Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación

Clase 14: Búsqueda en listas

Rodrigo Toro Icarte (rntoro@uc.cl)

IIC1103 Introducción a la Programación - Sección 5

06 de Mayo, 2015

Clases pasadas

El mundo está lleno de listas.

Clases pasadas

Sintaxis

lista = [elemento_1, elemento_2, elemento_3, ...]

Muchas veces necesitamos ordenar una lista.

- l.sort()
- SelectSort()
- InsertSort()

Búsqueda

Problema: Encontrar elementos en una lista que cumplan una o más condiciones.

Búsqueda en listas

Implemente la función `buscar(l,e)` que retorna True ssi $e \in l$.

Búsqueda en listas

Implemente la función `buscar(l,e)` que retorna True ssi $e \in l$.

```
1  def buscar(l,e):
2    for l_e in l:
3      if(e == l_e):
4        return True
5    return False
6
7  # Ejemplo
8  l = [5,3,7,6,5,8,12,54,3]
9  print(buscar(l,12))          # >>> True
10 print(buscar(l,13))          # >>> False
```

Búsqueda en listas

Implemente la función `buscar(l,e)` que retorna True ssi $e \in l$.

```
1  def buscar(l,e):
2    for l_e in l:
3      if(e == l_e):
4        return True
5    return False
6
7  # Ejemplo
8  l = [5,3,7,6,5,8,12,54,3]
9  print(buscar(l,12))          # >>> True
10 print(buscar(l,13))          # >>> False
```

¿Complejidad?

Búsqueda en listas

Implemente la función `buscar(l,e)` que retorna True ssi $e \in l$.

```
1  def buscar(l,e):
2    for l_e in l:
3      if(e == l_e):
4        return True
5    return False
6
7  # Ejemplo
8  l = [5,3,7,6,5,8,12,54,3]
9  print(buscar(l,12))          # >>> True
10 print(buscar(l,13))          # >>> False
```

¿Complejidad? $\rightarrow O(n)$.

Búsqueda en listas

Yo les dije: *"Es más rápido buscar en una lista ordenada"*.

Búsqueda en listas

Yo les dije: *"Es más rápido buscar en una lista ordenada"*.

Búsqueda binaria

Experimento:

73	95	82	42	8	77	32	42	68	96	2	58	80	30
30	39	57	75	28	86	91	31	91	49	92	16	49	57
46	51	49	19	44	42	96	89	1	62	4	15	0	79

¿93 pertenece a la lista?

Búsqueda binaria

Experimento:

73	95	82	42	8	77	32	42	68	96	2	58	80	30
30	39	57	75	28	86	91	31	91	49	92	16	49	57
46	51	49	19	44	42	96	89	1	62	4	15	0	79

¿93 pertenece a la lista?

0	1	2	4	8	15	16	19	28	30	30	31	32	39
42	42	42	44	46	49	49	49	51	57	57	58	62	68
73	75	77	79	80	82	86	89	91	91	92	95	96	96

¿93, 8, 82, 42 pertenecen a la lista?

Búsqueda binaria

¿Cómo aprovechamos la lista ordenada para buscar?

Búsqueda binaria

¿Cómo aprovechamos la lista ordenada para buscar?

Idea: Mirar el elemento central.

- Si es igual al valor buscado retornar True.
- Si es mayor busco a la izquierda.
- Si es menor busco a la derecha.

Búsqueda binaria

Ejemplo: Busquemos 82

0	1	2	4	8	15	16	19	28	30	30	31	32	39
42	42	42	44	46	49	49	49	51	57	57	58	62	68
73	75	77	79	80	82	86	89	91	91	92	95	96	96

Búsqueda binaria

Ejemplo: Busquemos 82

0	1	2	4	8	15	16	19	28	30	30	31	32	39
42	42	42	44	46	49	49	49	51	57	57	58	62	68
73	75	77	79	80	82	86	89	91	91	92	95	96	96

El valor central es 49

0	1	2	4	8	15	16	19	28	30	30	31	32	39
42	42	42	44	46	49	**49**	49	51	57	57	58	62	68
73	75	77	79	80	82	86	89	91	91	92	95	96	96

$82 > 49 \rightarrow$ Descartamos lado izquierdo.

