

## Concurso Programacion 2

Tecnicatura Universitaria en Inteligencia Artificial

Dámian Ariel Marotte

## Section 1

### Fuerza Bruta

# Problema

Perdida de contraseña.

# Algoritmo

```
1  from password import primer_intento, es_solucion, siguiente
2
3  intento_actual = primer_intento()
4
5  while not es_solucion(intento_actual):
6      intento_actual = siguiente(intento_actual)
7
8  print(intento_actual)
```

## Detalles

```
1  alfabeto = "0123456789abcdefghijklmnopqrstuvwxyz"
2
3  def es_solucion(intento):
4      return intento == "abc123"
5
6  def primer_intento():
7      return "a"
8
9  def siguiente(intento):
10     if intento == alfabeto[-1] * len(intento):
11         return alfabeto[0] * (len(intento) + 1)
12     elif intento[-1] != alfabeto[-1]:
13         return intento[:-1] + alfabeto[alfabeto.find(intento[-1]) + 1]
14     else:
15         return siguiente(intento[:-1]) + alfabeto[0]
```

## Section 2

### Greedy

# Problema

Dar el vuelto.

# Algoritmo

```
1  from coins import es_solucion, elegir_candidato, candidatos, es_factible
2
3  eleccion_actual = []
4
5  while not es_solucion(eleccion_actual):
6      x = elegir_candidato()
7      candidatos.remove(x)
8      if es_factible(eleccion_actual + [x]):
9          eleccion_actual.append(x)
10
11 print(eleccion_actual)
```



## Detalles

```
1 total = 6
2 candidatos = [4] * total + [3] * total + [1] * total
3
4 def es_solucion(eleccion_actual):
5     return sum(eleccion_actual) == total
6
7 def elegir_candidato():
8     return max(candidatos)
9
10 def es_factible(eleccion):
11     return sum(eleccion) <= total
```

## Section 3

# Divide & Conquer

# Problema

Ordenar numeros.

# Algoritmo

```
1  from sort import problema, es_caso_base, dividir, fusionar
2
3  def resolver(problema):
4      if es_caso_base(problema):
5          return problema
6
7      subproblema1, subproblema2 = dividir(problema)
8      solucion1, solucion2 = resolver(subproblema1), resolver(subproblema2)
9
10     return fusionar(solucion1, solucion2)
11
12 print(resolver(problema))
```

## Detalles

```
1  problema = [3, 5, 22, 1, 0, 2, -1, 6, 7, 11]
2
3  def es_caso_base(problema):
4      return len(problema) <= 1
5
6  def dividir(problema):
7      mitad = len(problema) // 2
8      return problema[:mitad], problema[mitad:]
9
10 def fusionar(solucion1, solucion2):
11     solucion = []
12
13     while solucion1 and solucion2:
14         if solucion1[0] <= solucion2[0]:
15             solucion.append(solucion1.pop(0))
16         else:
17             solucion.append(solucion2.pop(0))
18
19     return solucion + (solucion1 if solucion1 else solucion2)
```