INF-354-2P-P4

November 26, 2023

1 Segundo Examen Parcial INF - 354

1.0.1 Nombre: Steve Brandom Nina Huacani

1.1 Pregunta N°4: Con el uso de DEAP, resolver el anterior ejercicio.

[2]:
```
#importamos la libreria random
import random
#importamos la libreria numpy
import numpy as np
#importamos la libreria deap
from deap import base, creator, tools, algorithms
```

[3]:
```
#Creamos la 'caja de herramientas' de deap
toolbox = base.Toolbox()
#creamos una instancia de la clase FitnessMin para representar la funcion de
#aptitud que queremos minimizar
creator.create('FitnessMin', base.Fitness, weights=(-1,))
creator.create('Individual', list, fitness=creator.FitnessMin)
#Definimos la cantidad de nodos del grafo
n_nodos = 5
#Registramos la funcion llamada genes que generara una permutacion aleatoria
toolbox.register('Genes', np.random.permutation, n_nodos)
#registramos la funcion individuals para inicializar un nuevo individuo
toolbox.register('Individuals', tools.initIterate, creator.Individual, toolbox.
 ↪Genes)
#registramos la funcion de poblacion como una lista de individuios
toolbox.register('Population', tools.initRepeat, list, toolbox.Individuals)
#registrmos una funcion de cruce
toolbox.register('mate', tools.cxPartialyMatched)
#registramos ua funcion de mutacion
toolbox.register('mutate', tools.mutShuffleIndexes, indpb = 0.1)
#registramos una funcion de seleccion
toolbox.register('select', tools.selTournament, tournsize = 2)
#Creamos una poblacion de tamaño 10
poblacion = toolbox.Population(n=10)

#Definimos el grafo
```

```
grafo = [[0, 7, 9, 8, 20],
         [7, 0, 10, 4, 11],
         [9, 10, 0, 15, 5],
         [8, 4, 15, 0, 17],
         [20, 11, 5, 17, 0]]
```

[4]:
```
#definimos la funcion para calcular la distancia entre los nodos
def calcular_distancia(lista_nodos):
  #definimos la distancia inicialmente en cero
  distancia_total = 0
  #iteramos sobre los nodos
  for i in range(n_nodos-1):
    #obtenemos el valor del nodo a
    nodo_a = lista_nodos[i]
    #obtenemos el valor del nodo b
    nodo_b = lista_nodos[i+1]
    #hallamos la distancia entre los nodos
    distancia = grafo[nodo_a][nodo_b]
    #almacenamos las distancias
    distancia_total += distancia
  #añadimos la distancia de vuelta al nodo inicial
  distancia_total += grafo[lista_nodos[-1]][lista_nodos[0]]
  #retornamos la distancia total
  return distancia_total,

#definimos la funcion para guardar la forma
def guardar_lista(individuos):
  #retornamos la distancia calculada
  return individuos.calcular_distancia

#registramos la funcion evaluar
toolbox.register('evaluate', calcular_distancia)

#definiremos las estadisticas a calcular
estadisticas = tools.Statistics()
#registraremos la media
estadisticas.register('mean', np.mean)
#registramos el minimo
estadisticas.register('min', np.min)
#registramos el maximo
estadisticas.register('max', np.max)
```

[5]:
```
#Definiremos el salon de la fama almacenaremos al mejor
salon_de_la_fama = tools.HallOfFame(1)

#definimos los resultados y el log
resultados, log = algorithms.eaSimple(poblacion,
```

```
                                    toolbox,
                                    cxpb = 0.8,
                                    mutpb = 0.1,
                                    stats = estadisticas,
                                    ngen = 1000,
                                    halloffame = salon_de_la_fama,
                                    verbose = True)
```

gen	nevals	mean	min	max
0	10	2	0	4
1	8	2	0	4
2	8	2	0	4
3	8	2	0	4
4	10	2	0	4
5	8	2	0	4
6	10	2	0	4
7	10	2	0	4
8	6	2	0	4
9	6	2	0	4
10	10	2	0	4
11	8	2	0	4
12	7	2	0	4
13	10	2	0	4
14	8	2	0	4
15	10	2	0	4
16	10	2	0	4
17	8	2	0	4
18	10	2	0	4
19	10	2	0	4
20	7	2	0	4
21	10	2	0	4
22	10	2	0	4
23	10	2	0	4
24	10	2	0	4
25	8	2	0	4
26	8	2	0	4
27	5	2	0	4
28	8	2	0	4
29	5	2	0	4
30	6	2	0	4
31	8	2	0	4
32	6	2	0	4
33	8	2	0	4
34	10	2	0	4
35	4	2	0	4
36	8	2	0	4
37	8	2	0	4
38	10	2	0	4

