UNIVERSIDAD SIMÓN BOLÍVAR
Inteligencia Artificial I – Proyecto 1
CI-5437

Estudiante:
Alejandro Meneses 18-10536
Oliver Bueno 15-10192

Proyecto 1

Objetivo: El objetivo general de este proyecto es analizar y comparar diferentes métodos de búsqueda heurística para resolver problemas de inteligencia artificial. Los problemas que se estudiaron son el N-puzzle, el Cubo de Rubik, el Top Spin y la Torre de Hanoi con 4 astas, los cuales son problemas clásicos de manipulación de objetos que requieren de una secuencia de acciones para alcanzar un estado objetivo. Para representar los problemas se utilizó el lenguaje PSVN, el cual permite definir de forma sencilla el espacio de estados, las acciones y el costo de cada problema. Se estudiaron los árboles de búsqueda sin poda y con poda de ancestros, que son estructuras de datos que almacenan los estados explorados por la búsqueda.. Se utilizaron algoritmos informados como A* e IDA*, usando heurísticas PDBs y Manhattan, se espera evaluar el impacto de la poda de ancestros en el tiempo y el espacio de búsqueda, comparar el rendimiento de los algoritmos informados con distintas heurísticas PDBs, y analizar la calidad y la complejidad de las soluciones obtenidas.

Árboles de búsqueda: Se estudiaron los árboles de búsqueda de los distintos problemas. Se usó el algoritmo UCS, tomando como basé la implementación dada por el archivo “distSummary.cpp” que provee PSVN. Se realizaron dos tipos de implementaciones, una que aplica poda parcial (eliminación de ancestros) y otra que no.

La generación de los árboles de búsqueda se hizo en un computador con un procesador de 3.70 GHZ y con 16Gb de ram (acotando que disponible para las pruebas se tenían aproximadamente 8Gb de Ram).

A continuación se presentan las tablas con los resultados:

15-puzzles: Sin eliminación de duplicados

Tiempo de ejecución: 39 segundos

Profundidad	Número de nodos	Factor de ramificación
0	1	0
1	2	2
2	6	3
3	18	3
4	58	3,222222
5	186	3,206897
6	602	3,236559
7	1946	3,232558
8	6298	3,236382
9	20378	3,23563
10	65946	3,236137
11	213402	3,236011
12	690586	3,23608
13	2234778	3,23606
14	7231898	3,23607
15	23402906	3,236067
16	75733402	3,236068

15-puzzles: Con eliminación de duplicados

Tiempo de ejecución: 71 segundos

Profundidad	Número de nodos	Factor de ramificación
0	1	0
1	2	2
2	4	2
3	10	2,5
4	24	2,4
5	54	2,25
6	107	1,981481
7	212	1,981308
8	446	2,103774
9	946	2,121076
10	1948	2,059197
11	3938	2,021561
12	7808	1,982732
13	15544	1,990779
14	30821	1,982823
15	60842	1,974044
16	119000	1,955886
17	231844	1,948269
18	447342	1,929496
19	859744	1,921894
20	1637383	1,904501
21	3098270	1,892208
22	5802411	1,972791
23	10783780	1,8585
24	19826318	1,838531

24-puzzles: Sin eliminación de duplicados

Tiempo de ejecución: 35.5 segundos

Profundidad	Número de nodos	Factor de ramificación
0	1	0
1	2	2
2	6	3
3	18	3
4	60	3,333333
5	198	3,3
6	684	3,454545
7	2322	3,394737
8	8100	3,488372
9	27702	3,42
10	96876	3,497076
11	331938	3,426421
12	1161540	3,499268
13	3981798	3,428034
14	13935564	3,499817
15	47777202	3,428437

24-puzzles: Con eliminación de duplicados

Tiempo de ejecución: 64 segundos

Profundidad	Número de nodos	Factor de ramificación
0	1	0
1	2	2
2	4	2
3	10	2,5
4	26	2,6
5	64	2,461539
6	159	2,484375
7	366	2,301887
8	862	2,355191
9	1904	2,208817
10	4538	2,383403
11	10238	2,25606
12	24098	2,35378
13	53186	2,207071
14	123435	2,320817
15	268416	2,174553
16	616374	2,296339
17	1326882	2,152722
18	3021126	2,276861
19	6438828	2,131268
20	14524718	2,255801

Rubiks Cube 3x3x3: Sin eliminación de duplicados

Tiempo de ejecución: 27 segundos

Profundidad	Número de nodos	Factor de ramificación
0	1	0
1	18	18
2	324	18
3	5832	18
4	104976	18
5	1889568	18

Rubiks Cube 3x3x3: Con eliminación de duplicados

Tiempo de ejecución: 29 segundos

Profundidad	Número de nodos	Factor de ramificación
0	1	0
1	18	18
2	243	13,5
3	3240	13,333333
4	43239	13,34537
5	574908	13,296052

