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

import _datos.DatosCafes;
import _datos.DatosCafes.Variedad;

public class SolucionCafes {

    public static SolucionCafes of_Range(List<Integer> value) {
        return new SolucionCafes(value);
    }

    private Double beneficio;
    private List<Variedad> soluciones;
    private List<Integer> solucion;

    private SolucionCafes() {
        beneficio=0.;
        soluciones = new ArrayList<>();
        solucion = new ArrayList<>();
    }

    private SolucionCafes(List<Integer> ls) {
        beneficio = 0.;
        soluciones = new ArrayList<>();
        solucion = new ArrayList<>();
        for (int i=0; i<ls.size(); i++) {
            if (ls.get(i)>0) {
                Integer kg = ls.get(i);
                Integer bv = DatosCafes.getBeneficioVariedad(i)*kg;
                soluciones.add(DatosCafes.getVariedades().get(i));
                solucion.add(ls.get(i));
                beneficio += bv;
            }
        }
    }

    public static SolucionCafes empty() {
        return new SolucionCafes();
    }

    public String toString() {
        String s = soluciones.stream()
            .map(v -> "P"+(v.id()+1)+" : "+solucion.get(soluciones.indexOf(v)))
            .collect(Collectors.joining(" Kg\n", "Kgs producidos de cada variedad:\n", " Kg\n"));
        return String.format("%sBeneficio obtenido: %.1f", s, beneficio) ;
    }

}

```

```

package ejercicio1;

import java.util.List;

import _datos.DatosCafes;
import _soluciones.SolucionCafes;
import us.lsi.ag.ValuesInRangeData;
import us.lsi.ag.agchromosomes.ChromosomeFactory.ChromosomeType;

public class InRangeCafesAG
    implements ValuesInRangeData<Integer, SolucionCafes> {

    public InRangeCafesAG(String fichero) {
        DatosCafes.iniDatos(fichero);
    }

    @Override
    public Integer size() {
        return DatosCafes.getNumVariedades();
    }

    @Override
    public ChromosomeType type() {
        return ChromosomeType.Range;
    }

    @Override
    public Double fitnessFunction(List<Integer> value) {
        double goal = 0, error = 0, dif = 0, k = 0;
        for (int i=0; i<size(); i++) {
            if (value.get(i)>0) {
                goal += value.get(i)*DatosCafes.getBeneficioVariedad(i);
            }
        }
        for (int j=0; j<DatosCafes.getNumTipos(); j++) {
            dif=0;
            // Restricción de kg disponibles de cada tipo
            for (int i=0; i<size(); i++) {
                dif += value.get(i)*DatosCafes.getKgTipoVariedad(j, i);
            }
            if (dif>DatosCafes.getKgTipo(j)) {
                error += dif - DatosCafes.getKgTipo(j);
            }
        }

        // Cálculo de k
        for (int i=0; i<size(); i++) {
            k += Math.pow((DatosCafes.getMaxKgVariedad(i)*DatosCafes.getBeneficioVariedad(i)), 2);
        }

        return goal -k*error;
    }

    @Override
    public SolucionCafes solucion(List<Integer> value) {
        return SolucionCafes.of_Range(value);
    }

    @Override
    public Integer max(Integer i) {
        return DatosCafes.getMaxKgVariedad(i)+1;
    }

    @Override
    public Integer min(Integer i) {
        return 0;
    }
}

```



```
package ejercicio1;

import java.util.List;
import java.util.Locale;

import _soluciones.SolucionCafes;
import us.lsi.ag.agchromosomes.AlgoritmoAG;
import us.lsi.ag.agstopping.StoppingConditionFactory;

public class TestCafeAGRange {

    public static void main(String[] args) {
        Locale.setDefault(new Locale("en", "US"));

        AlgoritmoAG.ELITISM_RATE = 0.10;
        AlgoritmoAG.CROSSOVER_RATE = 0.95;
        AlgoritmoAG.MUTATION_RATE = 0.8;
        AlgoritmoAG.POPULATION_SIZE = 1000;

        StoppingConditionFactory.NUM_GENERATIONS = 1000;
        StoppingConditionFactory.stoppingConditionType =
        StoppingConditionFactory.StoppingConditionType.GenerationCount;

        for (int i=0; i<3; i++) {
            InRangeCafesAG p = new InRangeCafesAG("ficheros/Ejercicio1DatosEntrada" + (i+1) + ".txt");

            AlgoritmoAG<List<Integer>, SolucionCafes> ap = AlgoritmoAG.of(p);
            ap.ejecuta();

            System.out.println("=====");
            System.out.println(ap.bestSolution());
            System.out.println("=====\n");
        }
    }
}
```



```

package _datos;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import us.lsi.common.Files2;

public class DatosCursos {

    public static List<Curso> cursos;
    public static Integer maxCentros;

    public record Curso(Integer id, List<Integer> tematicas, Double precio, Integer centro) {
        public static int cont;
        public static Curso create(String linea) {
            List<Integer> aux = new ArrayList<>();
            String[] params = linea.split(":");
            String[] tem = params[0].substring(1, params[0].length()-1).split(",");
            for (String str : tem) {
                aux.add(Integer.parseInt(str.trim()));
            }
            return new Curso(cont++, new ArrayList<>(aux), Double.parseDouble(params[1].trim()),
                Integer.parseInt(params[2].trim()));
        }
    }

    public static void iniDatos(String fichero) {
        List<Curso> aux = new ArrayList<>();
        Curso.cont=0;

        List<String> lineas = Files2.linesFromFile(fichero);
        maxCentros = Integer.parseInt(lineas.get(0).split("=")[1].trim());

        for (String s : lineas.subList(1, lineas.size())) {
            aux.add(Curso.create(s));
        }
        cursos = new ArrayList<>(aux);
        toConsole();
    }

    public static Integer getMaxCentros() {
        return maxCentros;
    }

    public static Integer getNumCursos() {
        return cursos.size();
    }

    public static List<Integer> getTematicas() {
        Set<Integer> s = new HashSet<>();
        for (Curso c : cursos) {
            s.addAll(c.tematicas());
        }
        return new ArrayList<>(s);
    }

    public static Integer getNumTematicas() {
        return getTematicas().size();
    }

    public static List<Integer> getTematicasCurso(Integer i) {
        return cursos.get(i).tematicas();
    }

    public static Integer getNumTematicasCurso(Integer i) {
        return getTematicasCurso(i).size();
    }
}

```

```

public static Integer contieneTematica(Integer i, Integer j) {
    return cursos.get(i).tematicas().contains(getTematicas().get(j))?
        1:
        0;
}

public static Double getPrecioCurso(Integer i) {
    return cursos.get(i).precio();
}

public static Integer getCentroCurso(Integer i) {
    return cursos.get(i).centro();
}

public static List<Integer> getCentros() {
    Set<Integer> s = new HashSet<>();
    for (Curso c : cursos) {
        s.add(c.centro());
    }
    return new ArrayList<>(s);
}

public static Integer getNumCentros() {
    return getCentros().size();
}

public static Integer ofreceCurso(Integer i, Integer k) {
    return cursos.get(i).centro().equals(getCentros().get(k))?
        1:
        0;
}

public static void toConsole() {
    System.out.println(
        "Número máximo de centros a seleccionar: "+maxCentros+
        "\nCursos disponibles: "+cursos
    );
}

public static void main(String[] args) {
    for (int i=0; i<3; i++) {
        System.out.println("===== DATOS DE ENTRADA "+(i+1)+" =====");
        String fichero = "ficheros/Ejercicio2DatosEntrada"+String.valueOf(i+1)+".txt";
        iniDatos(fichero);
        System.out.println("\n\n");
    }
}
}

```

```

package ejercicio2;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Locale;
import java.util.Set;

import _datos.DatosCursos;
import _datos.DatosCursos.Curso;
import us.lsi.gurobi.GurobiLp;
import us.lsi.gurobi.GurobiSolution;
import us.lsi.solve.AuxGrammar;

public class Ejercicio2PLE {

    public static List<Curso> cursos;
    public static Integer maxCentros;

    public static Integer getMaxCentros() {
        return maxCentros;
    }

    public static Integer getNumCursos() {
        return cursos.size();
    }

    public static List<Integer> getTematicas() {
        Set<Integer> s = new HashSet<>();
        for (Curso c : cursos) {
            s.addAll(c.tematicas());
        }
        return new ArrayList<>(s);
    }

    public static Integer getNumTematicas() {
        return getTematicas().size();
    }

    public static List<Integer> getTematicasCurso(Integer i) {
        return cursos.get(i).tematicas();
    }

    public static Integer getNumTematicasCurso(Integer i) {
        return getTematicasCurso(i).size();
    }

    public static Integer contieneTematica(Integer i, Integer j) {
        return cursos.get(i).tematicas().contains(getTematicas().get(j))?
            1:
            0;
    }

    public static Double getPrecioCurso(Integer i) {
        return cursos.get(i).precio();
    }

    public static Integer getCentroCurso(Integer i) {
        return cursos.get(i).centro();
    }

    public static List<Integer> getCentros() {
        Set<Integer> s = new HashSet<>();
        for (Curso c : cursos) {
            s.add(c.centro());
        }
        return new ArrayList<>(s);
    }

    public static Integer getNumCentros() {
        return getCentros().size();
    }
}

```