39	9	2	0	4
40	10	2	0	4
41	8	2	0	4
42	10	2	0	4
43	8	2	0	4
44	10	2	0	4
45	6	2	0	4
46	6	2	0	4
47	7	2	0	4
48	8	2	0	4
49	9	2	0	4
50	8	2	0	4
51	8	2	0	4
52	8	2	0	4
53	9	2	0	4
54	10	2	0	4
55	8	2	0	4
56	8	2	0	4
57	10	2	0	4
58	10	2	0	4
59	10	2	0	4
60	10	2	0	4
61	10	2	0	4
62	10	2	0	4
63	8	2	0	4
64	7	2	0	4
65	10	2	0	4
66	10	2	0	4
67	6	2	0	4
68	6	2	0	4
69	8	2	0	4
70	8	2	0	4
71	6	2	0	4
72	9	2	0	4
73	7	2	0	4
74	2	2	0	4
75	10	2	0	4
76	10	2	0	4
77	6	2	0	4
78	7	2	0	4
79	9	2	0	4
80	8	2	0	4
81	8	2	0	4
82	10	2	0	4
83	8	2	0	4
84	10	2	0	4
85	10	2	0	4
86	7	2	0	4

87	8	2	0	4
88	10	2	0	4
89	10	2	0	4
90	8	2	0	4
91	6	2	0	4
92	8	2	0	4
93	6	2	0	4
94	4	2	0	4
95	10	2	0	4
96	10	2	0	4
97	8	2	0	4
98	10	2	0	4
99	5	2	0	4
100	7	2	0	4
101	8	2	0	4
102	10	2	0	4
103	10	2	0	4
104	8	2	0	4
105	10	2	0	4
106	3	2	0	4
107	6	2	0	4
108	6	2	0	4
109	8	2	0	4
110	6	2	0	4
111	5	2	0	4
112	10	2	0	4
113	10	2	0	4
114	8	2	0	4
115	8	2	0	4
116	8	2	0	4
117	7	2	0	4
118	4	2	0	4
119	10	2	0	4
120	8	2	0	4
121	10	2	0	4
122	8	2	0	4
123	5	2	0	4
124	10	2	0	4
125	6	2	0	4
126	8	2	0	4
127	10	2	0	4
128	8	2	0	4
129	8	2	0	4
130	8	2	0	4
131	8	2	0	4
132	10	2	0	4
133	8	2	0	4
134	10	2	0	4

135	4	2	0	4
136	10	2	0	4
137	8	2	0	4
138	5	2	0	4
139	10	2	0	4
140	6	2	0	4
141	8	2	0	4
142	6	2	0	4
143	8	2	0	4
144	9	2	0	4
145	10	2	0	4
146	9	2	0	4
147	4	2	0	4
148	8	2	0	4
149	8	2	0	4
150	9	2	0	4
151	10	2	0	4
152	7	2	0	4
153	6	2	0	4
154	8	2	0	4
155	6	2	0	4
156	8	2	0	4
157	8	2	0	4
158	8	2	0	4
159	7	2	0	4
160	6	2	0	4
161	5	2	0	4
162	8	2	0	4
163	8	2	0	4
164	8	2	0	4
165	8	2	0	4
166	8	2	0	4
167	10	2	0	4
168	4	2	0	4
169	2	2	0	4
170	6	2	0	4
171	8	2	0	4
172	10	2	0	4
173	9	2	0	4
174	6	2	0	4
175	4	2	0	4
176	10	2	0	4
177	10	2	0	4
178	10	2	0	4
179	6	2	0	4
180	9	2	0	4
181	8	2	0	4
182	10	2	0	4

