

**Universidad Nacional Mayor de San Marcos**

(Universidad del Perú, DECANA DE AMÉRICA)

**Facultad de Ingeniería de Sistemas e Informática**

**Escuela de Ingeniería de Software**

# **Laboratorio 10. Algoritmos de Retroceso**

**Asignatura:** Análisis y Diseño de Algoritmos

**Docente:** Robert Elías Espinoza Domínguez

**Semestre:** 2020 - I

**Grupo: 4**

**Integrantes:**

* Calderon Saldaña, Hilton Eduardo 15200202
* Cifuentes Michuy, Daniel Ernesto 18200125
* Ramos Villarreal, Edward Junior 18200288
* Rodríguez Tocas, Miguel Ángel 18200290
* Salcedo Alfaro, John Marco 18200101
* Torres Briones, Diego Leonardo 18200322
* Zambrano Quispe, Arian Alberto 18200107

*11 de setiembre de 2020*

**1.**Modifica la aplicación para que muestre un mensaje en caso no encuentre ningún subconjunto que sume exactamente un valor dado y además muestre todas las soluciones, si hubieran.

**Metodo** Backtracking (Entero valores [], Entero suma)

v <- Entero [valores.length]

numSubSets <- ( Double (2^valores.length)).intValue ()

s <-Entero [numSubSets][valores.length]

v <-valores

tact <- 0

p <-suma

Solucion<- 0

**Fin Metodo**

**Metodo hallarSubconjunto()**

Entero nivel <- 0

**Para i** <- 0 hasta numSubSets **hacer**

**Para** j <- 0 hasta s[i].length hacer

s[i][j] <- -1

**FinPara**

**FinPara**

printS()

**Hacer**

generar(nivel)

printS()

**Si** solucion(nivel, p) **entonces**

solucion<- Solucion + 1

**FinSi**

**Si** criterio(nivel, p) **entonces**

nivel<-nivel + 1

**Finsi**

**Sino**

**Mientras**(!masHermanos(nivel)) **hacer** nivel<-retroceder(nivel)

**Si** (nivel < 0) **entonces**

retornar

**Finsi**

**FinMientras**

**Mientras (verdadero)**

**Fin Metodo**

**Metodo generar (Entero nivel)**

s[Solucion ][nivel] <- s[Solucion][nivel] + 1

**Si** (s[Solucion][nivel] == 1) **entonces**

tact <- tact + v[nivel]

**Fin Si**

**FinMetodo**

**Metodo booleano** solucion(Enteor nivel, Entero p)

Entero n <- s[0].length

retornar (nivel == n - 1 && tact == p)

**FinMetodo**

**Metodo booleano** criterio(Entero nivel, Entero p)

Entero n <- s[0].length

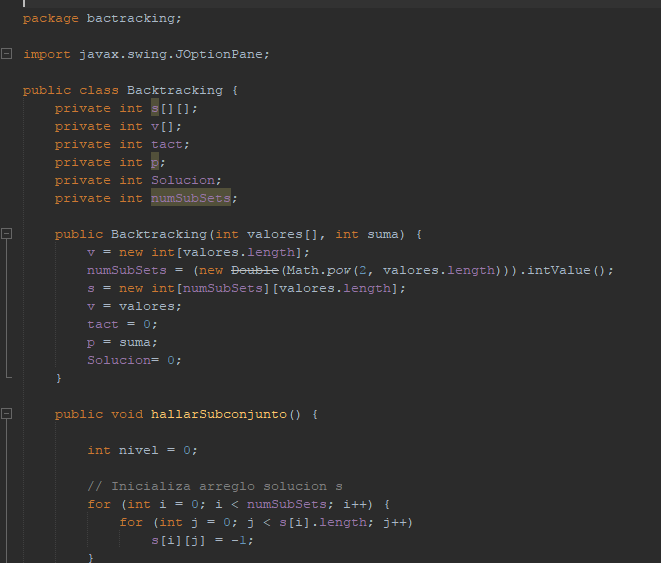
Retornar (nivel < n - 1 && tact <= p)

