



Tiempo restante: 49 minutos, 46 se

Mis Cursos

Estudiantes

Ayuda

Mis Redes

Estado de finalización de la pregunta:


20192

## Realizar examen: Examen de Ejemplo

### Información del examen

Descripción	Respondus
Instrucciones	
Examen con limitación de tiempo	Este examen tiene un límite de tiempo de 50 minutos.El examen se guardará y se enviará automáticamente cuando el tiempo se agote. Se mostrará una advertencia cuando falte <b>la mitad del tiempo, 5 minutos, 1 minuto y 30 segundos.</b>
Intentos múltiples	Este examen permite 10 intentos. Intento número 2.
Forzar terminación	Este examen se puede guardar y reanudar en cualquier momento hasta que el tiempo se haya agotado. El tiempo seguirá corriendo aunque salga del examen.  Este examen no permite volver atrás. No es posible modificar la respuesta después de la entrega.  Las respuestas se guardan automáticamente.

Guardar y enviar

 Haga clic en **Enviar** para completar esta evaluación.

Pregunta 1 de 1

### Pregunta 1

1 puntos

Guardar respuesta

#### Problema de la mochila

Se tiene un conjunto de  $n$  objetos de los que se conoce su peso unitario, su valor unitario, y su multiplicidad. Se tiene una mochila con una capacidad dada que establece el peso máximo que puede almacenar. Se desea determinar los objetos que deben añadirse en la mochila que maximicen el valor total acumulado.

Los datos de entrada se modelan mediante los tipos:

*DatosMochila*, que tiene las propiedades:

- `objetosDisponibles`: `List<ObjetoMochila>`, lista de objetos
- `n`: Integer, número de objetos, derivada
- `c0`: Integer, capacidad de la mochila

Tiempo restante: **49 minutos, 46 segundos.**

🔍 Estado de finalización de la pregunta:

La solución la modelamos con el tipo:

*SolucionMochila*, que tiene las propiedades:

- *solucion*: Map<ObjetoMochila,Integer>, número de veces que aparece cada objeto en la solución

Factoría:

*SolucionMochila*.of(Map<ObjetoMochila,Integer> s)

Resuelva el problema mediante Programación Dinámica. Para ello, se cuenta con la clase

*ProblemaMochilaPD* implements *ProblemaPD* <*SolucionMochila*, A, *ProblemaMochilaPD*>, que tiene las propiedades:

- *i*: Integer, básica
- *c*: Integer, básica

Factoría:

*ProblemaMochilaPD*.of(Integer *i*, Integer *c*)

### Ejemplo de preguntas

- A] Indique qué valores tendrán las propiedades *i* y *c* del problema inicial
- B] Indique el tipo de algoritmo, elija entre: *Min*, *Max*
- C] ¿Cuál es el tipo *A* de las alternativas adecuado para este problema? Explique el significado de tomar como alternativa uno de los valores del tipo elegido para *A*.
- D] Implemente el método *ProblemaMochilaPD* *getSubProblema*(A *a*)
- E] Implemente el método *int size()*
- F] Implemente el método *List<A> getAlternativas()*
- G] Implemente el método *Sp<A> getSolucionParcialCasoBase()*
- H] Implemente el método *Sp<A> getSolucionParcialPorAlternativa* (A *a*, *Sp<A>r*)
- I] Implemente el método *Boolean esCasoBase()*.
- J] Implemente el método *SolucionMochila* *getSolucionReconstruidaCasoBase*(*Sp<A>sp*).
- K] Implemente el método *SolucionMochila* *getSolucionReconstruidaCasoRecurso*(*Sp<A>sp*, *SolucionMochila s*).

			Arial	▼	3 (12pt)	▼										
--	--	--	-------	---	----------	---	--	--	--	--	--	--	--	--	--	--

Tiempo restante: **49 minutos, 46 segundos.**

🚩 Estado de finalización de la pregunta:

Guardar y enviar