183	8	2	0	4
184	10	2	0	4
185	7	2	0	4
186	7	2	0	4
187	5	2	0	4
188	8	2	0	4
189	6	2	0	4
190	9	2	0	4
191	6	2	0	4
192	8	2	0	4
193	10	2	0	4
194	8	2	0	4
195	9	2	0	4
196	8	2	0	4
197	6	2	0	4
198	10	2	0	4
199	8	2	0	4
200	8	2	0	4
201	8	2	0	4
202	6	2	0	4
203	8	2	0	4
204	10	2	0	4
205	10	2	0	4
206	8	2	0	4
207	10	2	0	4
208	8	2	0	4
209	4	2	0	4
210	10	2	0	4
211	10	2	0	4
212	6	2	0	4
213	8	2	0	4
214	10	2	0	4
215	9	2	0	4
216	8	2	0	4
217	10	2	0	4
218	8	2	0	4
219	8	2	0	4
220	8	2	0	4
221	8	2	0	4
222	7	2	0	4
223	8	2	0	4
224	8	2	0	4
225	10	2	0	4
226	8	2	0	4
227	4	2	0	4
228	10	2	0	4
229	6	2	0	4
230	8	2	0	4

231	5	2	0	4
232	4	2	0	4
233	8	2	0	4
234	10	2	0	4
235	10	2	0	4
236	10	2	0	4
237	8	2	0	4
238	10	2	0	4
239	8	2	0	4
240	6	2	0	4
241	10	2	0	4
242	10	2	0	4
243	10	2	0	4
244	7	2	0	4
245	6	2	0	4
246	8	2	0	4
247	8	2	0	4
248	6	2	0	4
249	6	2	0	4
250	10	2	0	4
251	8	2	0	4
252	10	2	0	4
253	10	2	0	4
254	5	2	0	4
255	10	2	0	4
256	8	2	0	4
257	8	2	0	4
258	10	2	0	4
259	10	2	0	4
260	8	2	0	4
261	8	2	0	4
262	8	2	0	4
263	10	2	0	4
264	10	2	0	4
265	10	2	0	4
266	7	2	0	4
267	5	2	0	4
268	10	2	0	4
269	8	2	0	4
270	8	2	0	4
271	10	2	0	4
272	10	2	0	4
273	10	2	0	4
274	10	2	0	4
275	6	2	0	4
276	7	2	0	4
277	8	2	0	4
278	8	2	0	4

279	8	2	0	4
280	6	2	0	4
281	6	2	0	4
282	6	2	0	4
283	10	2	0	4
284	10	2	0	4
285	6	2	0	4
286	8	2	0	4
287	8	2	0	4
288	4	2	0	4
289	10	2	0	4
290	8	2	0	4
291	6	2	0	4
292	10	2	0	4
293	9	2	0	4
294	8	2	0	4
295	5	2	0	4
296	10	2	0	4
297	6	2	0	4
298	10	2	0	4
299	10	2	0	4
300	10	2	0	4
301	6	2	0	4
302	8	2	0	4
303	8	2	0	4
304	10	2	0	4
305	8	2	0	4
306	6	2	0	4
307	10	2	0	4
308	8	2	0	4
309	10	2	0	4
310	9	2	0	4
311	10	2	0	4
312	9	2	0	4
313	10	2	0	4
314	4	2	0	4
315	8	2	0	4
316	8	2	0	4
317	10	2	0	4
318	10	2	0	4
319	8	2	0	4
320	10	2	0	4
321	6	2	0	4
322	6	2	0	4
323	7	2	0	4
324	10	2	0	4
325	8	2	0	4
326	9	2	0	4

327	10	2	0	4
328	10	2	0	4
329	8	2	0	4
330	7	2	0	4
331	8	2	0	4
332	6	2	0	4
333	10	2	0	4
334	6	2	0	4
335	6	2	0	4
336	10	2	0	4
337	10	2	0	4
338	8	2	0	4
339	10	2	0	4
340	8	2	0	4
341	10	2	0	4
342	6	2	0	4
343	5	2	0	4
344	10	2	0	4
345	8	2	0	4
346	6	2	0	4
347	6	2	0	4
348	8	2	0	4
349	10	2	0	4
350	6	2	0	4
351	8	2	0	4
352	8	2	0	4
353	8	2	0	4
354	8	2	0	4
355	8	2	0	4
356	6	2	0	4
357	8	2	0	4
358	10	2	0	4
359	9	2	0	4
360	6	2	0	4
361	10	2	0	4
362	4	2	0	4
363	10	2	0	4
364	9	2	0	4
365	10	2	0	4
366	7	2	0	4
367	6	2	0	4
368	8	2	0	4
369	6	2	0	4
370	10	2	0	4
371	6	2	0	4
372	4	2	0	4
373	8	2	0	4
374	7	2	0	4