Topspin 12-4: Sin eliminación de duplicados

Tiempo de ejecución: 70 segundos

Profundidad	Número de nodos	Factor de ramificación
0	1	0
1	12	12
2	144	12
3	1728	12
4	20736	12
5	248832	12
6	2985984	12
7	35831808	12

Topspin 12-4: Con eliminación de duplicados

Tiempo de ejecución: 196 segundos

Profundidad	Número de nodos	Factor de ramificación
0	1	0
1	12	12
2	102	8,5
3	784	7,686275
4	5823	7,427296
5	41842	7,185643
6	287243	6,864944
7	1856702	6,463872
8	10791471	5,812172
9	49089200	4,548889

Topspin 14-4: Sin eliminación de duplicados

Tiempo de ejecución: 48 segundos

Profundidad	Número de nodos	Factor de ramificación
0	1	0
1	14	14
2	196	14
3	2744	14
4	38416	14
5	537824	14
6	7529536	14

Topspin 14-4: Con eliminación de duplicados

Tiempo de ejecución: 50.5 segundos

Profundidad	Número de nodos	Factor de ramificación
0	1	0
1	14	14
2	133	9,5
3	1106	8,315789
4	8722	7,886076
5	66654	7,642055
6	494319	7,416194
7	3549518	7,180622

Topspin 14-4: Sin eliminación de duplicados

Tiempo de ejecución: 32 segundos

Profundidad	Número de nodos	Factor de ramificación
0	1	17
1	17	17
2	289	17
3	4913	17
4	83521	17
5	1419857	17

Topspin 14-4: Con eliminación de duplicados

Tiempo de ejecución: 55 segundos

Profundidad	Número de nodos	Factor de ramificación
0	1	0
1	17	17
2	187	11
3	1734	9,272727
4	14841	8,558824
5	121214	8,180985
6	960942	7,91459
7	7392807	7,693292

Torre de hanoi 4-12: Sin eliminación de duplicados

Tiempo de ejecución: 12 segundos

Profundidad	Número de nodos	Factor de Ramificación
0	1	0
1	3	3
2	15	5
3	75	5
4	393	5,24
5	2109	5,366412
6	11487	5,446657
7	63375	5,517106
8	352755	5,566154
9	1978341	5,608258
10	11161197	5,641695

Torre de hanoi 4-12: Con eliminación de duplicados

Tiempo de ejecución: 20 segundos

Profundidad	Número de nodos	Factor de Ramificación
0	1	0
1	3	3
2	6	2
3	12	2
4	30	2,5
5	30	1
6	66	2,2
7	96	1,454545
8	126	1,3125
9	210	1,666667
10	330	1,571429
11	318	0,963636
12	462	1,45283

13	816	1,766234
14	1032	1,264706
15	936	0,906977
16	1044	1,115385
17	1752	1,678161
18	2610	1,489726
19	3036	1,163218
20	3528	1,162055
21	3306	0,937075
22	4578	1,384755
23	6318	1,380079
24	9108	1,441595
25	10674	1,171937
26	11580	1,084879
27	11844	1,022798
28	13374	1,129179
29	17124	1,280395
30	23664	1,38192
31	32184	1,360041
32	36984	1,149142
33	39822	1,076736
34	38544	0,967907
35	39936	1,036115
36	45936	1,15024
37	57900	1,262409
38	77262	1,332333
39	96366	1,247263
40	116052	1,204284
41	127092	1,09513
42	122142	0,961052
43	112086	0,91767
44	116814	1,042182
45	135078	1,156351
46	157326	1,164705
47	192498	1,223561

48	246468	1,280367
49	304638	1,236014
50	353958	1,161897
51	387960	1,096062
52	414768	1,0691
53	426558	1,028426
54	409446	0,959884
55	397866	0,971718
56	442128	1,111248
57	505836	1,144094
58	574950	1,136633
59	672774	1,170143
60	819612	1,218258
61	969552	1,18294
62	1084908	1,118979
63	1145952	1,056267
64	1174230	1,024676
65	1134474	0,966143
66	1018938	0,898159
67	832344	0,816874
68	647394	0,777796
69	458982	0,708969
70	319698	0,696537
71	195384	0,611152
72	102024	0,522172
73	44154	0,432781
74	14664	0,33211
75	3618	0,246727
76	1266	0,349917
77	294	0,232227
78	156	0,530612
79	72	0,461538
80	18	0,25
81	6	0,333333