```

public static Integer ofreceCurso(Integer i, Integer k) {
    return cursos.get(i).centro().equals(getCentros().get(k)) ?
        1 :
        0;
}

public static void Ejercicio2_model() throws IOException {
    for (int i=0; i<3; i++) {

        DatosCursos.iniDatos("ficheros/Ejercicio2DatosEntrada" + (i+1) + ".txt");

        cursos = DatosCursos.cursos;
        maxCentros = DatosCursos.maxCentros;

        AuxGrammar.generate(Ejercicio2PLE.class, "lsi_models/Ejercicio2.lsi",
"gurobi_models/Ejercicio2-" + (i+1) + ".lp");
        GurobiSolution solucion = GurobiLp.gurobi("gurobi_models/Ejercicio2-" + (i+1) + ".lp");
        Locale.setDefault(new Locale("en", "US"));
        System.out.println(solucion.toString((s,d)->d>0.));

    }
}

public static void main(String[] args) throws IOException {
    Ejercicio2_model();
}
}

```

# Ejercicio2.lsi

head section

```
Integer getNumCursos()
Integer getNumCentros()
Integer getNumTematicas()
Integer getMaxCentros()
Double getPrecioCurso(Integer i)
Integer contieneTematica(Integer i, Integer j)
Integer ofreceCurso(Integer i, Integer k)
```

```
Integer n = getNumCursos()
Integer m = getNumTematicas()
Integer nc = getNumCentros()
Integer maxCentros = getMaxCentros()
```

goal section

```
// Objetivo: minimizar el precio de los cursos seleccionados
min sum(getPrecioCurso(i) x[i], i in 0 .. n)
```

constraints section

```
// Cada temática se tiene que seleccionar al menos una vez
sum(contieneTematica(i,j) x[i], i in 0 .. n) >= 1, j in 0 .. m
```

```
// No se puede superar el número máximo de centros permitidos
sum(y[k], k in 0 .. nc) <= maxCentros
```

```
// Para cada centro de cada curso, si se selecciona un curso de dicho centro,  
se selecciona también el centro  
ofreceCurso(i,k) x[i] - y[k] <= 0, i in 0 .. n, k in 0 .. nc
```

bin

```
x[i], i in 0 .. n
y[k], k in 0 .. nc
```

```
package _soluciones;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

import _datos.DatosCursos;
import _datos.DatosCursos.Curso;

public class SolucionCursos {

    public static SolucionCursos of_Range(List<Integer> value) {
        return new SolucionCursos(value);
    }

    private Double precio;
    private List<Curso> cursos;

    public SolucionCursos() {
        precio = 0.;
        cursos = new ArrayList<>();
    }

    public SolucionCursos(List<Integer> ls) {
        precio = 0.;
        cursos = new ArrayList<>();
        for (int i=0; i<ls.size(); i++) {
            if (ls.get(i)>0) {
                precio += DatosCursos.getPrecioCurso(i);
                cursos.add(DatosCursos.cursos.get(i));
            }
        }
    }

    public static SolucionCursos empty() {
        return new SolucionCursos();
    }

    @Override
    public String toString() {
        String s = cursos.stream()
            .map(c -> "S"+c.id())
            .collect(Collectors.joining(", ", "Cursos seleccionados: {", "}\n"));
        return String.format("%sCoste total: %.1f", s, precio);
    }
}
```

```

package ejercicio2;

import java.util.HashSet;
import java.util.List;
import java.util.Set;

import _datos.DatosCursos;
import _soluciones.SolucionCursos;
import us.lsi.ag.BinaryData;

public class BinCursosAG
    implements BinaryData<SolucionCursos> {

    public BinCursosAG(String fichero) {
        DatosCursos.iniDatos(fichero);
    }

    @Override
    public Integer size() {
        return DatosCursos.getNumCursos();
    }

    @Override
    public Double fitnessFunction(List<Integer> value) {
        // TODO Auto-generated method stub
        double goal = 0, error = 0, k = 0, suma = 0;

        for (int i=0; i<value.size(); i++) {
            if (value.get(i)>0) {
                goal += DatosCursos.getPrecioCurso(i);
            }
        }

        Set<Integer> ts = new HashSet<>();
        Set<Integer> cs = new HashSet<>();
        for (int i=0; i<value.size(); i++) {
            if (value.get(i)>0) {
                ts.addAll(DatosCursos.getTematicasCurso(i));
                cs.add(DatosCursos.getCentroCurso(i));
            }
        }
        Integer m = DatosCursos.getNumTematicas();
        Integer nc = DatosCursos.getMaxCentros();
        // Restricción de selección de temáticas
        if (ts.size() < m) {
            error += m-ts.size();
        }
        // Restricción de selección de centros
        if (cs.size() > nc) {
            error += cs.size()-nc;
        }

        // Cálculo de k
        for (int i=0; i<value.size(); i++) {
            suma += DatosCursos.getPrecioCurso(i);
        }

        k += Math.pow(suma, 2);

        return -goal -k*error;
    }

    @Override
    public SolucionCursos solucion(List<Integer> value) {
        return SolucionCursos.of_Range(value);
    }
}

```





```
package ejercicio2;

import java.util.List;
import java.util.Locale;

import _soluciones.SolucionCursos;
import us.lsi.ag.agchromosomes.AlgoritmoAG;
import us.lsi.ag.agstopping.StoppingConditionFactory;

public class TestCursosAGBin {

    public static void main(String[] args) {
        Locale.setDefault(new Locale("en", "US"));

        AlgoritmoAG.ELITISM_RATE = 0.10;
        AlgoritmoAG.CROSSOVER_RATE = 0.95;
        AlgoritmoAG.MUTATION_RATE = 0.8;
        AlgoritmoAG.POPULATION_SIZE = 1000;

        StoppingConditionFactory.NUM_GENERATIONS = 1000;
        StoppingConditionFactory.stoppingConditionType =
        StoppingConditionFactory.StoppingConditionType.GenerationCount;

        for (int i=0; i<3; i++) {
            BinCursosAG p = new BinCursosAG("ficheros/Ejercicio2DatosEntrada" + (i+1) + ".txt");

            AlgoritmoAG<List<Integer>, SolucionCursos> ap = AlgoritmoAG.of(p);
            ap.ejecuta();

            System.out.println("=====");
            System.out.println(ap.bestSolution());
            System.out.println("=====\n");
        }
    }
}
```

```

package _datos;

import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;

import us.lsi.common.Files2;

public class DatosInvestigadores {

    public static List<Investigador> investigadores;
    public static List<Trabajo> trabajos;

    public record Investigador(Integer id, Integer capacidad, Integer especialdiad) {
        public static int cont;
        public static Investigador create(String linea) {
            String[] inv = linea.split(";");
            Integer cap = Integer.parseInt(inv[0].trim().split("=")[1].trim());
            Integer esp = Integer.parseInt(inv[1].trim().split("=")[1].trim());
            return new Investigador(cont++, cap, esp);
        }
    }

    public record Trabajo(Integer id, Integer calidad, List<Integer> dias) {
        public static int cont;
        public static Trabajo create(String linea) {
            String[] trab = linea.split(";");
            Integer cal = Integer.parseInt(trab[0].trim().split("=")[1].trim());
            String[] rep = trab[1].trim().split("=")[1].trim().split(",");
            List<Integer> dias = new ArrayList<>();
            for (String s : rep) {
                s = s.replace("(", "").replace(")", "").trim();
                dias.add(Integer.parseInt(s.split(":")[1].trim()));
            }
            return new Trabajo(cont++, cal, dias);
        }
    }

    public static void iniDatos(String fichero) {

        Investigador.cont=0;
        Trabajo.cont=0;

        investigadores = new ArrayList<>();
        trabajos = new ArrayList<>();

        List<String> lineas = Files2.linesFromFile(fichero);
        List<String> invs = lineas.subList(1, lineas.indexOf("// TRABAJOS"));
        List<String> trabs = lineas.subList(lineas.indexOf("// TRABAJOS")+1, lineas.size());

        for (String i : invs) {
            investigadores.add(Investigador.create(i));
        }

        for (String t : trabs) {
            trabajos.add(Trabajo.create(t));
        }

        toConsole();
    }

    public static Integer getNumInvestigadores() {
        return investigadores.size();
    }

    public static Integer getNumEspecialidades() {
        return trabajos.get(0).dias().size();
    }
}

```

