Lab 2

Rodrigo Cayazaya Marín

Rol: 201773538-4

Greedy

(a) (15 puntos) Describa su algoritmo Greedy, explicando sus componentes, su funcionamiento y manejo de factibilidad. Obtenga una solución candidata factible para la instancia asignada, detallando los objetos seleccionados y su ganancia total.

Mi algoritmo Greedy esta dividido en 3 partes:

Primero, se ordena la lista dependiendo de si la función miope busca maximizar la ganancia, minimizar el peso o maximizar el ratio ganancia/peso (este último no se utilizó).

```python
def ordenar_objetos(objetos,version):
    if(version == 1):#Ordenar por peso
        objetos.sort(key=lambda x: x[1])
    elif(version == 2):#Ordenar por ganancia
        objetos.sort(key=lambda x: x[0], reverse = True)
    elif(version == 3):#Ordenar por ratio
        objetos.sort(key=lambda x: x[0]/x[1], reverse = True)
    return objetos
```

Segundo, se agregan los valores factibles, los cuales son los que el peso máximo lo permita (lo tachado es para el algoritmo greedy con aleatoriedad).

```python
def greedy(n,cmax,objetos,aleatorio = 0, alpha = 0):
    solucion = []
    peso = 0
    ganancia = 0
    if(aleatorio == 1):
        n = n - 1
        objeto_random = objetos[random.randint(0, n)]
        solucion.append(objeto_random)
        peso += objeto_random[1]
        ganancia += objeto_random[0]
        objetos.remove(objeto_random)
        print("Se ha agregado el objeto: ", objeto_random)
    if(alpha == 0):
        for i in range(n):
            objeto = objetos[i]
            if(peso + objeto[1] <= cmax):
                solucion.append(objeto)
                peso += objeto[1]
                ganancia += objeto[0]
```

Tercero, se codifica en una lista binaria con los resultados dependiendo de la posición del objeto, siendo 1 si entró a la mochila y 0 si no entró.

```python
def lista_binaria(objetos,n):
    lista_bin = []
    objetos.sort(key=lambda x: x[2])
    for i in range(n):
        flag = 0
        for j in range(len(objetos)):
            if(objetos[j][2] == i):
                lista_bin.append(1)
                flag = 1
            if(objetos[j][2] > i):
                break
        if(flag == 0):
            lista_bin.append(0)
    return lista_bin
```

La solución factible se realizó para la instancia 2 y utilizando la función miope de maximizar ganancia.

```python
archivo = "Inst02.txt"

n,cmax,objetos = leer_txt(archivo)
# print("Objetos:",objetos)

objetos = ordenar_objetos(objetos,2)
# print("Objetos:",objetos)

solucion_greedy,peso,ganancia = greedy(n,cmax,objetos,aleatorio = 0, alpha = 0)
print("Peso",peso)
print("Ganancia",ganancia)
# print("Objetos:",objetos)
# print("Greedy:",solucion_greedy)

solucion = lista_binaria(solucion_greedy,n)
print("Lista binaria:",solucion)
```

Obteniendo de resultado un peso de 981 y una ganancia de 1041, siendo 2 objetos agregados a la mochila.

```
Peso 981
Ganancia 1041
Lista binaria: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

(b) (15 puntos) Del algoritmo de la pregunta (a), modifique su función miope y obtenga una solución candidata factible. Compare los objetos seleccionados y la ganancia total de ambas soluciones. Explique la razón de las diferencias (en caso de existir)

Utilizando la función miope de minimizar peso se obtienen los siguientes resultados.

```
archivo = "Inst02.txt"

n,cmax,objetos = leer_txt(archivo)
# print("Objetos:",objetos)

objetos = ordenar_objetos(objetos,1)
# print("Objetos:",objetos)

solucion_greedy,peso,ganancia = greedy(n,cmax,objetos,aleatorio = 0, alpha = 0)
print("Peso",peso)
print("Ganancia",ganancia)
# print("Objetos:",objetos)
# print("Greedy:",solucion_greedy)

solucion = lista_binaria(solucion_greedy,n)
print("Lista binaria:",solucion)
```

Obteniendo de resultado un peso de 965 y una ganancia de 862, siendo 13 objetos agregados a la mochila.

```
Peso 965
Ganancia 862
Lista binaria: [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Debido a que Greedy busca óptimos locales, el óptimo local que encontró al optimizar la ganancia fue mejor que el obtenido al optimizar el peso, es por esto que existe esta diferencia de valores.