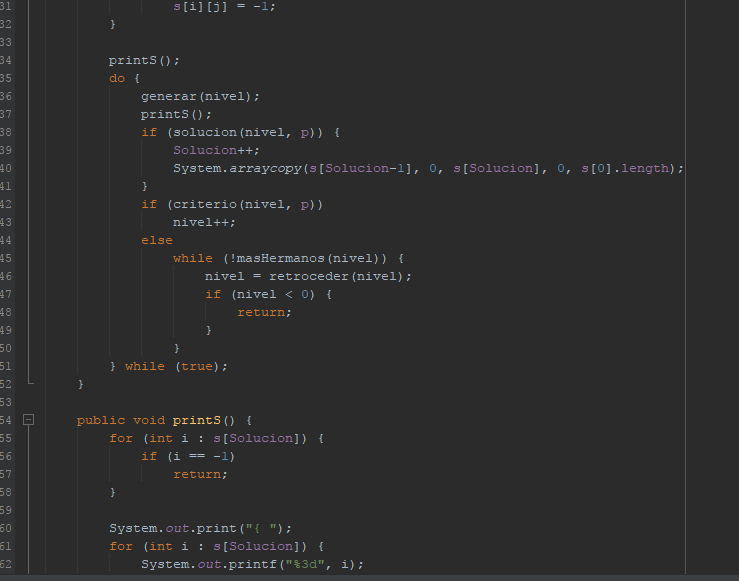
**FinMetodo**

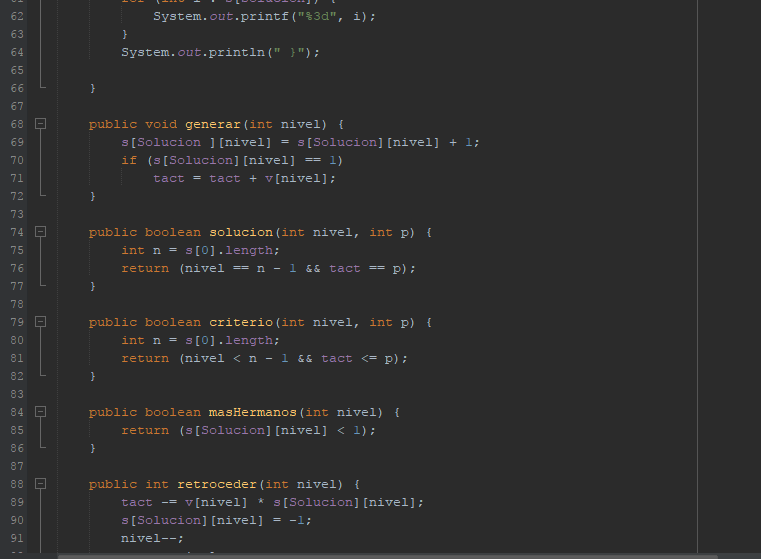
**Metodo booleano** masHermanos(Entero nivel)

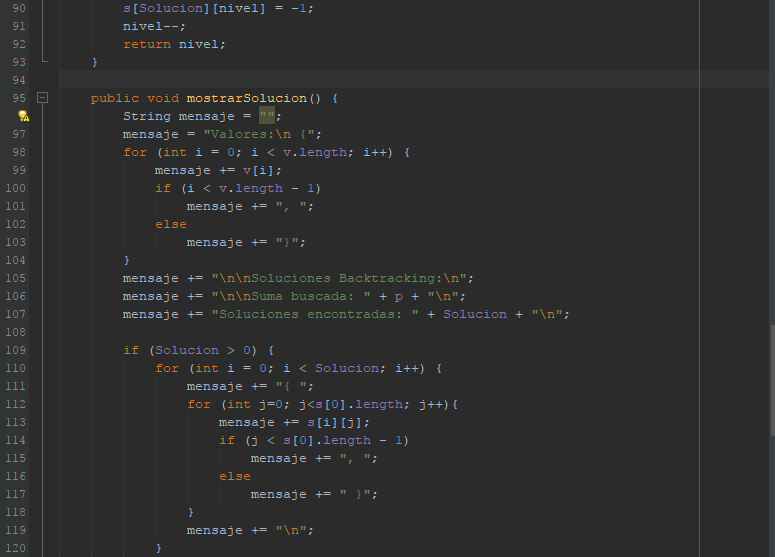
Retornar (s[Solucion][nivel] < 1)