```
}

public static Integer getNumTrabajos() {
    return trabajos.size();
}

public static Integer trabajadorEspecialidad(Integer i, Integer k) {
    return investigadores.get(i).especialidad().equals(k)?
        1:
        0;
}

public static Integer diasDisponibles(Integer i) {
    return investigadores.get(i).capacidad();
}

public static Integer diasNecesarios(Integer j, Integer k) {
    return trabajos.get(j).dias().get(k);
}

public static Integer getCalidad(Integer j) {
    return trabajos.get(j).calidad();
}

public static Integer getMM() {
    return investigadores.stream()
        .map(i -> i.capacidad())
        .max(Comparator.naturalOrder())
        .get() +1;
}

private static void toConsole() {
    System.out.println(investigadores);
    System.out.println(trabajos);
}

public static void main(String[] args) {
    iniDatos("ficheros/Ejercicio3DatosEntrada1.txt");
}

}
```

```

package ejercicio3;

import java.io.IOException;
import java.util.Comparator;
import java.util.List;
import java.util.Locale;

import _datos.DatosInvestigadores;
import _datos.DatosInvestigadores.Investigador;
import _datos.DatosInvestigadores.Trabajo;
import us.lsi.gurobi.GurobiLp;
import us.lsi.gurobi.GurobiSolution;
import us.lsi.solve.AuxGrammar;

public class Ejercicio3PLE {

    public static List<Investigador> investigadores;
    public static List<Trabajo> trabajos;

    public static Integer getNumInvestigadores() {
        return investigadores.size();
    }

    public static Integer getNumEspecialidades() {
        return trabajos.get(0).dias().size();
    }

    public static Integer getNumTrabajos() {
        return trabajos.size();
    }

    public static Integer trabajadorEspecialidad(Integer i, Integer k) {
        return investigadores.get(i).especialidad().equals(k) ?
            1 :
            0;
    }

    public static Integer diasDisponibles(Integer i) {
        return investigadores.get(i).capacidad();
    }

    public static Integer diasNecesarios(Integer j, Integer k) {
        return trabajos.get(j).dias().get(k);
    }

    public static Integer getCalidad(Integer j) {
        return trabajos.get(j).calidad();
    }

    public static Integer getMM() {
        return investigadores.stream()
            .map(i -> i.capacidad())
            .max(Comparator.naturalOrder())
            .get() + 1;
    }

    public static void Ejercicio3_model() throws IOException {
        for (int i=0; i<3; i++) {

            DatosInvestigadores.iniDatos("ficheros/Ejercicio3DatosEntrada" + (i+1) + ".txt");

            investigadores = DatosInvestigadores.investigadores;
            trabajos = DatosInvestigadores.trabajos;

            AuxGrammar.generate(Ejercicio3PLE.class, "lsi_models/Ejercicio3.lsi",
"guobi_models/Ejercicio3-" + (i+1) + ".lp");

```

```
GurobiSolution solucion = GurobiLp.gurobi("gurobi_models/Ejercicio3-"+(i+1)+".lp");
Locale.setDefault(new Locale("en", "US"));
System.out.println(solucion.toString((s,d)->d>0.));

    }
}

public static void main(String[] args) throws IOException {
    Ejercicio3_model();
}

}
```

# Ejercicio3.lsi

head section

```
Integer getNumInvestigadores()  
Integer getNumEspecialidades()  
Integer getNumTrabajos()  
Integer trabajadorEspecialidad(Integer i, Integer k)  
Integer diasDisponibles(Integer i)  
Integer diasNecesarios(Integer j, Integer k)  
Integer getCalidad(Integer j)  
Integer getMM()
```

```
Integer n = getNumInvestigadores()  
Integer e = getNumEspecialidades()  
Integer m = getNumTrabajos()  
Integer MM = getMM()
```

goal section

```
// Objetivo: maximizar la calidad de los trabajos  
max sum(getCalidad(j) y[j], j in 0 .. m)
```

constraints section

```
// Para cada investigador, la suma de las horas realizadas en cada trabajo no  
puede exceder las horas disponibles de dicho investigador  
sum(x[i,j], j in 0 .. m) <= diasDisponibles(i), i in 0 .. n
```

```
// Para cada especialidad en cada trabajo, la suma de las horas realizadas  
por cada investigador con la especialidad indicada debe ser igual a las horas  
necesarias  
sum(trabajadorEspecialidad(i,k) x[i,j], i in 0 .. n) - diasNecesarios(j,k)  
y[j] = 0, j in 0 .. m, k in 0 .. e
```

```
// Para cada investigador en cada trabajo, si el trabajo j no se realiza, las  
horas realizadas en j serán 0  
x[i,j] - MM y[j] <= 0, j in 0 .. m, i in 0 .. n
```

bounds section

```
// Imponemos que y que era int solo pueda tomar los valores '0' o '1'  
y[j] <= 1, j in 0 .. m
```

```
int  
x[i,j], i in 0 .. n, j in 0 .. m  
y[j], j in 0 .. m
```

```

package _soluciones;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

import _datos.DatosInvestigadores;
import _datos.DatosInvestigadores.Investigador;

public class SolucionInvestigadores {

    public static SolucionInvestigadores of_Range(List<Integer> value) {
        return new SolucionInvestigadores(value);
    }

    private Integer calidad;
    private List<Investigador> investigadores;
    private List<List<Integer>> horas;

    private SolucionInvestigadores() {
        calidad = 0;
        investigadores = new ArrayList<>();
        horas = new ArrayList<>();
    }

    private SolucionInvestigadores(List<Integer> ls) {
        Integer numInv = DatosInvestigadores.getNumInvestigadores();
        Integer numTrab = DatosInvestigadores.getNumTrabajos();
        Integer numEsp = DatosInvestigadores.getNumEspecialidades();
        calidad = 0;
        investigadores = new ArrayList<>();
        investigadores.addAll(DatosInvestigadores.investigadores);
        horas = new ArrayList<>();
        // Añadimos una lista por cada investigador
        for (int i=0; i<numInv; i++) {
            horas.add(new ArrayList<>());
        }
        for (int j=0; j<numTrab; j++) {
            Integer jj = j*numInv;
            List<Integer> trab = ls.subList(jj, jj+numInv);
            // Añadimos a la lista i, las horas del trabajador i
            for (int i=0; i<numInv; i++) {
                horas.get(i).add(trab.get(i));
            }
            Boolean realiza=true;
            for (int k=0; k<numEsp; k++) {
                Integer suma=0;
                for (int i=0; i<numInv; i++) {
                    suma += trab.get(i)*DatosInvestigadores.trabajadorEspecialidad(i, k);
                }
                if (suma < DatosInvestigadores.diasNecesarios(j, k)) {
                    realiza = false;
                    k = numEsp;
                }
            }
        }
        // Si se realiza el trabajo, se suma su calidad
        if (realiza) {
            calidad += DatosInvestigadores.getCalidad(j);
        }
    }

    public static SolucionInvestigadores empty() {
        return new SolucionInvestigadores();
    }

    public String toString() {

```

```
String s = investigadores.stream()
    .map(i -> "INV"+(i.id()+1)+": "+horas.get(i.id()))
    .collect(Collectors.joining("\n", "Reparto de horas:\n", "\n"));
return String.format("%sSuma de las calidades de los trabajos realizados: %d", s, calidad);
}

}
```



```

package ejercicio3;

import java.util.List;

import _datos.DatosInvestigadores;
import _soluciones.SolucionInvestigadores;
import us.lsi.ag.ValuesInRangeData;
import us.lsi.ag.agchromosomes.ChromosomeFactory.ChromosomeType;

public class InRangeInvestigadoresAG
    implements ValuesInRangeData<Integer, SolucionInvestigadores> {

    public InRangeInvestigadoresAG(String fichero) {
        DatosInvestigadores.iniDatos(fichero);
    }

    @Override
    public Integer size() {
        // Deberá haber un cromosoma por cada trabajador en cada trabajo, es decir n*m
        return DatosInvestigadores.getNumInvestigadores()*DatosInvestigadores.getNumTrabajos();
    }

    @Override
    public ChromosomeType type() {
        return ChromosomeType.Range;
    }

    @Override
    public Double fitnessFunction(List<Integer> value) {
        double goal=0, error=0, kk=0, capacidad=0;
        // Definimos algunos valores por comodidad
        Integer numInv = DatosInvestigadores.getNumInvestigadores();
        Integer numTrab = DatosInvestigadores.getNumTrabajos();
        Integer numEsp = DatosInvestigadores.getNumEspecialidades();

        for (int j=0; j<numTrab; j++) {
            // Obtenemos un índice para dividir la lista de entrada en los distintos trabajos
            Integer jj = j*numInv;
            List<Integer> trab = value.subList(jj, jj+numInv);
            Boolean realiza=true;
            for (int k=0; k<numEsp; k++) {
                Integer suma=0;
                for (int i=0; i<numInv; i++) {
                    suma += trab.get(i)*DatosInvestigadores.trabajadorEspecialidad(i, k);
                }
                // Restricción de días necesarios
                if (suma != DatosInvestigadores.diasNecesarios(j, k)) {
                    realiza = false;
                    error += Math.abs(suma - DatosInvestigadores.diasNecesarios(j, k));
                }
            }
            if (realiza) {
                // Si el trabajo se realiza, sumamos su calidad
                goal += DatosInvestigadores.getCalidad(j);
            }
        }
        for (int i=0; i<numInv; i++) {
            capacidad=0;
            for (int ii=i; ii<value.size(); ii+=numInv) {
                capacidad += value.get(ii);
            }
            // Restricción de días disponibles
            if (capacidad > DatosInvestigadores.diasDisponibles(i)) {
                error += capacidad-DatosInvestigadores.diasDisponibles(i);
            }
        }
    }
}

```