Greedy Aleatorizado

(c) (15 puntos) Describa su algoritmo Greedy Aleatorizado, explicando sus componentes, su funcionamiento y manejo de factibilidad. Obtenga una solución candidata factible para la instancia asignada, detallando los objetos seleccionados y su ganancia total.

Su ordenamiento y codificación en lista binaria es igual al Greedy explicado anteriormente. El cambio se encuentra antes de buscar soluciones factibles, ya que agrega un objeto de forma aleatoria antes de empezar la búsqueda.

```python
def greedy(n,cmax,objetos,aleatorio = 0, alpha = 0):
    solucion = []
    peso = 0
    ganancia = 0
    if(aleatorio == 1):
        n -= 1
        objeto_random = objetos[random.randint(0, n)]
        solucion.append(objeto_random)
        peso += objeto_random[1]
        ganancia += objeto_random[0]
        objetos.remove(objeto_random)
        print("Se ha agregado el objeto: ", objeto_random)
    if(alpha == 0):
        for i in range(n):
            objeto = objetos[i]
            if(peso + objeto[1] <= cmax):
                solucion.append(objeto)
                peso += objeto[1]
                ganancia += objeto[0]
```

La solución factible se realizó para la instancia 2 y utilizando la función miope de maximizar ganancia.

```python
archivo = "Inst02.txt"

n,cmax,objetos = leer_txt(archivo)
# print("Objetos:",objetos)

objetos = ordenar_objetos(objetos,2)
# print("Objetos:",objetos)

solucion_greedy,peso,ganancia = greedy(n,cmax,objetos,aleatorio = 1, alpha = 0)
print("Peso",peso)
print("Ganancia",ganancia)
# print("Objetos:",objetos)
# print("Greedy:",solucion_greedy)

solucion = lista_binaria(solucion_greedy,n)
print("Lista binaria:",solucion)
```

Obteniendo un peso de 990, una ganancia de 1083, 2 objetos agregados a la mochila y siendo el objeto con ganancia 795, peso 799 e índice 95 (empezando desde el 0) el objeto insertado de forma aleatoria.

```
Se ha agregado el objeto:  (795, 799, 95)
Peso 990
Ganancia 1083
Lista binaria: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
 0, 0, 0]
```

(d) (15 puntos) Del algoritmo de la pregunta (c), modifique su función miope y obtenga una solución candidata factible. Compare los objetos seleccionados y la ganancia total de ambas soluciones. Explique la razón de las diferencias (en caso de existir)

La solución factible se realizó para la instancia 2 y utilizando la función miope de minimizar peso.

```
archivo = "Inst02.txt"

n,cmax,objetos = leer_txt(archivo)
# print("Objetos:",objetos)

objetos = ordenar_objetos(objetos,1)
# print("Objetos:",objetos)

solucion_greedy,peso,ganancia = greedy(n,cmax,objetos,aleatorio = 1, alpha = 0)
print("Peso",peso)
print("Ganancia",ganancia)
# print("Objetos:",objetos)
# print("Greedy:",solucion_greedy)

solucion = lista_binaria(solucion_greedy,n)
print("Lista binaria:",solucion)
```

Debido a que el primer objeto agregado es aleatorio, este objeto es distinto al agregado en la solución anterior. Obteniendo un peso de 932, una ganancia de 984, 12 objetos agregados a la mochila y siendo el objeto con ganancia 210, peso 196 e índice 63 (empezando desde el 0) el objeto insertado de forma aleatoria.

```
Se ha agregado el objeto:  (210, 196, 63)
Peso 932
Ganancia 984
Lista binaria: [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0]
```

Existe una diferencia entre los valores de ganancia y peso entre ambas soluciones, esto es debido a que el objeto agregado de forma aleatoria ayuda a la diversificación, sin embargo, no existe una certeza de que esta diversificación sea para mejor o para peor. En ambos casos esta diversificación fue mejor, ya que optimizando la ganancia sin aleatoriedad se había obtenido una ganancia total de 1041 y ahora se obtuvo una de 1083, para la optimización de peso sin aleatoriedad se había obtenido una ganancia total de 862 y ahora es de 984. Sin embargo, no cambió el hecho de que la primera implementación (optimizando ganancia y con aleatoriedad) obtuvo nuevamente mejores valores que la segunda implementación (optimizando peso y con aleatoriedad). Esto es debido a que el valor agregado de forma aleatoria en la primera implementación fue un valor con mucha ganancia y el valor agregado de forma aleatoria en la segunda implementación no es de los valores con mayor peso, por lo que el algoritmo pudo diversificar, pero no de una manera tan abrupta.