375	10	2	0	4
376	8	2	0	4
377	9	2	0	4
378	8	2	0	4
379	8	2	0	4
380	8	2	0	4
381	9	2	0	4
382	8	2	0	4
383	6	2	0	4
384	8	2	0	4
385	10	2	0	4
386	8	2	0	4
387	9	2	0	4
388	6	2	0	4
389	4	2	0	4
390	10	2	0	4
391	10	2	0	4
392	10	2	0	4
393	10	2	0	4
394	10	2	0	4
395	8	2	0	4
396	8	2	0	4
397	8	2	0	4
398	8	2	0	4
399	9	2	0	4
400	9	2	0	4
401	10	2	0	4
402	10	2	0	4
403	10	2	0	4
404	8	2	0	4
405	8	2	0	4
406	10	2	0	4
407	10	2	0	4
408	8	2	0	4
409	10	2	0	4
410	8	2	0	4
411	10	2	0	4
412	10	2	0	4
413	10	2	0	4
414	10	2	0	4
415	8	2	0	4
416	7	2	0	4
417	7	2	0	4
418	10	2	0	4
419	8	2	0	4
420	10	2	0	4
421	8	2	0	4
422	10	2	0	4

423	6	2	0	4
424	10	2	0	4
425	9	2	0	4
426	10	2	0	4
427	8	2	0	4
428	10	2	0	4
429	6	2	0	4
430	8	2	0	4
431	7	2	0	4
432	8	2	0	4
433	9	2	0	4
434	8	2	0	4
435	6	2	0	4
436	9	2	0	4
437	7	2	0	4
438	7	2	0	4
439	7	2	0	4
440	8	2	0	4
441	10	2	0	4
442	10	2	0	4
443	6	2	0	4
444	9	2	0	4
445	6	2	0	4
446	10	2	0	4
447	6	2	0	4
448	8	2	0	4
449	6	2	0	4
450	9	2	0	4
451	8	2	0	4
452	8	2	0	4
453	8	2	0	4
454	8	2	0	4
455	8	2	0	4
456	8	2	0	4
457	8	2	0	4
458	10	2	0	4
459	5	2	0	4
460	8	2	0	4
461	8	2	0	4
462	8	2	0	4
463	6	2	0	4
464	10	2	0	4
465	8	2	0	4
466	6	2	0	4
467	6	2	0	4
468	8	2	0	4
469	10	2	0	4
470	7	2	0	4

471	10	2	0	4
472	10	2	0	4
473	10	2	0	4
474	10	2	0	4
475	6	2	0	4
476	8	2	0	4
477	8	2	0	4
478	10	2	0	4
479	10	2	0	4
480	10	2	0	4
481	8	2	0	4
482	10	2	0	4
483	8	2	0	4
484	10	2	0	4
485	10	2	0	4
486	10	2	0	4
487	8	2	0	4
488	6	2	0	4
489	10	2	0	4
490	8	2	0	4
491	8	2	0	4
492	6	2	0	4
493	8	2	0	4
494	6	2	0	4
495	10	2	0	4
496	6	2	0	4
497	10	2	0	4
498	8	2	0	4
499	10	2	0	4
500	6	2	0	4
501	8	2	0	4
502	8	2	0	4
503	10	2	0	4
504	8	2	0	4
505	7	2	0	4
506	8	2	0	4
507	10	2	0	4
508	10	2	0	4
509	8	2	0	4
510	8	2	0	4
511	10	2	0	4
512	5	2	0	4
513	10	2	0	4
514	6	2	0	4
515	8	2	0	4
516	8	2	0	4
517	8	2	0	4
518	6	2	0	4