```
// Cálculo de k
Integer suma=0;
for (int j=0; j<numTrab; j++) {
    suma += DatosInvestigadores.getCalidad(j);
}
kk = Math.pow(suma, 2);

return goal -kk*error;
}

@Override
public SolucionInvestigadores solucion(List<Integer> value) {
    System.out.println(value);
    return SolucionInvestigadores.of_Range(value);
}

@Override
public Integer max(Integer i) {
// Para saber el max de un i más grande que n, tomamos su valor en módulo n
    Integer l = i%DatosInvestigadores.getNumInvestigadores();
    return DatosInvestigadores.diasDisponibles(l)+1;
}

@Override
public Integer min(Integer i) {
    return 0;
}
}
```

```
package ejercicio3;

import java.util.List;
import java.util.Locale;

import _soluciones.SolucionInvestigadores;
import us.lsi.ag.agchromosomes.AlgoritmoAG;
import us.lsi.ag.agstopping.StoppingConditionFactory;

public class TestInvestigadoresAGRange {

    public static void main(String[] args) {
        Locale.setDefault(new Locale("en", "US"));

        AlgoritmoAG.ELITISM_RATE = 0.10;
        AlgoritmoAG.CROSSOVER_RATE = 0.95;
        AlgoritmoAG.MUTATION_RATE = 0.8;
        AlgoritmoAG.POPULATION_SIZE = 1000;

        StoppingConditionFactory.NUM_GENERATIONS = 1000;
        StoppingConditionFactory.stoppingConditionType =
        StoppingConditionFactory.StoppingConditionType.GenerationCount;

        for (int i=0; i<3; i++) {
            InRangeInvestigadoresAG p = new InRangeInvestigadoresAG("ficheros/Ejercicio3DatosEntrada"+
            (i+1)+".txt");

            AlgoritmoAG<List<Integer>,SolucionInvestigadores> ap = AlgoritmoAG.of(p);
            ap.ejecuta();

            System.out.println("=====");
            System.out.println(ap.bestSolution());
            System.out.println("=====\n");
        }
    }
}
```

```

package _datos;

import java.util.ArrayList;
import java.util.List;

import org.jgrapht.Graph;

import us.lsi.graphs.Graphs2;
import us.lsi.graphs.GraphsReader;
import utils.Cliente;
import utils.Conexion;

public class DatosClientes {

    @SuppressWarnings("exports")
    public static Graph<Cliente, Conexion> grafo;

    public static void iniDatos(String fichero) {
        grafo = GraphsReader.newGraph(fichero, Cliente::ofFormat, Conexion::ofFormat,
        Graphs2::simpleWeightedGraph);
        toConsole();
    }

    public static Integer getNumVertices() {
        return grafo.vertexSet().size();
    }

    @SuppressWarnings("exports")
    public static Cliente getCliente(Integer i) {
        Cliente c = null;
        List<Cliente> vertices = new ArrayList<>(grafo.vertexSet());
        for (int k = 0; k < vertices.size(); k++) {
            if (vertices.get(k).id() == i) {
                c = vertices.get(k);
            }
        }
        return c;
    }

    public static Double getBeneficio(Integer i) {
        Cliente c = getCliente(i);
        return c.beneficio();
    }

    public static Boolean existeArista(Integer i, Integer j) {
        Cliente c1 = getCliente(i);
        Cliente c2 = getCliente(j);
        return grafo.containsEdge(c1, c2);
    }

    public static Double getPeso(Integer i, Integer j) {
        Cliente c1 = getCliente(i);
        Cliente c2 = getCliente(j);
        return grafo.getEdge(c1, c2).distancia();
    }

    private static void toConsole() {
        System.out.println("Número de vértices: " + grafo.vertexSet().size() + "\n\tVértices: " +
        grafo.vertexSet()
        + "\nNúmero de aristas: " + grafo.edgeSet().size() + "\n\tAristas: " + grafo.edgeSet());
    }

    public static void main(String[] args) {
        iniDatos("ficheros/Ejercicio4DatosEntrada1.txt");
        System.out.println(getPeso(2, 4));
    }
}

```



```
package _soluciones;

import java.util.ArrayList;
import java.util.List;

import _datos.DatosClientes;
import utils.Cliente;

public class SolucionClientes {

    public static SolucionClientes of_Range(List<Integer> value) {
        return new SolucionClientes(value);
    }

    private Double kms;
    private Double benef;
    private List<Cliente> clientes;

    private SolucionClientes() {
        kms = 0.;
        benef = 0.;
        clientes = new ArrayList<>();
        Cliente c0 = DatosClientes.getCliente(0);
        clientes.add(c0);
    }

    private SolucionClientes(List<Integer> value) {
        kms = 0.;
        benef = 0.;
        clientes = new ArrayList<>();
        Cliente c0 = DatosClientes.getCliente(0);
        clientes.add(c0);
        for (int i = 0; i < value.size(); i++) {
            Cliente c = DatosClientes.getCliente(value.get(i));
            clientes.add(c);
            if (i == 0) {
                if (DatosClientes.existeArista(0, value.get(i))) {
                    kms += DatosClientes.getPeso(0, value.get(i));
                    benef += DatosClientes.getBeneficio(value.get(i)) - kms;
                }
            } else {
                if (DatosClientes.existeArista(value.get(i - 1), value.get(i))) {
                    kms += DatosClientes.getPeso(value.get(i - 1), value.get(i));
                    benef += DatosClientes.getBeneficio(value.get(i)) - kms;
                }
            }
        }
    }

    public static SolucionClientes empty() {
        return new SolucionClientes();
    }

    public String toString() {
        List<Integer> ids = clientes.stream().map(c -> c.id()).toList();
        return "Camino a seguir:\n" + ids + "\nDistancia: " + kms + "\nBeneficio: " + benef;
    }
}
```

```

package ejercicio4;

import java.util.List;

import _datos.DatosClientes;
import _soluciones.SolucionClientes;
import us.lsi.ag.SeqNormalData;
import us.lsi.ag.agchromosomes.ChromosomeFactory.ChromosomeType;

public class PermutGrafoAG implements SeqNormalData<SolucionClientes> {

    public PermutGrafoAG(String fichero) {
        DatosClientes.iniDatos(fichero);
    }

    @Override
    public ChromosomeType type() {
        return ChromosomeType.Permutation;
    }

    @Override
    public Double fitnessFunction(List<Integer> value) {
        double goal = 0, error = 0, k = 0, suma = 0;

        for (int i = 0; i < value.size(); i++) {
            if (i == 0) {
                // Suma de beneficio - penalización por "tiempo"
                // PENALIZACIÓN: (1km/min) / (1cent/min) = 1cent/km
                // ES DECIR: restamos la suma de distancias al beneficio
                if (DatosClientes.existeArista(0, value.get(i))) {
                    suma += DatosClientes.getPeso(0, value.get(i));
                    goal += DatosClientes.getBeneficio(value.get(i)) - suma;
                }
                // Penalización por no existir la arista que estudiamos
            } else {
                error++;
            }
        } else {
            if (DatosClientes.existeArista(value.get(i - 1), value.get(i))) {
                suma += DatosClientes.getPeso(value.get(i - 1), value.get(i));
                goal += DatosClientes.getBeneficio(value.get(i)) - suma;
            } else {
                error++;
            }
        }
    }

    // Doblamos la penalización (o la hacemos =2 si es 0) en el caso en el que el último vértice
    // no sea el 0
    if (value.get(value.size() - 1) != 0) {
        if (error == 0) {
            error += 2;
        } else {
            error = error * 2;
        }
    }

    suma = 0.;
    for (int i = 0; i < value.size(); i++) {
        suma += DatosClientes.getBeneficio(value.get(i));
    }
    k = Math.pow(suma, 2);

    return goal - k * error;
}

@Override

```

```
public SolucionClientes solucion(List<Integer> value) {  
    return SolucionClientes.of_Range(value);  
}  
  
@Override  
public Integer itemsNumber() {  
    return DatosClientes.getNumVertices();  
}  
  
}
```



```
package ejercicio4;

import java.util.List;
import java.util.Locale;

import _soluciones.SolucionClientes;
import us.lsi.ag.agchromosomes.AlgoritmoAG;
import us.lsi.ag.agstopping.StoppingConditionFactory;

public class TestGrafoAGPermut {

    public static void main(String[] args) {
        Locale.setDefault(new Locale("en", "US"));