Búsqueda binaria

Ejemplo: Busquemos 82

0	1	2	4	8	15	16	19	28	30	30	31	32	39
42	42	42	44	46	49	49	49	51	57	57	58	62	68
73	75	77	79	80	82	86	89	91	91	92	95	96	96

Búsqueda binaria

Ejemplo: Busquemos 82

0	1	2	4	8	15	16	19	28	30	30	31	32	39
42	42	42	44	46	49	49	49	51	57	57	58	62	68
73	75	77	79	80	82	86	89	91	91	92	95	96	96

El nuevo valor central es 79

0	1	2	4	8	15	16	19	28	30	30	31	32	39
42	42	42	44	46	49	**49**	49	51	57	57	58	62	68
73	75	77	**79**	80	82	86	89	91	91	92	95	96	96

$82 > 79 \rightarrow$ Descartamos lado izquierdo.

Búsqueda binaria

Ejemplo: Busquemos 82

0	1	2	4	8	15	16	19	28	30	30	31	32	39
42	42	42	44	46	49	49	49	51	57	57	58	62	68
73	75	77	79	80	82	86	89	91	91	92	95	96	96

Búsqueda binaria

Ejemplo: Busquemos 82

0	1	2	4	8	15	16	19	28	30	30	31	32	39
42	42	42	44	46	49	49	49	51	57	57	58	62	68
73	75	77	79	80	82	86	89	91	91	92	95	96	96

El nuevo valor central es 91

0	1	2	4	8	15	16	19	28	30	30	31	32	39
42	42	42	44	46	49	49	49	51	57	57	58	62	68
73	75	77	79	80	82	86	89	**91**	91	92	95	96	96

$82 < 91 \rightarrow$ Descartamos lado derecho.

Búsqueda binaria

Ejemplo: Busquemos 82

0	1	2	4	8	15	16	19	28	30	30	31	32	39
42	42	42	44	46	49	49	49	51	57	57	58	62	68
73	75	77	79	80	82	86	89	91	91	92	95	96	96

Búsqueda binaria

Ejemplo: Busquemos 82

0	1	2	4	8	15	16	19	28	30	30	31	32	39
42	42	42	44	46	49	49	49	51	57	57	58	62	68
73	75	77	79	80	82	86	89	91	91	92	95	96	96

El nuevo valor central es 82

0	1	2	4	8	15	16	19	28	30	30	31	32	39
42	42	42	44	46	49	49	49	51	57	57	58	62	68
73	75	77	79	80	**82**	86	89	91	91	92	95	96	96

$82 = 82 \rightarrow$ Retorno True.

Búsqueda binaria

Entonces... ¿Cómo programamos esto?

Búsqueda binaria

Entonces... ¿Cómo programamos esto?

```
1  def bs_r(l,e,i,j):
2    # Caso base
3    if(i > j):
4      return False
5    # Llamado recursivo
6    n = (i + j) // 2
7    if(e < l[n]):     # Busco a la izquierda
8      return bs_r(l,e,i,n-1)
9    elif(e > l[n]):   # Busco a la derecha
10     return bs_r(l,e,n+1,j)
11   else:             # e == l[n]
12     return True
13
14 def binary_search(l,e):
15   return bs_r(l,e,0,len(l)-1)
```

Búsqueda sobre matrices

Para muchos problemas prácticos es necesario buscar valores sobre una matriz.

Búsqueda sobre matrices

Para muchos problemas prácticos es necesario buscar valores sobre una matriz.

Ejemplo: Pronóstico del tiempo Santiago.

Hora	7 may	8 may	9 may	10 may	11 may	12 may	13 may
00:00	07°C	07°C	07°C	09°C	11°C	13°C	10°C
04:00	05°C	05°C	06°C	08°C	09°C	11°C	07°C
08:00	10°C	13°C	14°C	17°C	18°C	17°C	11°C
12:00	16°C	19°C	20°C	23°C	22°C	22°C	16°C
16:00	16°C	18°C	19°C	23°C	22°C	20°C	14°C
20:00	09°C	09°C	10°C	12°C	12°C	12°C	07°C

Búsqueda sobre matrices

¿Cómo representamos una tabla en Python?

Búsqueda sobre matrices

¿Cómo representamos una tabla en Python?

```
T = [[ 7,  7,  7,  9, 11, 13, 10],
     [ 5,  5,  6,  8,  9, 11,  7],
     [10, 13, 14, 17, 18, 17, 11],
     [16, 19, 20, 23, 22, 22, 16],
     [16, 18, 19, 23, 22, 20, 14],
     [ 9,  9, 10, 12, 12, 12,  7]]
```

Búsqueda sobre matrices

¿Cómo representamos una tabla en Python?

```
T = [[ 7,  7,  7,  9, 11, 13, 10],
     [ 5,  5,  6,  8,  9, 11,  7],
     [10, 13, 14, 17, 18, 17, 11],
     [16, 19, 20, 23, 22, 22, 16],
     [16, 18, 19, 23, 22, 20, 14],
     [ 9,  9, 10, 12, 12, 12,  7]]
```

¿Cómo encontramos la temperatura máxima y mínima de la próxima semana?