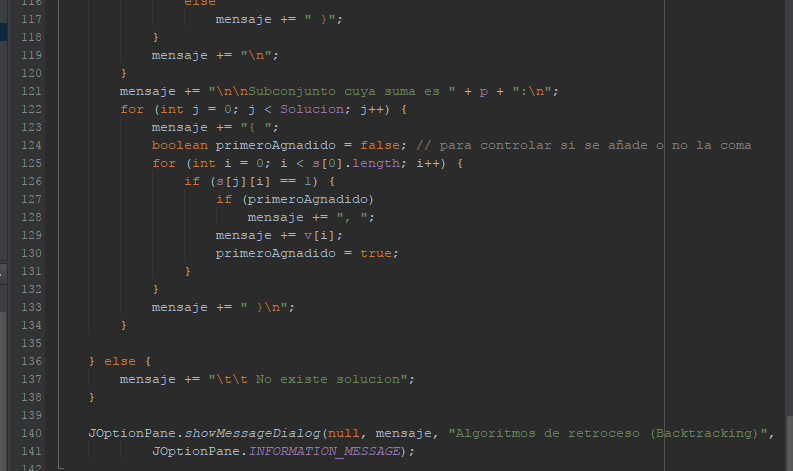
**FinMetodo**

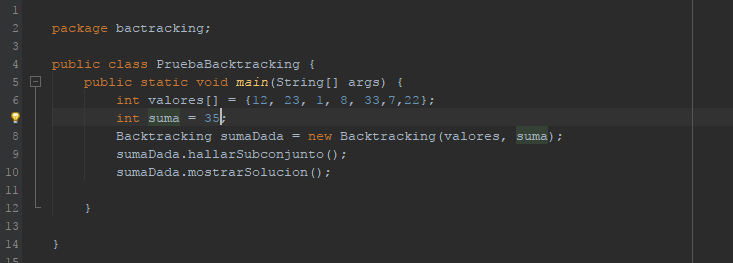
Código Java 

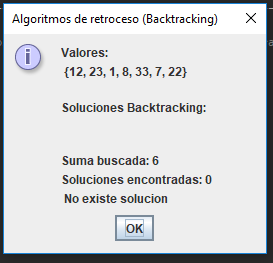
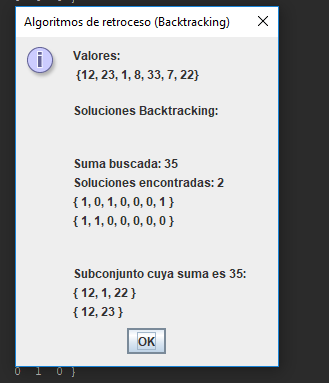








****



2. Diseñe el algoritmo e implemente en Java lo siguiente:

**Mochila 0/1 con Backtracking**

**Pseudocódigo**

**Método** mochila (Entero objetos[])

Entero n ← 1

Entero s ← 0

Entero voa, soa

Entero pesoActual←0

Entero beneficioActual←0

**Hacer**

objetos[n]←objetos[n]+1

**Si** objetos[n]=1 **entonces**

pesoActual←pesoActual+pesos[n]

beneficioActual←beneficioActual+beneficio[n]

**FinSi**

pesoActual←pesoActual+pesos[n]\*objetos[n]

beneficioActual←beneficioActual+beneficio[n]objetos[n]

**Si** Solucion(n,pesoActual) y (beneficioActual>voa) **entonces**

voa←beneficioActual

soa←s

**FinSi**

**Si** Criterio(n,pesoActual) **entonces**

n++

**Sino**

**Mientras** No(MasHermanos(n,objetos) y (n>0) **hacer**

Retroceder(n,objetos,pesoActual, beneficioActual)

**FinMientras**

**FinSi**

**Mientras** (n>0)

**FinMetodo**

**Metodo Booleano** Solucion(Entero n,Entero pesoActual)

Booleano retornar← (n=s) y pesoActual<=capacidadMochila

Devolver retornar

**FinMetodo**

**Metodo Booleano Criterio** (Entero n,Entero pesoActual)

Booleano retornar

retornar← (n<s) y pesoActual<=capacidadMochila

Devolver retornar

**FinMetodo**

**Metodo Booleano MasHermanos** (Entero n, Objeto objetos[])

Booleano retornar←objetos[n]<1

Devolver retornar

**FinMetodo**

**Metodo Retroceder** (Entero n, Objeto objetos[], Entero pesoActual, Entero beneficioActual)

pesoActual←pesoActual-pesos[n]\*objetos[n]

beneficioActual←beneficioActual-beneficio[n]objetos[n]