        AlgoritmoAG.ELITISM_RATE = 0.10;
        AlgoritmoAG.CROSSOVER_RATE = 0.95;
        AlgoritmoAG.MUTATION_RATE = 0.8;
        AlgoritmoAG.POPULATION_SIZE = 1000;

        StoppingConditionFactory.NUM_GENERATIONS = 1000;
        StoppingConditionFactory.stoppingConditionType =
        StoppingConditionFactory.StoppingConditionType.GenerationCount;

        for (int i = 0; i < 2; i++) {
            PermutGrafoAG p = new PermutGrafoAG("ficheros/Ejercicio4DatosEntrada" + (i + 1) + ".txt");

            AlgoritmoAG<List<Integer>, SolucionClientes> ap = AlgoritmoAG.of(p);
            ap.ejecuta();

            System.out.println("=====");
            System.out.println(ap.bestSolution());
            System.out.println("=====\n");
        }
    }
}
```

```
package utils;

public record Cliente(int id, Double beneficio) {

    public static Cliente of(int id, Double beneficio) {
        return new Cliente(id, beneficio);
    }

    public static Cliente ofFormat(String[] formato) {
        Integer id = Integer.valueOf(formato[0].trim());
        Double benef = Double.valueOf(formato[1].trim());
        return of(id, benef);
    }

    @Override
    public String toString() {
        return String.valueOf(this.id());
    }
}
```

```
package utils;

public record Conexion(int id, Double distancia) {

    public static int cont;

    public static Conexion of(Double distancia) {
        Integer id = cont;
        cont++;
        return new Conexion(id, distancia);
    }

    public static Conexion ofFormat(String[] formato) {
        Double dist = Double.valueOf(formato[2].trim());
        return of(dist);
    }

    @Override
    public String toString() {
        return "id: " + this.id() + "; distancia: " + this.distancia();
    }
}
```

# Ejercicio 1

```
Java - PH_jesqher/src/ejercicio1/Ejercicio1PLE.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Problems Javadoc Declaration Console Call Hierarchy
<terminated> Ejercicio1PLE [Java Application] C:\Users\jesus.p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.2.v20220903-1139\jre\bin\javaw.exe (9 mar 2023 20:19:49 - 20:19:52) [pid: 13828]
Found heuristic solution: objective 305.00000000
Presolve removed 6 rows and 3 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.01 seconds (0.00 work units)
Thread count was 1 (of 4 available processors)

Solution count 1: 305

Optimal solution found (tolerance 1.00e-04)
Best objective 3.050000000000e+02, best bound 3.050000000000e+02, gap 0.0000%

El valor objetivo es 305.00
Los valores de la variables
x_0 == 10
x_1 == 10
x_2 == 1

Kgs disponibles de cada tipo: [11, 9, 7, 12, 6]
Variedades disponibles: [Variedad[id=0, beneficio=20, composicion=[0.2, 0.4, 0.0, 0.0, 0.4]], Variedad[id=1, beneficio=10, composicion=[0.0, 0.3, 0.7, 0.0, 0.0]], Variedad[id=2, benefici
(getNumVariedades,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getNumVariedades())
(getKgTipoVariedad,public static java.lang.Double ejercicio1.Ejercicio1PLE.getKgTipoVariedad(java.lang.Integer,java.lang.Integer))
(getBeneficioVariedad,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getBeneficioVariedad(java.lang.Integer))
(getKgTipo,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getKgTipo(java.lang.Integer))
(getNumTipos,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getNumTipos())
(getNumVariedades,{}),(getKgTipoVariedad,{INT,INT}),(getBeneficioVariedad,{INT}),(getKgTipo,{INT}),(getNumTipos,{})
(1,2),(3,5),(4,3),(5,3),(n,5)

Tenga en cuenta que el formato intermedio LP no distingue entre desigualdades estrictas y no estrictas en las restricciones
Por lo que, por ejemplo, < y <= son equivalentes.

Set parameter Username

Java - PH_jesqher/src/ejercicio1/Ejercicio1PLE.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Problems Javadoc Declaration Console Call Hierarchy
<terminated> Ejercicio1PLE [Java Application] C:\Users\jesus.p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.2.v20220903-1139\jre\bin\javaw.exe (9 mar 2023 20:19:49 - 20:19:52) [pid: 13828]
RHS range [6e+00, 1e+01]
Found heuristic solution: objective 2000.00000000
Presolve removed 5 rows and 3 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.00 seconds (0.00 work units)
Thread count was 1 (of 4 available processors)

Solution count 1: 2000

Optimal solution found (tolerance 1.00e-04)
Best objective 2.000000000000e+03, best bound 2.000000000000e+03, gap 0.0000%

El valor objetivo es 2000.00
Los valores de la variables
x_0 == 15
x_1 == 10
x_2 == 20

Kgs disponibles de cada tipo: [35, 4, 12, 5, 30, 42, 3, 2, 20, 3]
Variedades disponibles: [Variedad[id=0, beneficio=60, composicion=[0.5, 0.0, 0.4, 0.0, 0.0, 0.0, 0.1, 0.0, 0.0, 0.0]], Variedad[id=1, beneficio=25, composicion=[0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]], Variedad[id=2, beneficio=10, composicion=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]], Variedad[id=3, beneficio=10, composicion=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]], Variedad[id=4, beneficio=10, composicion=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]], Variedad[id=5, beneficio=10, composicion=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]], Variedad[id=6, beneficio=10, composicion=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]], Variedad[id=7, beneficio=10, composicion=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]], Variedad[id=8, beneficio=10, composicion=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]], Variedad[id=9, beneficio=10, composicion=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]]
(getNumVariedades,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getNumVariedades())
(getKgTipoVariedad,public static java.lang.Double ejercicio1.Ejercicio1PLE.getKgTipoVariedad(java.lang.Integer,java.lang.Integer))
(getBeneficioVariedad,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getBeneficioVariedad(java.lang.Integer))
(getKgTipo,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getKgTipo(java.lang.Integer))
(getNumTipos,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getNumTipos())
(getNumVariedades,{}),(getKgTipoVariedad,{INT,INT}),(getBeneficioVariedad,{INT}),(getKgTipo,{INT}),(getNumTipos,{})
(1,2),(3,4),(4,3),(5,3),(n,10)

Tenga en cuenta que el formato intermedio LP no distingue entre desigualdades estrictas y no estrictas en las restricciones
Por lo que, por ejemplo, < y <= son equivalentes.
```

```
Java - PI4_jesusqher/src/ejercicio1/Ejercicio1PLE.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Problems Javadoc Declaration Console Call Hierarchy
terminated: Ejercicio1PLE [Java Application] C:\Users\jesus.p2\pooth.plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.2.v20220903-1139\jre\bin\javaw.exe (9 mar 2023 20:19:49 - 20:19:52) [pid: 13828]

RHS range [6e+00, 1e+01]
Found heuristic solution: objective 2000.0000000
Presolve removed 5 rows and 3 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.00 seconds (0.00 work units)
Thread count was 1 (of 4 available processors)

Solution count 1: 2000

Optimal solution found (tolerance 1.00e-04)
Best objective 2.000000000000e+03, best bound 2.000000000000e+03, gap 0.00000%

El valor objetivo es 2000.00
Los valores de la variables
x_0 == 15
x_1 == 10
x_2 == 20

Kgs disponibles de cada tipo: [35, 4, 12, 5, 30, 42, 3, 2, 20, 3]
Variedades disponibles: [Variedad[id=0, beneficio=60, composicion=[0.5, 0.0, 0.4, 0.0, 0.0, 0.0, 0.1, 0.0, 0.0, 0.0]], Variedad[id=1, beneficio=25, composicion=[0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]], Variedad[id=2, beneficio=10, composicion=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]]]
(getNumVariedades,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getNumVariedades())
(getKgTipoVariedad,public static java.lang.Double ejercicio1.Ejercicio1PLE.getKgTipoVariedad(java.lang.Integer,java.lang.Integer))
(getBeneficioVariedad,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getBeneficioVariedad(java.lang.Integer))
(getKgTipo,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getKgTipo(java.lang.Integer))
(getNumTipos,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getNumTipos())
(getNumVariedades,{}),(getKgTipoVariedad,(INT,INT))),(getBeneficioVariedad,(INT))),(getKgTipo,(INT))),(getNumTipos,{}))
(1,2),(3,4),(5,6),(7,8),(9,10)

=====
Tenga en cuenta que el formato intermedio LP no distingue entre desigualdades estrictas y no estrictas en las restricciones
Por lo que, por ejemplo, < y <= son equivalentes.
=====
```

```
Java - PI4_jesusqher/src/ejercicio1/TestCafeAGRange.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Problems Javadoc Declaration Console Call Hierarchy
terminated: TestCafeAGRange [Java Application] C:\Users\jesus.p2\pooth.plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.2.v20220903-1139\jre\bin\javaw.exe (9 mar 2023 20:22:14 - 20:22:35) [pid: 5592]