Modificando Greedy Aleatorizado

(e) (30 puntos) Suponga que usted tiene una lista de candidatos L de largo $\alpha = 5$. En la lista de candidatos usted incluirá los α objetos más prometedores (según una función miope que usted elija). Luego, entre esos α objetos, escogerá el siguiente a agregar en su solución candidata de manera aleatoria. Encuentre una solución para la instancia asignada. Compare con las soluciones obtenidas en las preguntas anteriores.

La solución factible se realizó para la instancia 2, utilizando la función miope de maximizar ganancia y Alpha = 5.

```
archivo = "Inst02.txt"

n,cmax,objetos = leer_txt(archivo)
# print("Objetos:",objetos)

objetos = ordenar_objetos(objetos,2)
# print("Objetos:",objetos)

solucion_greedy,peso,ganancia = greedy(n,cmax,objetos,aleatorio = 0 , alpha = 5)
print("Peso",peso)
print("Ganancia",ganancia)
# print("Objetos:",objetos)
# print("Greedy:",solucion_greedy)

solucion = lista_binaria(solucion_greedy,n)
print("Lista binaria:",solucion)
```

Con un Alpha igual a 5 se agregaron 3 objetos. El primero es un valor entre los 5 con más ganancia (ganancia 934, peso 936 e índice 67). Para la siguiente iteración buscó entre otros 5 valores posibles, agregando un valor con ganancia 1, peso 29 e índice 48. Por último, solo alcanzó a agregar el valor con ganancia 1, peso 9 e índice 10.

```
(Alpha) Se ha agregado el objeto:  (934, 936, 67)
(Alpha) Se ha agregado el objeto:  (1, 29, 48)
(Alpha) Se ha agregado el objeto:  (1, 9, 10)
Peso 974
Ganancia 936
Lista binaria: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Debido a que se deben escoger entre 5 valores de forma aleatoria, esto permitió diversificar al algoritmo. Es por ello que se obtuvo una mayor cantidad de objetos (3 objetos), lo cual es la mayor cantidad de objetos escogidos entre las funciones miopes de maximización de ganancia (pregunta *a* y *c*). Sin embargo, a pesar de tener más objetos obtuvo un peor resultado, ya que el primer objeto agregado tiene un peso muy alto y no se compensa con su ganancia, además el segundo y tercer objeto tienen mucho más peso que ganancia. Es por esto que se obtiene una solución que diversifica más, pero encuentra un óptimo peor.

(f) (10 puntos) Evalúe los cambios en la solución generada en (e) al considerar $\alpha = 10$.

La solución factible se realizó para la instancia 2, utilizando la función miope de maximizar ganancia y Alpha = 10.

```
archivo = "Inst02.txt"

n,cmax,objetos = leer_txt(archivo)
# print("Objetos:",objetos)

objetos = ordenar_objetos(objetos,2)
# print("Objetos:",objetos)

solucion_greedy,peso,ganancia = greedy(n,cmax,objetos,aleatorio = 0 , alpha = 10)
print("Peso",peso)
print("Ganancia",ganancia)
# print("Objetos:",objetos)
# print("Greedy:",solucion_greedy)

solucion = lista_binaria(solucion_greedy,n)
print("Lista binaria:",solucion)
```

Con un Alpha igual a 10 se agregaron 4 objetos. El primero es un valor entre los 10 con más ganancia (ganancia 931, peso 886 e índice 91). Para la siguiente iteración buscó entre otros 10 valores posibles, agregando un valor con ganancia 76, peso 46 e índice 53. Para la siguiente iteración buscó entre otros 10 valores posibles, agregando un valor con ganancia 1, peso 9 e índice 10. Por último, solo alcanzó a agregar el valor con ganancia 1, peso 29 e índice 48.

```
(Alpha) Se ha agregado el objeto:  (931, 886, 91)
(Alpha) Se ha agregado el objeto:  (76, 46, 53)
(Alpha) Se ha agregado el objeto:  (1, 9, 10)
(Alpha) Se ha agregado el objeto:  (1, 29, 48)
Peso 970
Ganancia 1009
Lista binaria: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
```

En este caso ocurrió que el primer valor encontrado tiene una ganancia más baja, pero con menor peso, logrando así poder agregar un objeto extra que no se pudo agregar con Alpha = 5, el cual tiene una ganancia de 76, obteniendo una ganancia total mayor a la de la parte *e*. Sin embargo, no fue suficiente para vencer a los algoritmos de la parte *a* y *c* (debido a lo explicado en la parte *e*). También este resultado es el que mayor cantidad de objetos pudo agregar de entre los algoritmos con función miope de optimización de ganancia, siendo 4 objetos los agregados.