Búsqueda sobre matrices

```
1 def obtener_max(T):
2   T_max = T[0][0]
3   for i in range(len(T)):
4     for j in range(len(T[i])):
5       if(T[i][j] > T_max):
6         T_max = T[i][j]
7   return T_max
```

Búsqueda sobre matrices

```
1  def obtener_max(T):
2    T_max = T[0][0]
3    for i in range(len(T)):
4      for j in range(len(T[i])):
5        if(T[i][j] > T_max):
6          T_max = T[i][j]
7    return T_max
```

```
9  def obtener_min(T):
10   T_min = T[0][0]
11   for i in range(len(T)):
12     for j in range(len(T[i])):
13       if(T[i][j] < T_min):
14         T_min = T[i][j]
15   return T_min
```

Búsqueda sobre matrices

Lo importante: Deben saber recorrer matrices.

Búsqueda sobre matrices

Lo importante: Deben saber recorrer matrices.

Matriz uni-dimensional:

```
1 for i in range(len(T)):
2   print(T[i])     # elemento de la matriz
```

Búsqueda sobre matrices

Lo importante: Deben saber recorrer matrices.

Matriz uni-dimensional:

```
1 for i in range(len(T)):
2   print(T[i])     # elemento de la matriz
```

Matriz bi-dimensional:

```
1 for i in range(len(T)):
2   for j in range(len(T[i])):
3     print(T[i][j])     # elemento de la matriz
```

Búsqueda sobre matrices

Matriz tri-dimensional:

```
for i in range(len(T)):
  for j in range(len(T[i])):
    for k in range(len(T[i][j])):
      print(T[i][j][k])    # elemento de la matriz
```

Búsqueda sobre matrices

Matriz tri-dimensional:

```
for i in range(len(T)):
  for j in range(len(T[i])):
    for k in range(len(T[i][j])):
      print(T[i][j][k])    # elemento de la matriz
```

Matriz tetra-dimensional:

```
for i in range(len(T)):
  for j in range(len(T[i])):
    for k in range(len(T[i][j])):
      for l in range(len(T[i][j][k])):
        print(T[i][j][k][l])  # elemento de la matriz
```

Búsqueda sobre matrices

Matriz tri-dimensional:

```
for i in range(len(T)):
  for j in range(len(T[i])):
    for k in range(len(T[i][j])):
      print(T[i][j][k])     # elemento de la matriz
```

Matriz tetra-dimensional:

```
for i in range(len(T)):
  for j in range(len(T[i])):
    for k in range(len(T[i][j])):
      for l in range(len(T[i][j][k])):
        print(T[i][j][k][l])  # elemento de la matriz
```

(...)

Búsqueda sobre matrices

Observación: Los códigos anteriores también funcionan cuando las dimensiones no concuerdan.

Búsqueda sobre matrices

Observación: Los códigos anteriores también funcionan cuando las dimensiones no concuerdan.

Ejemplo: Lista de polígonos.

```
figuras = [
  [(0,0),(0,1),(1,0)],
  [(0,0),(0,1),(1,1),(1,0)],
  [(0,0),(0,1),(1,1),(0.5,2),(1,0)],
  [(0,0),(0,1),(1,1),(0.5,2),(1,0),(0.5,0.5)]]
```

Búsqueda en listas de tuplas

Problema: Encuentre elementos en una lista que cumplan una o más condiciones.

Búsqueda en listas de tuplas

Problema: Encuentre elementos en una lista que cumplan una o más condiciones.

```
1  def cumple_condicion(e):
2    if(...):
3      return True
4    return False
5
6  def buscar(L):
7    ret = []
8    for e in L:
9      if(cumple_condicion(e)):
10       ret.append(e)
11   return ret
```

Búsqueda en listas de tuplas

Ejemplo: Lista con información sobre vuelos.

```
((año,mes,día),(‘país’,‘ciudad’))
```

Búsqueda en listas de tuplas

Ejemplo: Lista con información sobre vuelos.