Torre de hanoi 4-14: Sin eliminación de duplicados

Tiempo de ejecución: 12 segundos

Profundidad	Número de nodos	Factor de Ramificación
0	1	0
1	3	3
2	15	5
3	75	5
4	393	5,24
5	2109	5,366412
6	11487	5,446657
7	63375	5,517106
8	352755	5,566154
9	1978341	5,608258
10	11161197	5,641695

Torre de hanoi 4-14: Con eliminación de duplicados

Tiempo de ejecución: 70 segundos

Profundidad	Número de nodos	Factor de Ramificación
0	1	0
1	3	3
2	6	2
3	12	2
4	30	2,5
5	30	1
6	66	2,2
7	96	1,454545
8	126	1,3125
9	210	1,666667
10	330	1,571429

11	318	0,963636
12	462	1,45283
13	816	1,766234
14	1032	1,264706
15	936	0,906977
16	1044	1,115385
17	1752	1,678161
18	2610	1,489726
19	3036	1,163218
20	3528	1,162055
21	3306	0,937075
22	4578	1,384755
23	6318	1,380079
24	9108	1,441595
25	10674	1,171937
26	11580	1,084879
27	11844	1,022798
28	13374	1,129179
29	17124	1,280395
30	23664	1,38192
31	32184	1,360041
32	36984	1,149142
33	39822	1,076736
34	38544	0,967907
35	39936	1,036115
36	45936	1,15024
37	57900	1,262409
38	77262	1,332333
39	96366	1,247263
40	116052	1,204284
41	127092	1,09513
42	122142	0,961052
43	112086	0,91767
44	116814	1,042182
45	135078	1,156351

46	157326	1,164705
47	192498	1,223561
48	246468	1,280367
49	304658	1,236063
50	354018	1,162048
51	388128	1,096351
52	415206	1,069766
53	427710	1,030115
54	411804	0,962811
55	402504	0,977416
56	451236	1,121072
57	521094	1,154815
58	600684	1,152736
59	714750	1,189893
60	883602	1,236239
61	1064970	1,20526
62	1228656	1,1537
63	1349322	1,09821
64	1464336	1,085238
65	1539420	1,051275
66	1565604	1,017009
67	1554324	0,992795
68	1610436	1,036101
69	1702956	1,05745
70	1889394	1,109479
71	2143170	1,134316
72	2530788	1,180862
73	2984394	1,179235
74	3498924	1,172407
75	4006014	1,144927
76	4533006	1,13155

Torre de hanoi 4-18: Sin eliminación de duplicados

Tiempo de ejecución: 15 segundos

Profundidad	Número de nodos	Factor de Ramificación
0	1	0
1	3	3
2	15	5
3	75	5
4	393	5,24
5	2109	5,366412
6	11487	5,446657
7	63375	5,517106
8	352755	5,566154
9	1978341	5,608258
10	11161197	5,641695

Torre de hanoi 4-18: Con eliminación de duplicados

Tiempo de ejecución: 82 segundos

Profundidad	Número de nodos	Factor de Ramificación
0	1	0
1	3	3
2	6	2
3	12	2
4	30	2,5
5	30	1
6	66	2,2
7	96	1,454545
8	126	1,3125
9	210	1,666667
10	330	1,571429
11	318	0,963636
12	462	1,45283

13	816	1,766234
14	1032	1,264706
15	936	0,906977
16	1044	1,115385
17	1752	1,678161
18	2610	1,489726
19	3036	1,163218
20	3528	1,162055
21	3306	0,937075
22	4578	1,384755
23	6318	1,380079
24	9108	1,441595
25	10674	1,171937
26	11580	1,084879
27	11844	1,022798
28	13374	1,129179
29	17124	1,280395
30	23664	1,38192
31	32184	1,360041
32	36984	1,149142
33	39822	1,076736
34	38544	0,967907
35	39936	1,036115
36	45936	1,15024
37	57900	1,262409
38	77262	1,332333
39	96366	1,247263
40	116052	1,204284
41	127092	1,09513
42	122142	0,961052
43	112086	0,91767
44	116814	1,042182
45	135078	1,156351
46	157326	1,164705
47	192498	1,223561

48	246468	1,280367
49	304658	1,236063
50	354018	1,162048
51	388128	1,096351
52	415206	1,069766
53	427710	1,030115
54	411804	0,962811
55	402504	0,977416
56	451236	1,121072
57	521094	1,154815
58	600684	1,152736
59	714750	1,189893
60	883602	1,236239
61	1064970	1,20526
62	1228656	1,1537
63	1349322	1,09821
64	1464336	1,085238
65	1539420	1,051275
66	1565634	2,017025
67	1554408	0,99283
68	1610640	1,039176
69	1703436	1,057614
70	1890282	1,109688
71	2144742	1,134615
72	2533656	1,181334
73	2988834	1,179653
74	3506376	1,173159
75	4017900	1,145884
76	4551204	1,132732

Las tablas muestran la profundidad de la búsqueda, el número de nodos generados en cada nivel, así como el factor de ramificación. Además se muestran los tiempos de ejecución de cada problema hasta antes que el proceso fuera “matado” por el sistema por falta de memoria. Siendo el problema “torre de Hanoi 14-4” el único que generó todo el árbol de búsqueda.