519	8	2	0	4
520	10	2	0	4
521	10	2	0	4
522	10	2	0	4
523	8	2	0	4
524	10	2	0	4
525	10	2	0	4
526	8	2	0	4
527	8	2	0	4
528	9	2	0	4
529	6	2	0	4
530	8	2	0	4
531	10	2	0	4
532	8	2	0	4
533	8	2	0	4
534	8	2	0	4
535	6	2	0	4
536	10	2	0	4
537	10	2	0	4
538	10	2	0	4
539	6	2	0	4
540	10	2	0	4
541	8	2	0	4
542	9	2	0	4
543	9	2	0	4
544	8	2	0	4
545	6	2	0	4
546	7	2	0	4
547	9	2	0	4
548	8	2	0	4
549	10	2	0	4
550	8	2	0	4
551	8	2	0	4
552	6	2	0	4
553	9	2	0	4
554	10	2	0	4
555	7	2	0	4
556	9	2	0	4
557	10	2	0	4
558	8	2	0	4
559	10	2	0	4
560	8	2	0	4
561	8	2	0	4
562	8	2	0	4
563	10	2	0	4
564	9	2	0	4
565	7	2	0	4
566	10	2	0	4

567	10	2	0	4
568	2	2	0	4
569	9	2	0	4
570	8	2	0	4
571	8	2	0	4
572	8	2	0	4
573	8	2	0	4
574	10	2	0	4
575	8	2	0	4
576	8	2	0	4
577	9	2	0	4
578	8	2	0	4
579	7	2	0	4
580	8	2	0	4
581	8	2	0	4
582	6	2	0	4
583	8	2	0	4
584	4	2	0	4
585	8	2	0	4
586	8	2	0	4
587	6	2	0	4
588	10	2	0	4
589	10	2	0	4
590	8	2	0	4
591	10	2	0	4
592	8	2	0	4
593	7	2	0	4
594	7	2	0	4
595	8	2	0	4
596	10	2	0	4
597	8	2	0	4
598	10	2	0	4
599	9	2	0	4
600	8	2	0	4
601	8	2	0	4
602	4	2	0	4
603	6	2	0	4
604	6	2	0	4
605	10	2	0	4
606	5	2	0	4
607	9	2	0	4
608	8	2	0	4
609	5	2	0	4
610	6	2	0	4
611	8	2	0	4
612	8	2	0	4
613	8	2	0	4
614	8	2	0	4

615	10	2	0	4
616	10	2	0	4
617	10	2	0	4
618	8	2	0	4
619	9	2	0	4
620	8	2	0	4
621	8	2	0	4
622	8	2	0	4
623	6	2	0	4
624	6	2	0	4
625	10	2	0	4
626	8	2	0	4
627	6	2	0	4
628	10	2	0	4
629	10	2	0	4
630	7	2	0	4
631	7	2	0	4
632	10	2	0	4
633	9	2	0	4
634	10	2	0	4
635	7	2	0	4
636	9	2	0	4
637	7	2	0	4
638	10	2	0	4
639	8	2	0	4
640	8	2	0	4
641	6	2	0	4
642	8	2	0	4
643	8	2	0	4
644	5	2	0	4
645	8	2	0	4
646	8	2	0	4
647	10	2	0	4
648	10	2	0	4
649	10	2	0	4
650	7	2	0	4
651	10	2	0	4
652	6	2	0	4
653	8	2	0	4
654	8	2	0	4
655	10	2	0	4
656	10	2	0	4
657	10	2	0	4
658	8	2	0	4
659	8	2	0	4
660	6	2	0	4
661	8	2	0	4
662	6	2	0	4

663	8	2	0	4
664	8	2	0	4
665	8	2	0	4
666	8	2	0	4
667	10	2	0	4
668	6	2	0	4
669	8	2	0	4
670	10	2	0	4
671	8	2	0	4
672	8	2	0	4
673	10	2	0	4
674	5	2	0	4
675	7	2	0	4
676	6	2	0	4
677	8	2	0	4
678	9	2	0	4
679	8	2	0	4
680	8	2	0	4
681	7	2	0	4
682	10	2	0	4
683	6	2	0	4
684	6	2	0	4
685	7	2	0	4
686	9	2	0	4
687	10	2	0	4
688	10	2	0	4
689	9	2	0	4
690	10	2	0	4
691	8	2	0	4
692	8	2	0	4
693	8	2	0	4
694	8	2	0	4
695	7	2	0	4
696	6	2	0	4
697	10	2	0	4
698	10	2	0	4
699	8	2	0	4
700	10	2	0	4
701	8	2	0	4
702	10	2	0	4
703	10	2	0	4
704	10	2	0	4
705	8	2	0	4
706	6	2	0	4
707	9	2	0	4
708	8	2	0	4
709	8	2	0	4
710	8	2	0	4