`((año,mes,día),(‘país’,‘ciudad’))`

```
vuelos = [
  ((2014,1,2),('Peru','Lima')),
  ((2014,1,2),('Costa Rica','San Jose')),
  ((2014,1,2),('USA','Los Angeles')),
  ((2014,1,2),('Panama','C. de Panama')),
  ((2014,1,3),('Brasil','Sao Paulo')),
  ((2014,1,3),('Costa Rica','San Jose')),
  ((2014,5,1),('Peru','Lima')),
  ((2014,5,1),('Costa Rica','San Jose')),
  ((2014,5,1),('Argentina', 'Buenos Aires'))]
```

Búsqueda en listas de tuplas

Ejemplo: Busque si existe un vuelo para ir a `destino` en `fecha`.

Búsqueda en listas de tuplas

Ejemplo: Busque si existe un vuelo para ir a `destino` en `fecha`.

```
1  def cumple_condicion(e,destino,fecha):
2    if(e[0] == fecha and e[1] == destino):
3      return True
4    return False
5
6  def vuelos_destinos(vuelos,destino,fecha):
7    for e in vuelos:
8      if(cumple_condicion(e,destino,fecha)):
9        return True
10   return False
```

Búsqueda en listas de tuplas

Problema: ¿Cómo hacemos búsqueda binaria sobre una lista de tuplas?

Búsqueda en listas de tuplas

Problema: ¿Cómo hacemos búsqueda binaria sobre una lista de tuplas?

¿Qué necesitamos para ordenar (o buscar binariamente) una lista de *elementos* de algún tipo?

Búsqueda en listas de tuplas

Problema: ¿Cómo hacemos búsqueda binaria sobre una lista de tuplas?

¿Qué necesitamos para ordenar (o buscar binariamente) una lista de *elementos* de algún tipo?

... necesitamos poder compararlos.

Función comparar(a,b)

Idea: Definir una función que compare dos elementos de la lista.

Función comparar(a,b)

Idea: Definir una función que compare dos elementos de la lista.

Restricciones:

- Parámetro: Dos elementos cualquiera de la lista (`a` y `b`).
- Retorno:
  - `0` si `a == b`.
  - Número positivo si `a` $>$ `b`.
  - Número negativo si `a` $<$ `b`.

Función comparar(a,b)

Idea: Definir una función que compare dos elementos de la lista.

Restricciones:

- Parámetro: Dos elementos cualquiera de la lista (`a` y `b`).
- Retorno:
  - `0` si `a == b`.
  - Número positivo si `a` $>$ `b`.
  - Número negativo si `a` $<$ `b`.

Importante: Si definimos correctamente `comparar(a,b)`, podremos ordenar (o buscar binariamente) cualquier lista.

Función comparar(a,b)

SelectSort:

```
4  def selectSort(l):
5    # Recorremos elementos de la lista
6    for i in range(len(l)-1):
7      # encuentro posición del elemento mínimo desde i
       en adelante
8      pos = i
9      for j in range(pos+1,len(l)):
10       if(comparar(l[pos], l[j]) > 0): # l[pos] > l[j]
11         pos = j
12     # Intercambio 'i' por posición del mínimo (pos)
13     auxiliar = l[pos]
14     l[pos] = l[i]
15     l[i] = auxiliar
16
17   # Retornamos lista ordenada
18   return l
```

Función comparar(a,b)

InsertSort:

```
4  def insertSort(l):
5    # Recorro los elementos de l a partir de 1
6    for i in range(1,len(l)):
7      # Guardo valor y posición actual
8      aux = l[i]
9      j = i
10     # Mientras el elemento de la izquierda sea menor
       lo intercambio por j
11     while(j>0 and comparar(aux,l[j-1])<0):#aux<l[j-1]
12       l[j] = l[j-1]
13       j -= 1
14     l[j] = aux
15
16   # Retornamos lista ordenada
17   return l
```

Función comparar(a,b)

Búsqueda Binaria:

```
4  def bs_r(l,e,i,j):
5    # Caso base
6    if(i > j):
7      return False
8    # Llamado recursivo
9    n = (i + j) // 2
10   if(comparar(e,l[n]) < 0):      # e < l[n]
11     return bs_r(l,e,i,n-1)
12   elif(comparar(e,l[n]) > 0):    # e > l[n]
13     return bs_r(l,e,n+1,j)
14   else:                          # e == l[n]
15     return True
16
17 def binary_search(l,e):
18   return bs_r(l,e,0,len(l)-1)
```

Función comparar(a,b)

Ejemplos:

Ordenar números (ints o floats):

Función comparar(a,b)

Ejemplos:

