- MOTXILA ZATIEKIN PROBLEMA

Prozedura-hauteslea

Objektu (hautagai) guztietatik $\max\left\{\frac{balioa}{pisua}\right\}$ proportzio handiena daukana hartu.

Osogarria

Sartzen den bitartea, osorik edo proportzio bat, baina edukiera gainditu gabe.

SoluzioaDa?

Soluzio partziala beti izango da soluzioa (motxila betez goaz).

Kodea

```
public class MotxilaZatika {

      public double motxilaBeteZatiekin(LinkedList<Objektua>
objektuak, double pisuMax, LinkedList<Objektua> emaitza){
            emaitza.clear();
            double balioMax = 0;
            LinkedList<Objektua> objKopia = (LinkedList<Objektua>)
objektuak.clone();
            proportzioakKalkulatu(objKopia);
            Collections.sort(objKopia, new Comparator<Objektua>() {
                  public int compare(Objektua o1, Objektua o2) {
                        if(o1.getProportzioa()>o2.getProportzioa()){
                              return -1;
                        }else
if(o1.getProportzioa()==o2.getProportzioa()){
                              return 0;
                        }else{
                              return 1;
                        }
                  }
            }); /* Ordenaziorako MergeSort erabiltzen da */
            double geratzenDenPisua = pisuMax;
            while (geratzenDenPisua>0 && !objKopia.isEmpty()){
                  Objektua obj = objKopia.removeFirst();
                  if(obj.getPisua()<=geratzenDenPisua){
                        obj.setZenbatEraman(1);
                        geratzenDenPisua = geratzenDenPisua -
obj.getPisua();
                        balioMax =  balioMax + obj.getBalioa();
                  }else{

      obj.setZenbatEraman(geratzenDenPisua/obj.getPisua());
                        balioMax =  balioMax +
(obj.getProportzioa()*geratzenDenPisua);
                        geratzenDenPisua = 0;
                  }
                  emaitza.add(obj);
            }
            return balioMax;
      }
```

```
	private void proportzioakKalkulatu(LinkedList<Objektua>
objektuak) {
		for(Objektua obj : objektuak){
			obj.setProportzioa(obj.getBalioa()/obj.getPisua());
		}
	}
}
```

```
public class Objektua{
	private int pisua;
	private int balioa;
	private double proportzioa;
	private double zenbatEraman;

	//Getterrak setterrak eta método eraikitzailea
}
```

```
public class Frogak {

	public static void main(String[] args) {
		LinkedList<Objektua> objektuak = new
LinkedList<Objektua>();
		objektuak.add(new Objektua(10, 60));
		objektuak.add(new Objektua(20, 100));
		objektuak.add(new Objektua(30, 120));
		LinkedList<Objektua> emaitza = new LinkedList<Objektua>();
		MotxilaZatika mz = new MotxilaZatika();
		System.out.println(mz.motxilaBeteZatiekin(objektuak, 50,
emaitza));
		System.out.println(emaitza.toString());
	}

}
```

Analisia

N = Eskuragarri ditugun objektu kopurua

1. Proportzioak kalkulatu objektu guztientzat: O(N).
2. Objektuak ordenatu kalkulatutako proportzioaren arabera (orden beherakorrean). MergeSort erabiltzen da ordenazioa egiteko, beraz: O(N lgN).
3. Kasu txarrenean begiztan objektu guztiak tratatuko dira, eta horregatik, kostua lineala izango da objektu kopuruan: O(N).

Baturaren erregela aplikatuz: $N + N \lg N + N \in O(N \lg N)$