711	8	2	0	4
712	8	2	0	4
713	10	2	0	4
714	10	2	0	4
715	4	2	0	4
716	8	2	0	4
717	9	2	0	4
718	6	2	0	4
719	8	2	0	4
720	6	2	0	4
721	6	2	0	4
722	10	2	0	4
723	9	2	0	4
724	10	2	0	4
725	9	2	0	4
726	10	2	0	4
727	10	2	0	4
728	10	2	0	4
729	10	2	0	4
730	8	2	0	4
731	6	2	0	4
732	10	2	0	4
733	8	2	0	4
734	8	2	0	4
735	8	2	0	4
736	6	2	0	4
737	7	2	0	4
738	10	2	0	4
739	10	2	0	4
740	6	2	0	4
741	8	2	0	4
742	10	2	0	4
743	10	2	0	4
744	8	2	0	4
745	10	2	0	4
746	8	2	0	4
747	8	2	0	4
748	6	2	0	4
749	10	2	0	4
750	10	2	0	4
751	10	2	0	4
752	9	2	0	4
753	10	2	0	4
754	6	2	0	4
755	9	2	0	4
756	6	2	0	4
757	10	2	0	4
758	8	2	0	4

759	10	2	0	4
760	8	2	0	4
761	8	2	0	4
762	8	2	0	4
763	10	2	0	4
764	5	2	0	4
765	10	2	0	4
766	9	2	0	4
767	8	2	0	4
768	4	2	0	4
769	8	2	0	4
770	8	2	0	4
771	8	2	0	4
772	5	2	0	4
773	8	2	0	4
774	8	2	0	4
775	10	2	0	4
776	8	2	0	4
777	8	2	0	4
778	8	2	0	4
779	8	2	0	4
780	10	2	0	4
781	7	2	0	4
782	9	2	0	4
783	7	2	0	4
784	10	2	0	4
785	8	2	0	4
786	10	2	0	4
787	8	2	0	4
788	6	2	0	4
789	10	2	0	4
790	6	2	0	4
791	8	2	0	4
792	8	2	0	4
793	10	2	0	4
794	10	2	0	4
795	8	2	0	4
796	5	2	0	4
797	10	2	0	4
798	8	2	0	4
799	4	2	0	4
800	7	2	0	4
801	6	2	0	4
802	10	2	0	4
803	10	2	0	4
804	8	2	0	4
805	8	2	0	4
806	10	2	0	4

807	10	2	0	4
808	10	2	0	4
809	6	2	0	4
810	9	2	0	4
811	10	2	0	4
812	10	2	0	4
813	10	2	0	4
814	9	2	0	4
815	7	2	0	4
816	8	2	0	4
817	5	2	0	4
818	6	2	0	4
819	10	2	0	4
820	8	2	0	4
821	7	2	0	4
822	7	2	0	4
823	8	2	0	4
824	4	2	0	4
825	7	2	0	4
826	10	2	0	4
827	8	2	0	4
828	9	2	0	4
829	10	2	0	4
830	10	2	0	4
831	6	2	0	4
832	9	2	0	4
833	8	2	0	4
834	6	2	0	4
835	10	2	0	4
836	8	2	0	4
837	7	2	0	4
838	3	2	0	4
839	8	2	0	4
840	6	2	0	4
841	10	2	0	4
842	6	2	0	4
843	8	2	0	4
844	8	2	0	4
845	10	2	0	4
846	8	2	0	4
847	7	2	0	4
848	6	2	0	4
849	10	2	0	4
850	8	2	0	4
851	4	2	0	4
852	10	2	0	4
853	9	2	0	4
854	10	2	0	4

855	6	2	0	4
856	10	2	0	4
857	8	2	0	4
858	7	2	0	4
859	10	2	0	4
860	6	2	0	4
861	8	2	0	4
862	8	2	0	4
863	8	2	0	4
864	6	2	0	4
865	10	2	0	4
866	10	2	0	4
867	9	2	0	4
868	10	2	0	4
869	6	2	0	4
870	9	2	0	4
871	10	2	0	4
872	8	2	0	4
873	6	2	0	4
874	10	2	0	4
875	6	2	0	4
876	8	2	0	4
877	10	2	0	4
878	10	2	0	4
879	10	2	0	4
880	10	2	0	4
881	7	2	0	4
882	10	2	0	4
883	8	2	0	4
884	6	2	0	4
885	8	2	0	4
886	8	2	0	4
887	8	2	0	4
888	6	2	0	4
889	10	2	0	4
890	7	2	0	4
891	9	2	0	4
892	10	2	0	4
893	10	2	0	4
894	8	2	0	4
895	8	2	0	4
896	8	2	0	4
897	8	2	0	4
898	10	2	0	4
899	10	2	0	4
900	6	2	0	4
901	8	2	0	4
902	10	2	0	4