Ordenar números (ints o floats):

```
1 def comparar(a,b):
2   return a - b
```

Ordenar texto (alfabéticamente):

Función comparar(a,b)

Ejemplos:

Ordenar números (ints o floats):

```
1 def comparar(a,b):
2   return a - b
```

Ordenar texto (alfabéticamente):

```
1 def comparar(a,b):
2   if(a == b):
3     return 0
4   if(a < b):
5     return -1
6   if(a > b):
7     return 1
```

Función comparar(a,b)

Ordenar tuplas:

```
((año,mes,día),(‘país’,‘ciudad’))
```

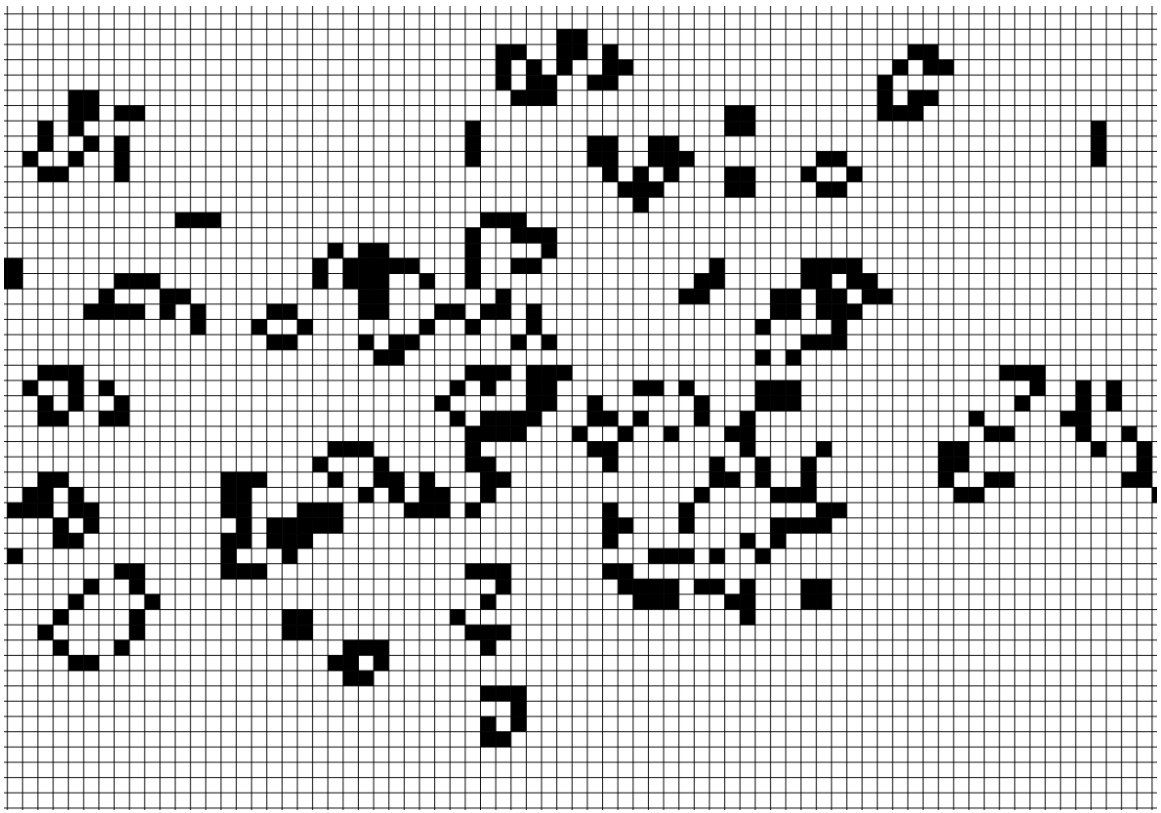
... según fecha y rompiendo empates por destino.

Función comparar(a,b)

```
1  def comparar_int(a,b):
2    return a - b
3  def comparar_str(a,b):
4    if(a == b): return 0
5    if(a < b): return -1
6    if(a > b): return 1
7
8  # a y b => ((año,mes,día),(país,ciudad))
9  def comparar(a,b):
10   # Primero comparo según fecha
11   for i in range(3):
12     if(a[0][i] != b[0][i]):
13       return comparar_int(a[0][i],b[0][i])
14   # Luego comparo nombres ciudades
15   for i in range(2):
16     if(a[1][i] != b[1][i]):
17       return comparar_str(a[1][i],b[1][i])
18   # Si llego aquí son iguales
19   return 0
```

Ejemplo notable!