=====
Kgs disponibles de cada tipo: [5, 4, 1, 2, 8, 1]
Variedades disponibles: [Variedad[id=0, beneficio=20, composicion=[0.5, 0.4, 0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]], Variedad[id=1, beneficio=10, composicion=[0.0, 0.0, 0.0, 0.2, 0.8, 0.0, 0.0, 0.0, 0.0, 0.0]], Variedad[id=2, beneficio=5, composicion=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]]]
(getNumVariedades,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getNumVariedades())
(getKgTipoVariedad,public static java.lang.Double ejercicio1.Ejercicio1PLE.getKgTipoVariedad(java.lang.Integer,java.lang.Integer))
(getBeneficioVariedad,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getBeneficioVariedad(java.lang.Integer))
(getKgTipo,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getKgTipo(java.lang.Integer))
(getNumTipos,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getNumTipos())
(getNumVariedades,{}),(getKgTipoVariedad,(INT,INT))),(getBeneficioVariedad,(INT))),(getKgTipo,(INT))),(getNumTipos,{}))
(1,2),(3,4),(5,6),(7,8),(9,10)

=====
Kgs producidos de cada variedad:
P1: 10 Kg
P2: 10 Kg
P3: 1 Kg
Beneficio obtenido: 305.0
=====

Kgs disponibles de cada tipo: [11, 9, 7, 12, 6]
Variedades disponibles: [Variedad[id=0, beneficio=20, composicion=[0.2, 0.4, 0.0, 0.0, 0.0, 0.4]], Variedad[id=1, beneficio=10, composicion=[0.0, 0.3, 0.7, 0.0, 0.0, 0.0]], Variedad[id=2, beneficio=5, composicion=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]]]
(getNumVariedades,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getNumVariedades())
(getKgTipoVariedad,public static java.lang.Double ejercicio1.Ejercicio1PLE.getKgTipoVariedad(java.lang.Integer,java.lang.Integer))
(getBeneficioVariedad,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getBeneficioVariedad(java.lang.Integer))
(getKgTipo,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getKgTipo(java.lang.Integer))
(getNumTipos,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getNumTipos())
(getNumVariedades,{}),(getKgTipoVariedad,(INT,INT))),(getBeneficioVariedad,(INT))),(getKgTipo,(INT))),(getNumTipos,{}))
(1,2),(3,4),(5,6),(7,8),(9,10)

=====
Kgs producidos de cada variedad:
P1: 15 Kg
P2: 10 Kg
P3: 20 Kg
Beneficio obtenido: 2000.0
=====

Kgs disponibles de cada tipo: [35, 4, 12, 5, 30, 42, 3, 2, 20, 3]
Variedades disponibles: [Variedad[id=0, beneficio=60, composicion=[0.5, 0.0, 0.4, 0.0, 0.0, 0.0, 0.0, 0.1, 0.0, 0.0, 0.0]], Variedad[id=1, beneficio=25, composicion=[0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]], Variedad[id=2, beneficio=10, composicion=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]]]
(getNumVariedades,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getNumVariedades())
(getKgTipoVariedad,public static java.lang.Double ejercicio1.Ejercicio1PLE.getKgTipoVariedad(java.lang.Integer,java.lang.Integer))
(getBeneficioVariedad,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getBeneficioVariedad(java.lang.Integer))
(getKgTipo,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getKgTipo(java.lang.Integer))
(getNumTipos,public static java.lang.Integer ejercicio1.Ejercicio1PLE.getNumTipos())
(getNumVariedades,{}),(getKgTipoVariedad,(INT,INT))),(getBeneficioVariedad,(INT))),(getKgTipo,(INT))),(getNumTipos,{}))
(1,2),(3,4),(5,6),(7,8),(9,10)

=====
Kgs producidos de cada variedad:
P1: 30 Kg
P2: 4 Kg
P3: 15 Kg
P4: 100 Kg
Beneficio obtenido: 12275.0
=====
```

## Ejercicio 2

```
Java - PID_jeseshqher/src/Ejercicio2/Ejercicio2PLE.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

-terminated: Ejercicio2PLE [Java Application] C:\Users\jesus.p2\poo\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.2\20220903-1139\re\bin\java.exe (9 mar 2023 20:23:36 - 20:23:38) [pid: 8800]

Bounds range [1e+00, 1e+00]
RHS range [1e+00, 1e+00]
Presolve removed 14 rows and 6 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.00 seconds (0.00 work units)
Thread count was 1 (of 4 available processors)

Solution count 1: 15

Optimal solution found (tolerance 1.00e-04)
Best objective 1.500000000000e+01, best bound 1.500000000000e+01, gap 0.0000%

El valor objetivo es 15.00
Los valores de la variables
x_0 == 1
x_3 == 1
y_0 == 1

*****
Numero maximo de centros a seleccionar: 2
Cursos disponibles: {Curso{id=0, tematicas={2, 3}, precio=2.0, centro=0}, Curso{id=1, tematicas={4}, precio=3.0, centro=0}, Curso{id=2, tematicas={1, 5}, precio=5.0, centro=0}, Curso{id=3, tematicas={6, 7}, precio=2.0, centro=2}}
(getMaxCentros,public static java.lang.Integer Ejercicio2PLE.getMaxCentros())
(getNumTematicas,public static java.lang.Integer Ejercicio2PLE.getNumTematicas())
(ofreceCurso,public static java.lang.Integer Ejercicio2PLE.ofreceCurso(java.lang.Integer,java.lang.Integer))
(getNumCursos,public static java.lang.Integer Ejercicio2PLE.getNumCursos())
(getNumCentros,public static java.lang.Integer Ejercicio2PLE.getNumCentros())
(getPrecioCurso,public static java.lang.Double Ejercicio2PLE.getPrecioCurso(java.lang.Integer))
(contieneTematica,public static java.lang.Integer Ejercicio2PLE.contieneTematica(java.lang.Integer,java.lang.Integer))
(getMaxCentros,{}), (getNumTematicas,{}), (ofreceCurso,{INT,INT}), (getNumCursos,{}), (getNumCentros,{}), (getPrecioCurso,{INT}), (contieneTematica,{INT,INT})
(nc,3), (maxCentros,2), (i,3), (j,4), (k,1), (m,5), (n,5)

*****
Tenga en cuenta que el formato intermedio LP no distingue entre desigualdades estrictas y no estrictas en las restricciones
Por lo que, por ejemplo, < y <= son equivalentes.

Java - PID_jeseshqher/src/Ejercicio2/Ejercicio2PLE.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

-terminated: Ejercicio2PLE [Java Application] C:\Users\jesus.p2\poo\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.2\20220903-1139\re\bin\java.exe (9 mar 2023 20:23:36 - 20:23:38) [pid: 8800]

Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.03 seconds (0.00 work units)
Thread count was 1 (of 4 available processors)

Solution count 1: 8.5

Optimal solution found (tolerance 1.00e-04)
Best objective 8.500000000000e+00, best bound 8.500000000000e+00, gap 0.0000%

El valor objetivo es 8.50
Los valores de la variables
x_0 == 1
x_2 == 1
x_4 == 1
y_0 == 1
y_1 == 1

*****
Numero maximo de centros a seleccionar: 3
Cursos disponibles: {Curso{id=0, tematicas={2, 6, 7}, precio=2.0, centro=2}, Curso{id=1, tematicas={7}, precio=3.0, centro=0}, Curso{id=2, tematicas={1, 5}, precio=5.0, centro=0}, Curso{id=3, tematicas={4}, precio=2.0, centro=2}}
(getMaxCentros,public static java.lang.Integer Ejercicio2PLE.getMaxCentros())
(getNumTematicas,public static java.lang.Integer Ejercicio2PLE.getNumTematicas())
(ofreceCurso,public static java.lang.Integer Ejercicio2PLE.ofreceCurso(java.lang.Integer,java.lang.Integer))
(getNumCursos,public static java.lang.Integer Ejercicio2PLE.getNumCursos())
(getNumCentros,public static java.lang.Integer Ejercicio2PLE.getNumCentros())
(getPrecioCurso,public static java.lang.Double Ejercicio2PLE.getPrecioCurso(java.lang.Integer))
(contieneTematica,public static java.lang.Integer Ejercicio2PLE.contieneTematica(java.lang.Integer,java.lang.Integer))
(getMaxCentros,{}), (getNumTematicas,{}), (ofreceCurso,{INT,INT}), (getNumCursos,{}), (getNumCentros,{}), (getPrecioCurso,{INT}), (contieneTematica,{INT,INT})
(nc,3), (maxCentros,3), (i,4), (j,4), (k,2), (m,7), (n,8)

*****
Tenga en cuenta que el formato intermedio LP no distingue entre desigualdades estrictas y no estrictas en las restricciones
Por lo que, por ejemplo, < y <= son equivalentes.
```