903	8	2	0	4
904	7	2	0	4
905	8	2	0	4
906	7	2	0	4
907	9	2	0	4
908	10	2	0	4
909	8	2	0	4
910	8	2	0	4
911	7	2	0	4
912	9	2	0	4
913	7	2	0	4
914	4	2	0	4
915	10	2	0	4
916	10	2	0	4
917	8	2	0	4
918	8	2	0	4
919	10	2	0	4
920	6	2	0	4
921	10	2	0	4
922	10	2	0	4
923	8	2	0	4
924	9	2	0	4
925	10	2	0	4
926	8	2	0	4
927	8	2	0	4
928	10	2	0	4
929	10	2	0	4
930	8	2	0	4
931	10	2	0	4
932	10	2	0	4
933	10	2	0	4
934	4	2	0	4
935	6	2	0	4
936	8	2	0	4
937	8	2	0	4
938	8	2	0	4
939	8	2	0	4
940	6	2	0	4
941	8	2	0	4
942	10	2	0	4
943	6	2	0	4
944	8	2	0	4
945	8	2	0	4
946	9	2	0	4
947	10	2	0	4
948	8	2	0	4
949	8	2	0	4
950	8	2	0	4

951	4	2	0	4
952	9	2	0	4
953	8	2	0	4
954	8	2	0	4
955	5	2	0	4
956	6	2	0	4
957	8	2	0	4
958	10	2	0	4
959	8	2	0	4
960	6	2	0	4
961	10	2	0	4
962	8	2	0	4
963	10	2	0	4
964	9	2	0	4
965	8	2	0	4
966	6	2	0	4
967	8	2	0	4
968	8	2	0	4
969	8	2	0	4
970	6	2	0	4
971	8	2	0	4
972	4	2	0	4
973	5	2	0	4
974	10	2	0	4
975	8	2	0	4
976	8	2	0	4
977	10	2	0	4
978	10	2	0	4
979	6	2	0	4
980	6	2	0	4
981	7	2	0	4
982	9	2	0	4
983	6	2	0	4
984	10	2	0	4
985	6	2	0	4
986	7	2	0	4
987	8	2	0	4
988	8	2	0	4
989	10	2	0	4
990	10	2	0	4
991	8	2	0	4
992	9	2	0	4
993	10	2	0	4
994	6	2	0	4
995	6	2	0	4
996	10	2	0	4
997	8	2	0	4
998	8	2	0	4

```
999     10      2       0       4
1000    6       2       0       4
```

[6]:
```
#MOstraremos los resultados
print(resultados)
#Mostraremos el salon de la fama
print(salon_de_la_fama)
```

```
[[4, 1, 3, 0, 2], [4, 1, 3, 0, 2], [4, 1, 3, 0, 2], [4, 1, 3, 0, 2], [4, 1, 3,
0, 2], [4, 1, 3, 0, 2], [4, 1, 3, 0, 2], [4, 1, 3, 0, 2], [4, 1, 3, 0, 2], [4,
1, 3, 0, 2]]
[[4, 1, 3, 0, 2]]
```

[7]:
```
#Obtendremos la mejor ruta
def ruta(posicion):
  nodos = ['A','B','C','D','E']
  return nodos[posicion]
#Mostramos la mejor ruta
print("La mejor ruta es: ")
for i in range(n_nodos):
  print(ruta(salon_de_la_fama[0][i]), end=' -> ')
print(ruta(salon_de_la_fama[0][0]),"\n")
#Mostramos el costo de la mejor solucion
print(f"El costo de la mejor solucion es:␣
 ↪{calcular_distancia(salon_de_la_fama[0])}")
```

```
La mejor ruta es:
E -> B -> D -> A -> C -> E

El costo de la mejor solucion es: (37,)
```

[ ]:
```

```