El juego de la vida



Ejemplo notable!

Idea: Se tiene una cuadrícula con células que nacen, sobreviven, se reproducen y mueren.

Ejemplo notable!

Idea: Se tiene una cuadrícula con células que nacen, sobreviven, se reproducen y mueren.

Reglas:

- Célula con menos de 2 vecinos muere... de soledad u.u

Ejemplo notable!

Idea: Se tiene una cuadrícula con células que nacen, sobreviven, se reproducen y mueren.

Reglas:

- Célula con menos de 2 vecinos muere... de soledad u.u
- Célula con 2 o 3 vecinos sobrevive.

Ejemplo notable!

Idea: Se tiene una cuadrícula con células que nacen, sobreviven, se reproducen y mueren.

Reglas:

- Célula con menos de 2 vecinos muere... de soledad u.u
- Célula con 2 o 3 vecinos sobrevive.
- Célula con más de 3 vecinos muere por sobre-población.

Ejemplo notable!

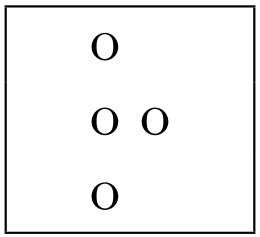
Idea: Se tiene una cuadrícula con células que nacen, sobreviven, se reproducen y mueren.

Reglas:

- Célula con menos de 2 vecinos muere... de soledad u.u
- Célula con 2 o 3 vecinos sobrevive.
- Célula con más de 3 vecinos muere por sobre-población.
- Espacio vacío con 3 células vecinas nace!

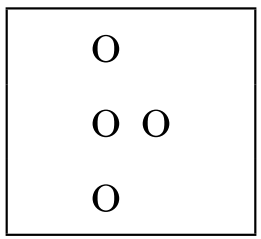
Ejemplo notable!

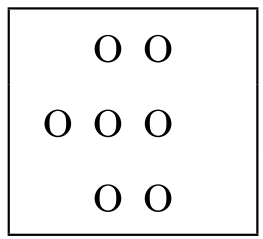
Ejemplo:



Ejemplo notable!

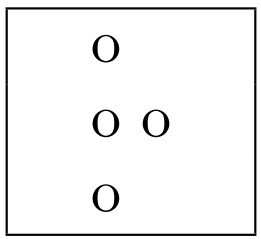
Ejemplo:

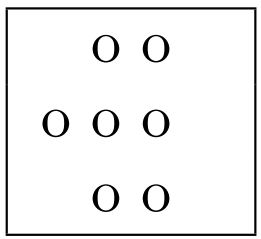


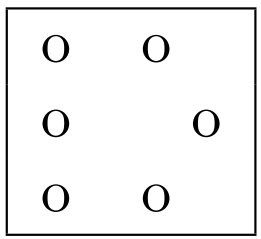


Ejemplo notable!

Ejemplo:

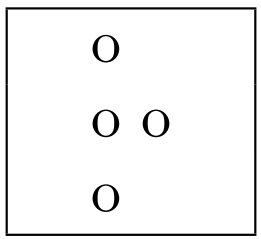


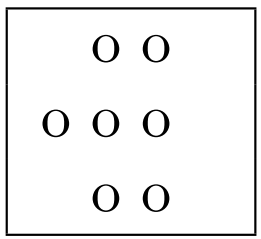


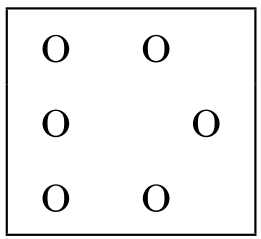


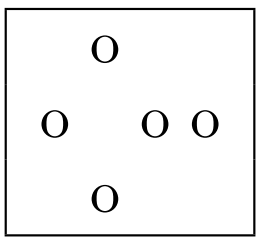
Ejemplo notable!

Ejemplo:



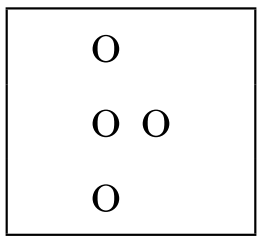


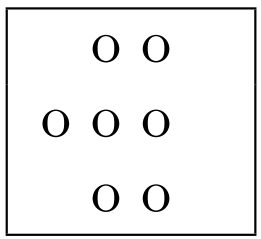


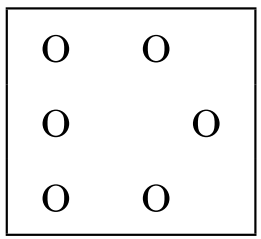


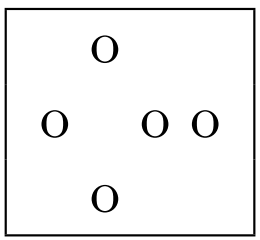
Ejemplo notable!