En cada problema se observa que al realizar la poda de ancestros, disminuye considerablemente el número de nodos generados para cada nivel, por lo cual se puede llegar a una profundidad mayor de búsqueda, lo que produce que se visiten estados que no se visitaron aplicando el mismo algoritmos (sin poda).

Analizando el factor de ramificación de los problemas, se observa que para el caso en que se aplica el algoritmos sin implementar poda, se mantiene el factor de ramificación constante, esto es debido a al tipo de movimiento (reglas) de cada problema, esto solo exceptuando a los N-puzzels. Por otro lado, el factor de ramificación aplicando el algoritmo e implementando la poda de ancestros tiende a disminuir.

Concluyendo, se observa que aplicando poda de ancestros se puede recorrer una mayor cantidad niveles en el árbol de búsqueda, por lo cual aumenta la posibilidad de encontrar la solución en menor tiempo, además que es más conveniente para sistemas limitados.

Heurísticas: Se implementaron heurísticas para los distintos problemas, en el caso de 15-puzzel se usa la heurística Manhattan, así mismos para el mismo 15-puzzel y el 24-puzzel se usan diferentes PDBs aditivas. Por el lado de el cubo de rubik se usa tomando el máximo PDBs de 8 esquinas y 2 bordes. Mientras para los problemas de topspin y torre de Hanoi se toma el máximo de diferentes PDBs.

Algoritmos informados: Se analiza el comportamiento de los algoritmos informados A* e IDA* para cada problema usando la heuristicas mencionadas anteriormente.

Para el caso de 15-puzzel usando la heurística Manhattan, dio pruebas eficientes para estados “faciles”, los cuales no necesitan muchos movimientos, llegando a la solución en poco tiempo usando ambos algoritmos. Mientras que para estados más complejos en la mayoría de las ocasiones el proceso era “terminado· por el sistema por la cantidad de memoria consumida.

Para el caso de 15-puzzel y 24 puzzle usando la heurísticas PDBs aditivas, se observó en comparación a usar la heurística Manhattan que encontrar la solución en estados “fáciles es más rápido” y se encontró la solución a estados que con la heurística Manhattan no se logró, esto es porque el consumo de memoria es más eficiente, aunque para ciertos estados el tiempo de ejecución supera los 15 minutos.

En cuanto al cubo de rubik usando tomando el máximo PDBs de 8 esquinas y 2 bordes, se encontró solución para estados que no requieren muchos movimientos, en tiempos aceptables,

rondando los 10 minutos, para estados complejos no se llegó a la solución por limitaciones de memoria.

El análisis de topspin es muy similar al del cubo de rubik, solo que para este caso encontrar solución para estados poco complejos requiere menos tiempo, y esto se puede ver evidenciado en el árbol de búsqueda, en donde el factor de ramificación de ambos problemas es similar.

Finalmente para las torres de Hanoi se encontró solución para la mayoría de los estados fuesen complejos o no (exceptuando torres de Hanoi 4-18, en donde no se encontró solución para algunos estados complejos, esto por limitación de memoria), el tiempo para encontrar solución para estados faciles no superó los 5 minutos y para estados más complejos, el tiempo generalmente era menor a los 15 minutos.

Ejecución: Para ejecutar los algoritmos de búsqueda informados debemos ir a la carpeta “problems”, la cual tiene carpetas con los nombres de cada problema, a su vez estas carpetas contienen una subcarpeta llamada PDB(excepto la carpeta n-puzzels, la cual el nombre de la subcarpeta es “15-puzzels-PDB”, “24-puzzels-PDB”, “15-puzzels-Man”, dependiendo del caso), estas carpetas contienen los archivos necesarios para compilar y ejecutar los algoritmos.

Para la ejecución usar los siguientes comandos:

```
./nombrePDB.sh
```

Este comando genera las heurísticas PDBs, en donde “nombre” es el nombre del archivo.hs en la carpeta.

```
make -f makeSearch.mk nombre.<algorithm>
```

Donde <algorithm> puede ser aStar para el algoritmo A* o idastar para el algoritmo IDA*, y “nombre” es el nombre del problema.

```
./15-puzzles.<algorithms> <testFile>
```

Donde <testFile> es el nombre del archivo de prueba, el cual debe estar en un archivo .txt (un estado por línea).

Para el caso de la carpeta “15-puzzels-Man”, no es necesario usar el primer comando.