```
Java - PI4_jesqher/src/ejercicio2/Ejercicio2PLE.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console X Call Hierarchy
terminated: Ejercicio2PLE [Java Application] C:\Users\jesus.p2\poo\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.2.v20220903-1139\jre\bin\javaw.exe (9 mar 2023 20:23:36 - 20:23:36) [pid: 8800]
T: 32 rows, 11 columns, 54 nonzeros
Gurobi Optimizer version 9.5.0 build v9.5.0rc5 (win64)
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads
Optimize a model with 32 rows, 11 columns and 54 nonzeros
Model fingerprint: 0x00e94806
Variable types: 0 continuous, 11 integer (11 binary)
Coefficient statistics:
  Matrix range [1e+00, 1e+00]
  Objective range [1e+00, 6e+00]
  Bounds range [1e+00, 1e+00]
  RHS range [1e+00, 3e+00]
Found heuristic solution: objective 12.00000000
Found heuristic solution: objective 6.50000000
Presolve removed 32 rows and 11 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.00 seconds (0.00 work units)
Thread count was 1 (of 4 available processors)

Solution count 2: 6.5 12

Optimal solution found (tolerance 1.00e-04)
Best objective 6.500000000000e+00, best bound 6.500000000000e+00, gap 0.0000%

El valor objetivo es 6.50
Los valores de la variables
x_0 == 1
x_3 == 1
x_7 == 1
y_0 == 1
y_1 == 1
y_2 == 1
```

```
Java - PI4_jesqher/src/ejercicio2/TestCursosAGBin.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console X Call Hierarchy
terminated: TestCursosAGBin [Java Application] C:\Users\jesus.p2\poo\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.2.v20220903-1139\jre\bin\javaw.exe (9 mar 2023 20:24:38 - 20:24:40) [pid: 11540]
Número máximo de centros a seleccionar: 1
Cursos disponibles: [Curso[id=0, tematicas=[1, 2, 3, 4], precio=10.0, centro=0], Curso[id=1, tematicas=[1, 4], precio=3.0, centro=0], Curso[id=2, tematicas=[5], precio=1.5, centro=1], C
=====
Cursos seleccionados: {S0, S3}
Coste total: 15.0
=====

Número máximo de centros a seleccionar: 2
Cursos disponibles: [Curso[id=0, tematicas=[2, 3], precio=2.0, centro=0], Curso[id=1, tematicas=[4], precio=3.0, centro=0], Curso[id=2, tematicas=[1, 5], precio=5.0, centro=0], Curso[id
=====
Cursos seleccionados: {S0, S2, S4}
Coste total: 8.5
=====

Número máximo de centros a seleccionar: 3
Cursos disponibles: [Curso[id=0, tematicas=[2, 6, 7], precio=2.0, centro=2], Curso[id=1, tematicas=[7], precio=3.0, centro=0], Curso[id=2, tematicas=[1, 5], precio=5.0, centro=0], Curso
=====
Cursos seleccionados: {S0, S3, S7}
Coste total: 6.5
=====

|
```

## Ejercicio 3

```
Java - PH_Jesúsgher/src/ejercicio3/Ejercicio3PLE.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Problems Javadoc Declaration Console Call Hierarchy
<terminated> Ejercicio3PLE [Java Application] C:\Users\jesus.p2\poo\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.2.v20220903-1139\jre\bin\javaw.exe (9 mar 2023 20:26:05 - 20:26:07) [pid: 5080]
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.00 seconds (0.00 work units)
Thread count was 1 (of 4 available processors)

Solution count 2: 15 -0

Optimal solution found (tolerance 1.00e-04)
Best objective 1.500000000000e+01, best bound 1.500000000000e+01, gap 0.0000%

El valor objetivo es 15.00
Los valores de la variables
x_0_0 == 6
x_1_1 == 0
x_2_1 == 8
y_0 == 1
y_1 == 1

[Investigador[id=0, capacidad=10, especialidad=0], Investigador[id=1, capacidad=5, especialidad=1], Investigador[id=2, capacidad=8, especialidad=2], Investigador[id=3, capacidad=2, espe
[Trabajo[id=0, calidad=7, dias=[2, 0, 5, 0]], Trabajo[id=1, calidad=9, dias=[8, 4, 3, 0]], Trabajo[id=2, calidad=5, dias=[2, 0, 0, 7]]]
(getNumEspecialidades,public static java.lang.Integer ejercicio3.Ejercicio3PLE.getNumEspecialidades())
(trabajadorEspecialidad,public static java.lang.Integer ejercicio3.Ejercicio3PLE.trabajadorEspecialidad(java.lang.Integer,java.lang.Integer))
(diasDisponibles,public static java.lang.Integer ejercicio3.Ejercicio3PLE.diasDisponibles(java.lang.Integer))
(diasNecesarios,public static java.lang.Integer ejercicio3.Ejercicio3PLE.diasNecesarios(java.lang.Integer,java.lang.Integer))
(getCalidad,public static java.lang.Integer ejercicio3.Ejercicio3PLE.getCalidad(java.lang.Integer))
(getNumTrabajos,public static java.lang.Integer ejercicio3.Ejercicio3PLE.getNumTrabajos())
(getMW,public static java.lang.Integer ejercicio3.Ejercicio3PLE.getMW())
(getNumInvestigadores,public static java.lang.Integer ejercicio3.Ejercicio3PLE.getNumInvestigadores())
(getNumEspecialidades,{}),(trabajadorEspecialidad,{INT,INT}),(diasDisponibles,{INT}),(diasNecesarios,{INT,INT}),(getCalidad,{INT}),(getNumTrabajos,{}),(getMW,{}),(getNumInvestigadores,{
(MW,11),(e,4),(i,2),(j,2),(k,2),(m,3),(n,5)

Tenga en cuenta que el formato Intermedio LP no distingue entre desigualdades estrictas y no estrictas en las restricciones
Por lo que, por ejemplo, < y <= son equivalentes.
<

Java - PH_Jesúsgher/src/ejercicio3/Ejercicio3PLE.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Problems Javadoc Declaration Console Call Hierarchy
<terminated> Ejercicio3PLE [Java Application] C:\Users\jesus.p2\poo\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.2.v20220903-1139\jre\bin\javaw.exe (9 mar 2023 20:26:05 - 20:26:07) [pid: 5080]
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.00 seconds (0.00 work units)
Thread count was 1 (of 4 available processors)

Solution count 2: 16 -0

Optimal solution found (tolerance 1.00e-04)
Best objective 1.600000000000e+01, best bound 1.600000000000e+01, gap 0.0000%

El valor objetivo es 16.00
Los valores de la variables
x_0_0 == 2
x_0_1 == 8
x_1_1 == 4
x_2_0 == 5
x_2_1 == 8
y_0 == 1
y_1 == 1

[Investigador[id=0, capacidad=1, especialidad=2], Investigador[id=1, capacidad=10, especialidad=1], Investigador[id=2, capacidad=3, especialidad=0], Investigador[id=3, capacidad=4, espe
[Trabajo[id=0, calidad=8, dias=[2, 0, 2, 0]], Trabajo[id=1, calidad=5, dias=[8, 5, 4, 2]], Trabajo[id=2, calidad=8, dias=[0, 5, 0, 15]], Trabajo[id=3, calidad=5, dias=[0, 7, 8, 5]], Tra
(getNumEspecialidades,public static java.lang.Integer ejercicio3.Ejercicio3PLE.getNumEspecialidades())
(trabajadorEspecialidad,public static java.lang.Integer ejercicio3.Ejercicio3PLE.trabajadorEspecialidad(java.lang.Integer,java.lang.Integer))
(diasDisponibles,public static java.lang.Integer ejercicio3.Ejercicio3PLE.diasDisponibles(java.lang.Integer))
(diasNecesarios,public static java.lang.Integer ejercicio3.Ejercicio3PLE.diasNecesarios(java.lang.Integer,java.lang.Integer))
(getCalidad,public static java.lang.Integer ejercicio3.Ejercicio3PLE.getCalidad(java.lang.Integer))
(getNumTrabajos,public static java.lang.Integer ejercicio3.Ejercicio3PLE.getNumTrabajos())
(getMW,public static java.lang.Integer ejercicio3.Ejercicio3PLE.getMW())
(getNumInvestigadores,public static java.lang.Integer ejercicio3.Ejercicio3PLE.getNumInvestigadores())
(getNumEspecialidades,{}),(trabajadorEspecialidad,{INT,INT}),(diasDisponibles,{INT}),(diasNecesarios,{INT,INT}),(getCalidad,{INT}),(getNumTrabajos,{}),(getMW,{}),(getNumInvestigadores,{
(MW,31),(e,4),(i,4),(j,2),(k,3),(m,5),(n,8)

Tenga en cuenta que el formato Intermedio LP no distingue entre desigualdades estrictas y no estrictas en las restricciones
Por lo que, por ejemplo, < y <= son equivalentes.
<
```



```
Java - PI4_jesusqher/src/ ejercicio3/Ejercicio3PLE.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console Call Hierarchy
terminated: Ejercicio3PLE [Java Application] C:\Users\jesus\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.2.v20220903-1139\jre\bin\javaw.exe (9 mar 2023 20:26:03 - 20:26:07) [pid: 5080]
Presolved: 6 rows, 5 columns, 14 moneros
Found heuristic solution: objective 16.00000000
Variable types: 0 continuous, 5 integer (2 binary)