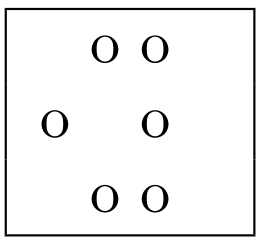
Ejemplo:





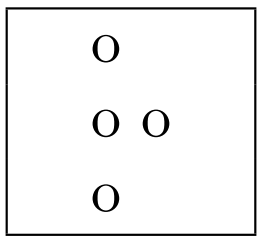


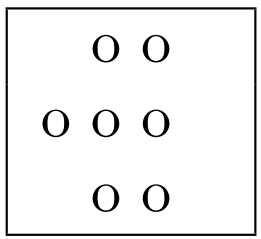


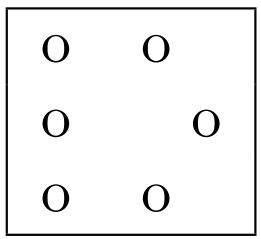


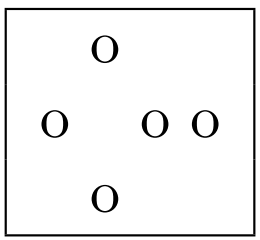
Ejemplo notable!

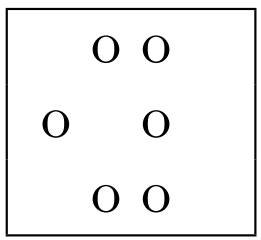
Ejemplo:

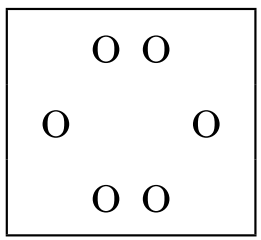






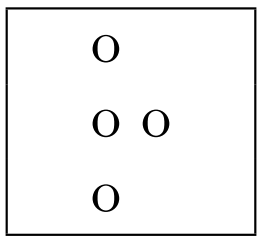
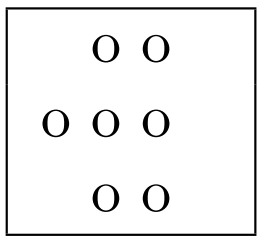
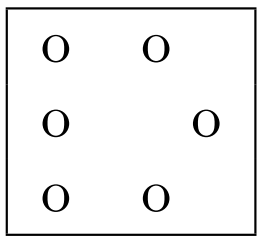
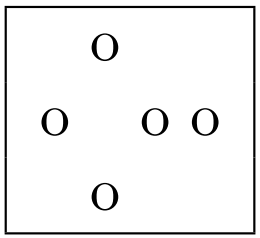
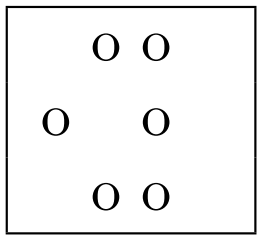
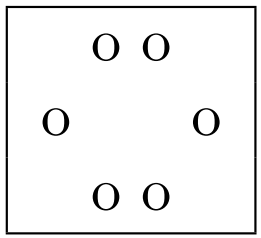






Ejemplo notable!

Ejemplo:

Ver Demo.

Ejemplo notable!

Estrategia:

- Tablero será lista de lista 2D.
- Dimensiones dadas por usuario.
- Tablero inicial random (probabilidad usuario).
- En tablero ‘1’ es célula y ‘0’ es nada.

Ejemplo notable!

Crear tablero inicial:

Ejemplo notable!

Crear tablero inicial:

```
15 # función que retorna un tablero inicial de nxm
16 # y probabilidad p
17 def crear_tablero_inicial(n,m,p):
18   T = []
19   for i in range(n):
20     fila = []
21     for j in range(m):
22       if(random.random() < p):
23         # agrego célula
24         fila.append(1)
25       else:
26         # casilla vacía
27         fila.append(0)
28     T.append(fila)
29
30   return T
```

Ejemplo notable!

Mostrar tablero:

Ejemplo notable!

Mostrar tablero:

```
3  # Muestro tablero
4  def mostrar_tablero(T):
5    print('\n'+'-'*2*len(T)+'\n')
6    for i in range(len(T)):
7      fila = " "
8      for j in range(len(T[i])):
9        if(T[i][j] == 1):
10         fila += "o"
11       else:
12         fila += " "
13     print(fila)
```

Ejemplo notable!