objetos[n]←-1

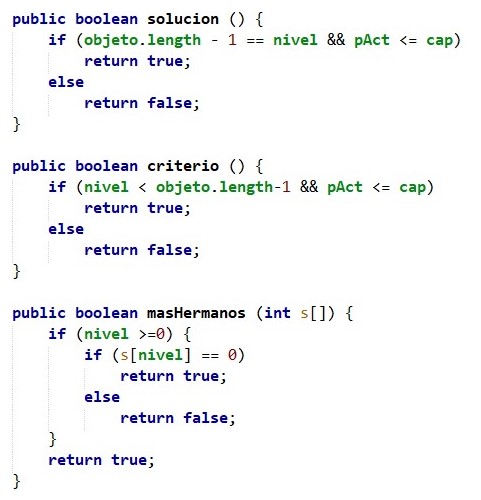
n<-n-1

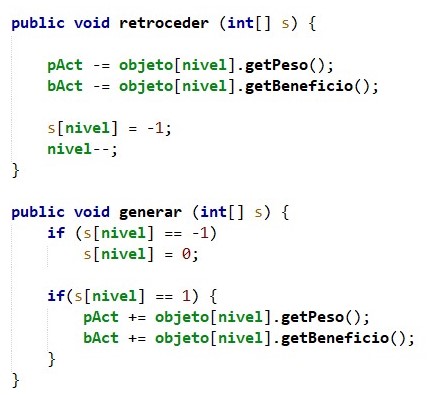
**FinMetodo**

**Fuente: Espinoza, R.(2020) *Algoritmos de retroceso.* UNMSM, Lima. pp 51-52**

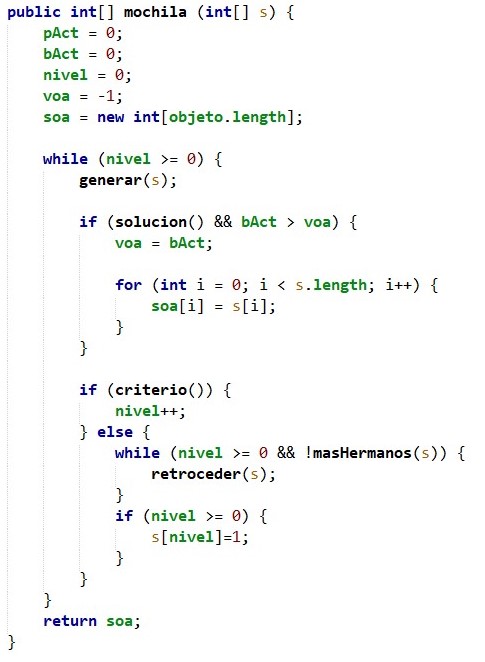
**Código**

Métodos Generales de Backtracking



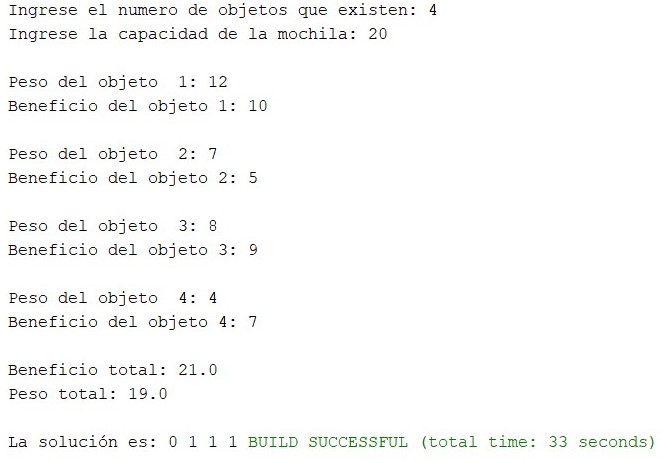


Método mochila



Fuente: Propia.

**Ejecución**

****

**PROBLEMA DE LAS N REINAS**

**Pseudocódigo**

Método Reinas (Entero Solución[], Entero k, Entero indice, Entero n)

Booleano valido

Mientras (indice<n Y k<=n Y indice>=0 Y k>=0) entonces

valido <- Valido(Solución[], indice, k, n)

Si (valido)

Solución[indice] <- k

indice <- indice + 1

k <- 0

Sino

k <- k + 1

Si (k>=n) entonces

Solución[indice] <- -1

indice <- indice - 1

Si (indice>=0) entonces

k <- Solución[indice] + 1

Solución[indice] <- -1

Fin Si

Fin Si

Fin Si

Fin Mientras

Si (k<n) entonces

Reinas(Solución[], k+1, indice-1, n)

Fin Si

Fin Método

Booleano Método Valido (Entero Solución[], Entero fila, Entero columna, Entero n)