Root relaxation: objective 2.500000e+01, 1 iterations, 0.00 seconds (0.00 work units)

Nodes | Current Node | Objective Bounds | Work
Expl Unexpl | Obj Depth IntInf | Incumbent BestBd Gap | It/Node Time
* 0 0 0 25.00000000 25.00000 0.00% - 0s

Explored 1 nodes (1 simplex iterations) in 0.00 seconds (0.00 work units)
Thread count was 4 (of 4 available processors)

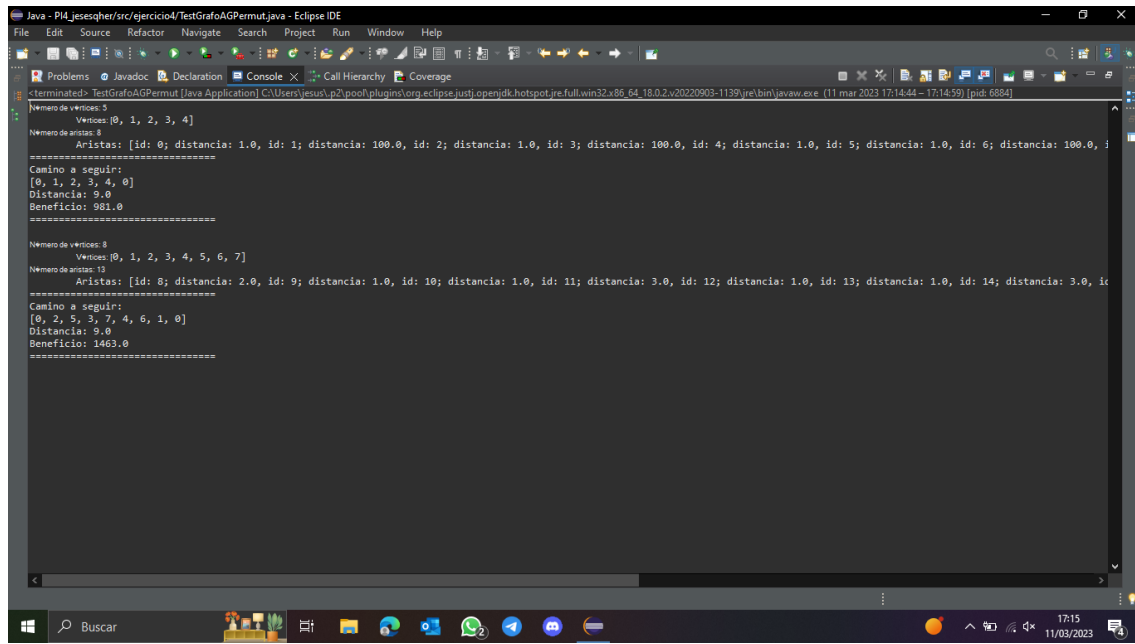
Solution count 3: 25 16 -0

Optimal solution found (tolerance 1.00e-04)
Best objective 2.500000000000e+01, best bound 2.500000000000e+01, gap 0.0000%

El valor objetivo es 25.00
Los valores de la variables
x_0_0 == 1
x_1_2 == 5
x_1_4 == 5
x_2_4 == 3
x_3_0 == 2
x_3_4 == 2
x_6_0 == 1
x_7_2 == 15
x_7_4 == 2
y_0 == 1
y_2 == 1
y_4 == 1
```

```
Java - PI4_jesusqher/src/ ejercicio3/TestInvestigadoresAGRange.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console Call Hierarchy
terminated: TestInvestigadoresAGRange [Java Application] C:\Users\jesus\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.2.v20220903-1139\jre\bin\javaw.exe (9 mar 2023 20:29:30 - 20:29:58) [pid: 2348]
=====
[6, 0, 0, 0, 3, 8]
Reparto de horas:
INW1: [6, 0]
INW2: [0, 3]
INW3: [0, 0]
Suma de las calidades de los trabajos realizados: 15
=====
[Investigador[id=0, capacidad=10, especialdiad=0], Investigador[id=1, capacidad=5, especialdiad=1], Investigador[id=2, capacidad=8, especialdiad=2], Investigador[id=3, capacidad=2, especialdiad=0], Investigador[id=4, capacidad=10, especialdiad=0], Investigador[id=5, capacidad=5, especialdiad=1], Investigador[id=6, capacidad=8, especialdiad=2], Investigador[id=7, capacidad=2, especialdiad=0], Investigador[id=8, capacidad=10, especialdiad=0], Investigador[id=9, capacidad=5, especialdiad=1], Investigador[id=10, capacidad=8, especialdiad=2], Investigador[id=11, capacidad=2, especialdiad=0], Investigador[id=12, capacidad=10, especialdiad=0], Investigador[id=13, capacidad=5, especialdiad=1], Investigador[id=14, capacidad=8, especialdiad=2], Investigador[id=15, capacidad=2, especialdiad=0]]
[Trabajo[id=0, calidad=7, dias=[2, 0, 5, 0]], Trabajo[id=1, calidad=9, dias=[0, 4, 3, 0]], Trabajo[id=2, calidad=5, dias=[2, 0, 0, 7]]]
=====
[1, 0, 5, 1, 0, 7, 4, 3, 1, 0, 2, 0, 0, 0, 5]
Reparto de horas:
INW1: [1, 7, 2]
INW2: [0, 4, 0]
INW3: [5, 3, 0]
INW4: [1, 1, 0]
INW5: [0, 0, 5]
Suma de las calidades de los trabajos realizados: 16
=====
[Investigador[id=0, capacidad=1, especialdiad=2], Investigador[id=1, capacidad=10, especialdiad=1], Investigador[id=2, capacidad=3, especialdiad=0], Investigador[id=3, capacidad=4, especialdiad=0], Investigador[id=4, capacidad=10, especialdiad=0], Investigador[id=5, capacidad=5, especialdiad=1], Investigador[id=6, capacidad=8, especialdiad=2], Investigador[id=7, capacidad=2, especialdiad=0], Investigador[id=8, capacidad=10, especialdiad=0], Investigador[id=9, capacidad=5, especialdiad=1], Investigador[id=10, capacidad=8, especialdiad=2], Investigador[id=11, capacidad=2, especialdiad=0], Investigador[id=12, capacidad=10, especialdiad=0], Investigador[id=13, capacidad=5, especialdiad=1], Investigador[id=14, capacidad=8, especialdiad=2], Investigador[id=15, capacidad=2, especialdiad=0]]
[1, 0, 2, 0, 0, 0, 1, 0, 0, 3, 2, 3, 0, 0, 0, 2, 0, 5, 0, 0, 0, 1, 0, 5, 0, 3, 0, 0, 0, 1, 0, 4, 0, 5, 3, 2, 1, 0, 0, 1]
Reparto de horas:
INW1: [1, 0, 0, 0, 0]
INW2: [0, 3, 5, 3, 5]
INW3: [2, 2, 0, 0, 3]
INW4: [0, 3, 0, 0, 2]
INW5: [0, 0, 0, 0, 1]
INW6: [0, 0, 1, 1, 0]
INW7: [1, 0, 0, 0, 0]
INW8: [0, 2, 5, 4, 1]
Suma de las calidades de los trabajos realizados: 25
=====
```

## Ejercicio 4



The screenshot shows the Eclipse IDE interface with a Java application running. The console output displays the results of a graph algorithm, including the number of vertices, edges, and the shortest path found for two different graphs.

```
Java - PH_jesegher/src/ejercicio4/TestGrafoGPermut.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

<terminated> TestGrafoGPermut [Java Application] C:\Users\jesus\p2\pooa\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.2.v20220903-1139\jre\bin\javaw.exe (11 mar 2023 17:14:44 - 17:14:59) [pid: 6884]

Numero de vertices: 5
  Vertices: [0, 1, 2, 3, 4]
Numero de aristas: 8
  Aristas: [id: 0; distancia: 1.0, id: 1; distancia: 100.0, id: 2; distancia: 1.0, id: 3; distancia: 100.0, id: 4; distancia: 1.0, id: 5; distancia: 1.0, id: 6; distancia: 100.0, id: 7]
Camino a seguir:
[0, 1, 2, 3, 4, 0]
Distancia: 9.0
Beneficio: 981.0
=====

Numero de vertices: 8
  Vertices: [0, 1, 2, 3, 4, 5, 6, 7]
Numero de aristas: 13
  Aristas: [id: 8; distancia: 2.0, id: 9; distancia: 1.0, id: 10; distancia: 1.0, id: 11; distancia: 3.0, id: 12; distancia: 1.0, id: 13; distancia: 1.0, id: 14; distancia: 3.0, id: 15]
Camino a seguir:
[0, 2, 5, 3, 7, 4, 6, 1, 0]
Distancia: 9.0
Beneficio: 1463.0
=====
```