¿Cómo avanzamos un turno?

Ejemplo notable!

¿Cómo avanzamos un turno?

Estrategia:

- Creamos tablero alternativos con ceros.
- Por cada célula, sumo 1 a posiciones vecinas.
- **Resultado**: Tablero con el conteo de células vecinas.

Ejemplo notable!

¿Cómo avanzamos un turno?

Estrategia:

- Creamos tablero alternativos con ceros.
- Por cada célula, sumo 1 a posiciones vecinas.
- **Resultado**: Tablero con el conteo de células vecinas.

Ejemplo:

	o			2	2	3	1
	o o	$\rightarrow$		3	3	3	1
	o			2	2	3	1

Ejemplo notable!

```
33 def avanzar_turno(T,n,m):
34   # creo un nuevo tablero para el siguiente turno
35   T_n = crear_tablero_inicial(n,m,-1)
36
37   # recorro el tablero y por cada célula
38   # sumo uno a sus vecinos
39   for i in range(n):
40     for j in range(m):
41       # si la casilla posee una célula
42       if(T[i][j] == 1):
43         # agrego 1 a mis vecinos
44         for di in [-1,0,1]:
45           for dj in [-1,0,1]:
46             if(di == dj == 0): continue
47             # chequeo límites
48             if(-1 < i + di < n and -1 < j + dj < m):
49               # sumo 1 a la casilla
50               T_n[i + di][j + dj] += 1
```

Ejemplo notable!

Tablero de vecinos indica quién nacen, vive y muere.

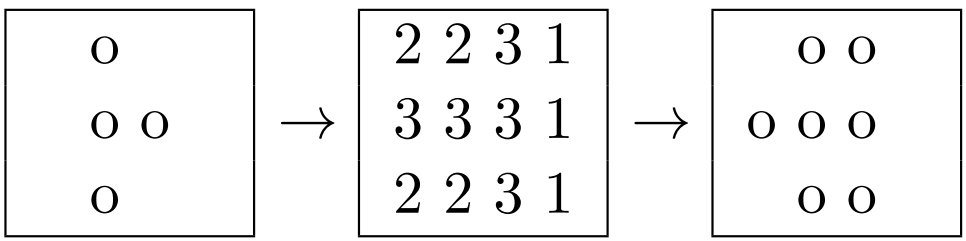
- < 2 vecinos $\rightarrow$ muere.
- 2 vecinos $\rightarrow$ sobrevive.
- 3 vecinos $\rightarrow$ Nace o sobrevive.
- > 3 vecinos $\rightarrow$ muere.

Ejemplo notable!

Tablero de vecinos indica quién nacen, vive y muere.

- < 2 vecinos $\rightarrow$ muere.
- 2 vecinos $\rightarrow$ sobrevive.
- 3 vecinos $\rightarrow$ Nace o sobrevive.
- > 3 vecinos $\rightarrow$ muere.

Ejemplo:



Ejemplo notable!

```
52    # Reglas!
53    for i in range(n):
54      for j in range(m):
55        # si es '3' nace o sigue viviendo
56        if(T_n[i][j] == 3):
57          T_n[i][j] = 1
58        # si es '2' y ya vivía, entonces sobrevive
59        elif(T_n[i][j] == 2 and T[i][j] == 1):
60          T_n[i][j] = 1
61        # otro caso muere o sigue muerta
62        else:
63          T_n[i][j] = 0
64
65    return T_n
```

Ejemplo notable!

Finalmente jugar es un loop infinito...

```
67 # Inicia juego con un tablero de nxm y probabilidad p
68 def jugar(n,m,p):
69
70   # Creo el tablero
71   T = crear_tablero_inicial(n,m,p)
72
73   while(True):
74     # Muestro el tablero
75     mostrar_tablero(T)
76     T = avanzar_turno(T,n,m)
77     input()
```

Ejercicios

1) Para la tabla de temperaturas encuentre:

- Temperatura mínima y máxima de cada día.
- Temperatura mínima y máxima por cada hora.
- Temperatura media de cada día.
- Temperatura media de cada hora.
- Número de veces en que la temperatura superará los 15 °C.

2) Cree un programa que muestre el perímetro de cada una de las figuras de la `lista de polígonos`.

3) Agregue células mutantes al juego de la vida. Ellas nacen con probabilidad 0.1 y sólo mueren si tienen 5 o más vecinos.