Booleano check = Verdadero

Entero i

Para i<-0 a n-1 hacer

Si (Solución[i] <> -1) entonces

Si (Solución[i] = columna) entonces

check <- Falso

Fin Si

Fin Si

Fin Para

Para i<-0 a n-1 hacer

Si (Solución[i] <> -1) entonces

Si (Abs(i, fila) = Abs(Solución[i], columna)) entonces

check <- Falso

Fin Si

Fin Si

Fin Para

retornar check

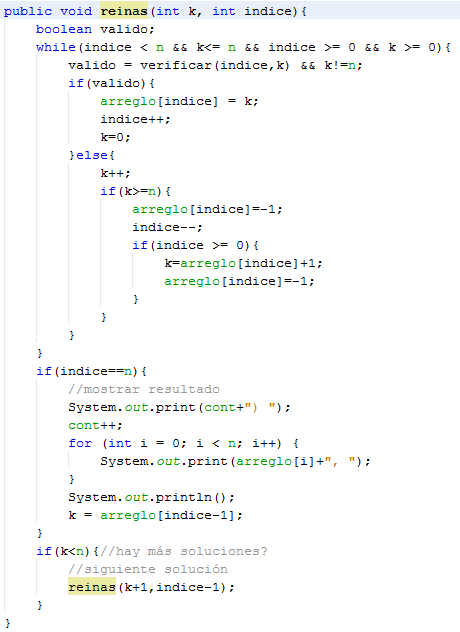
Fin Método

Entero Método Abs(Entero x, Entero y)

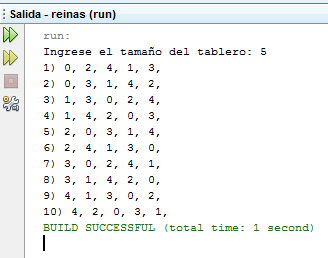
retornar |x-y|

Fin Método

**Código**

****

**Ejecución**

**  
Fuente:** Original

**ASIGNACIÓN**

**Pseudocódigo**

Clase Asignacion

tarifas[][] <- { { 1, 3, 9 } , { 6, 4, 8 }, { 15, 3, 18 } }

x[] <- entero(tarifas[0].length)

mínimo <- 9999

método asignación( entero k )

entero c

x[k] = 0

hacer

x[k] <- x[k] +1

Si aceptable(k) entonces

SI k < tarifas[0].length entonces

Si k != tarifas[0].length - 1 entonces

asignación( k +1 )

Sino

c <- coste()

Si mínimo > c entonces

mejor <- x

mínimo <- 0

fin Si

fin Si

fin Si fin Si

mientras x[k] = tarifas[0].length

fin Metodo

Metodo booleano aceptable ( entero k )

Para i<-0 hasta k-1 hacer

Si x[k] = x[1] entonces

retornar false

fin Si

fin Para

retornar true

fin Metodo

Método entero coste( )

Suma <- 0

Para i <-0 hasta tarifas[0].length hacer

Suma <- suma + tarifas[1] [x[1]]

fin Para

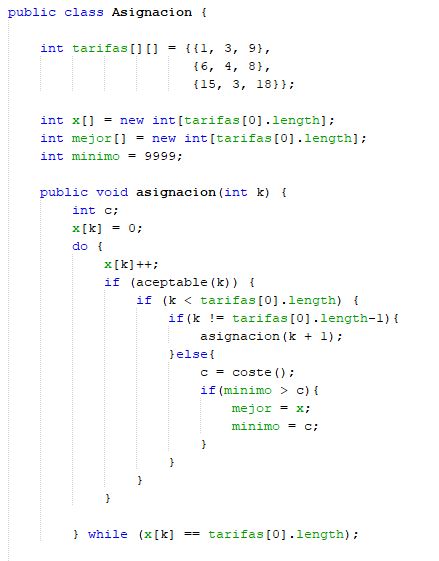
retornar suma

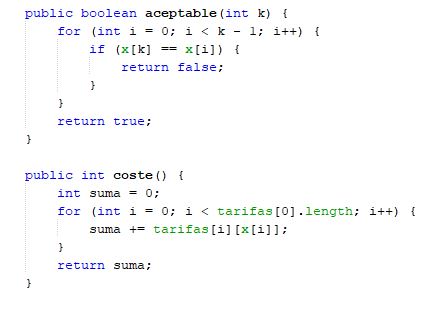
fin Metodo

fin Clase

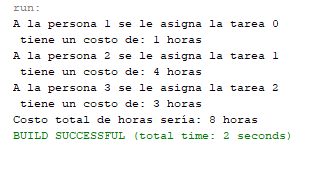
**Guerequeta R., Vallecillo A., (2000). *Técnicas de Diseño de Algoritmos.* Málaga, España*:* Editorial servicio de publicaciones de la universidad de Málaga.**

**Código Asignación**

****

****

**Ejecución**

****