2 Parte 2

2.1 Respuesta 1

2.1.1 a)

Para que una heurística sea consistente, se deben cumplir las siguientes dos propiedades

$$h(s) = 0 \qquad \text{para todo } s \in G \tag{1}$$
$$h(s) \leq c(s, s') + h(s') \qquad \text{para todo vecinos}' \text{ de } G \tag{2}$$

Para el demostrar (1), tenemos que en el puzzle de 15, un estado final corresponde a aquel en que todas las piezas se encuentran en la posición representada por su número. Además, la heurística de la distancia de Manhattan para este tipo de puzzles corresponde a la suma de las distancias vertical + horizontal de cada pieza con su posición correcta. la Si el puzzle se encuentra en un estado $s \in G$, entonces para cada pieza, la distancia a su posición correcta será 0, y la suma total será 0, por lo que $h(s) = 0$, y se comprueba que se cumple (1) para la heurística de Manhattan.

Para demostrar (2), sea s un estado cualquiera del problema. Los sucesores de s' de s corresponden a aquellos donde se intercambia el espacio en blanco por alguna de sus piezas vecinas. Dado que se realiza solamente un intercambio, el costo de pasar de un estado s a s' será siempre 1, es decir, $c(s, s') = 1$. De esta manera, se pueden dar solamente 2 casos:

- La pieza intercambiada se acerca a su objetivo. En este caso, $h(s') < h(s)$, pero dado que el intercambio se realiza con una pieza adyacente, y el movimiento solo ocurre en una dimensión (o en otras palabras, el la distancia a la nueva posición de la pieza es 1), el valor de la heurística a lo más pudo haber disminuido en 1. Entonces $h(s) = c(s, s') + h(s')$, lo cual es equivalente a $h(s) \leq c(s, s') + h(s')$.
- La pieza intercambiada se aleja de su objetivo. En este caso, aumenta el valor de la heurística, es decir, $h(s') > h(s)$, y entonces, $h(s) \leq c(s, s') + h(s')$

Por lo tanto, siempre se cumple la propiedad (2) para esta heurística. □

2.1.2 b)

En el algoritmo A*, cada vez que sacamos un nodo de la OPEN para expandirlo, si usamos una heurística consistente, podemos asegurar que logramos llegar al nodo con el menor costo posible, por lo tanto, bajo ninguna circunstancia es necesario volver a expandirlo para llegar a una solución.

Según la propiedad (2) de una heurística consistente, se debe cumplir que $h(s) \leq c(s, s') + h(s')$ para cada s' vecino de s. Además, en A* se define a la función f como $f(s) = g(s) + h(s)$, donde $h(s)$ es el valor de la heurística (el cual es estático), y $g(s)$ es el costo acumulado del camino actual hasta el nodo s. Notar que el valor de $g(s)$ puede cambiar a lo largo de la ejecución, si encontramos un camino más corto a dicho nodo.

Si expandimos un nodo cualquiera s, significa que este lo elegimos porque tenía el menor valor posible para f de la OPEN. También, dado que cualquier otro nodo expandido previamente a s fue expandido con esta regla, significa que encontramos el mejor camino posible desde el inicio hasta s, por lo tanto, $g(s)$ no podría tomar un valor menor. Si existiera un camino más corto hacia s (es decir, con nodos con un costo menor), ya se habrían encontrado, y por lo tanto, expandido, antes de expandir s. Esto, debido a que según la propiedad (2) de una heurística consistente, $h(s) \leq c(s, s') + h(s')$.

Si tuvieramos una heurística no consistente, no es necesario que se cumpla la regla mencionada, y se puede dar que $h(s) > c(s, s') + h(s')$, lo que daría a lugar que posteriormente a ser expandido un estado, se puede encontrar un camino mejor hacia el, provocando que se deba expandir un estado múltiples veces para encontrar una solución óptima (aunque esto también va ligado a la implementación que se tenga en código).

2.2 Respuesta 2

La siguiente tabla resume las expansiones que hace cada variante del algoritmo en los problemas del archivo `light_problems.txt`. Se siguió lo del enunciado, y se ejecutaron los *tests* para la heurística de Manhattan dividida por 15 (`div_15`), y la nueva heurística propuesta (`light`). Los valores de la tabla corresponden a la cantidad de expansiones realizadas para encontrar la solución. Para aquellas casillas en donde no hay un valor $(-)$, significa que no se pudo resolver el test en una cantidad razonable de tiempo, por lo que se detuvo la ejecución. También, se debe tener en cuenta que si bien para mayor w en general disminuyen las expansiones, no significa que el algoritmo sea mejor, debido a que el general también disminuye la calidad de la solución.

w	1		1.5		2		3		5	
puzzle	light	div_15	light	div_15	light	div_15	light	div_15	light	div_15
1	29	183	12	30	12	30	12	12	12	12
2	341	2545	235	442	106	442	123	115	215	161
3	758	5418	114	893	36	893	13	39	11	12
4	2487	18432	261	3068	60	3068	18	67	9	17
5	1921	12430	239	2297	59	2297	14	75	12	13
6	1472	10144	162	1805	48	1805	17	62	11	17
7	5549	37507	663	6746	155	6746	16	180	11	15
8	3515	25556	419	4397	85	4397	40	107	62	43
9	2922	20998	362	3609	69	3609	14	98	10	13
10	7722	48441	955	9422	224	9422	46	238	11	41
11	2268	15426	287	2797	81	2797	20	92	15	21
12	6357	67142	437	8261	140	8261	131	133	339	139
13	12659	90711	1028	15886	134	15886	31	197	29	34
14	6777	59631	669	8631	120	8631	26	179	34	27
15	12906	128291	968	16682	299	16682	75	341	21	59
16	39116	318443	9558	49048	4112	49048	3005	4494	7334	3115
17	170909	1784893	10807	220251	1317	220251	71	1799	16	46
18	-	-	389287	-	23648	11561332	3344	36877	690	2143
19	-	-	287419	-	25372	-	967	36267	398	519
20	-	-	191056	-	19877	-	485	28204	31	281
21	-	-	-	-	576331	-	51269	910424	7462	70701
22	-	-	-	-	64363	-	804	99219	433	652
23	-	-	-	-	6500	-	201	9191	138	123
24	-	-	-	-	14812	-	312	21326	88	207
25	-	-	-	-	605423	-	18616	822230	224	10371
26	-	-	-	-	10140	-	237	15208	1214	200
27	-	-	-	-	69815	-	10062	122644	2138	10505
28	-	-	-	-	129416	-	6401	159668	360	3170
29	-	-	-	-	332309	-	36449	507022	1644	20798