Trabajo Práctico Nº1

Métodos de Búsqueda Desinformado e Informados

Ribas, Ignacio
Marchetti, Gianfranco

Estructuras de Datos

Estructuras de Datos

Node

```
Node * p;
std::list<Node *> children;
State state;
int g;
double (*f)(Node *);
```

Estructuras de Datos

State

```
std::vector<std::pair<int, int>> boxes;
std::pair<int, int> user;
```

Costo

Costo

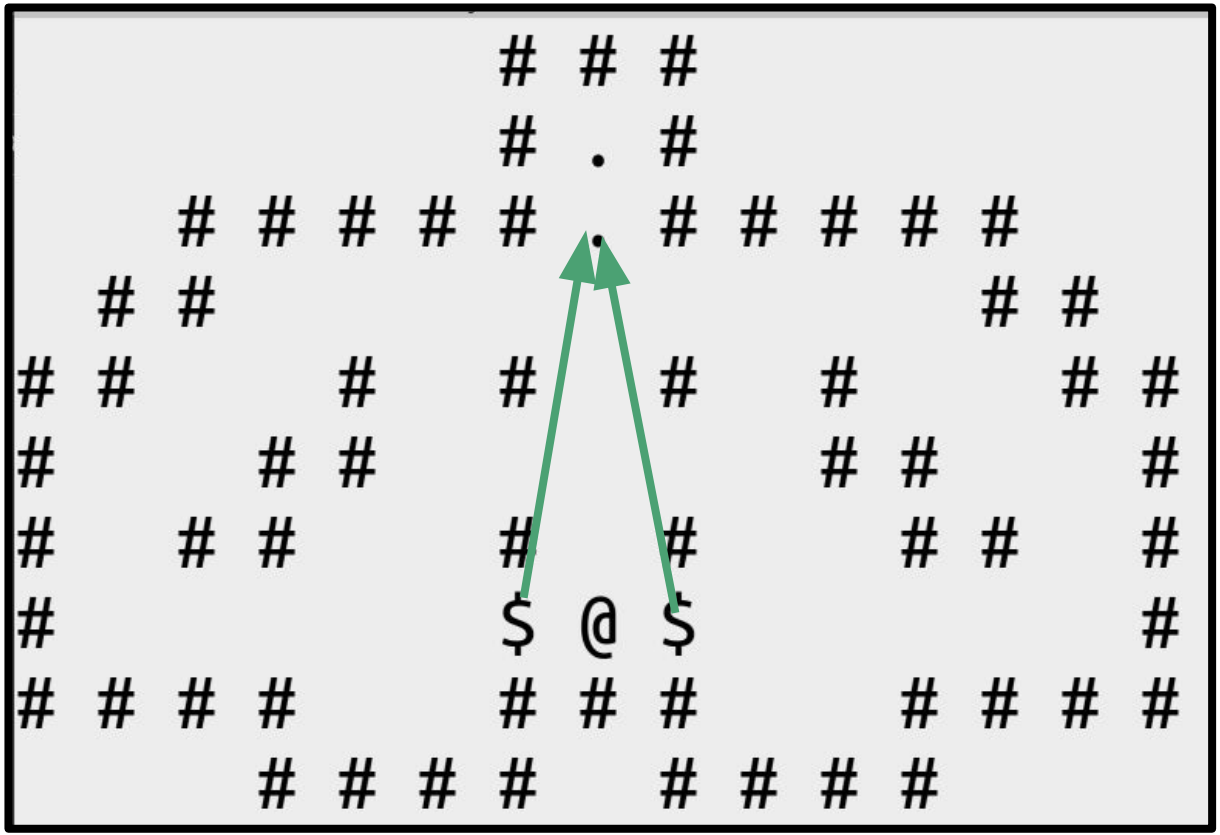
- Se incrementa en una unidad si @ se mueve empujando a **$**.
- Se incrementa en dos unidades si @ se mueve libre.

Profundidad

- Indica la profundidad en la cual fue hallada la solución.
- Coincide con el costo si el mismo fuera **unitario y uniforme**

Heurísticas

Heurística 0



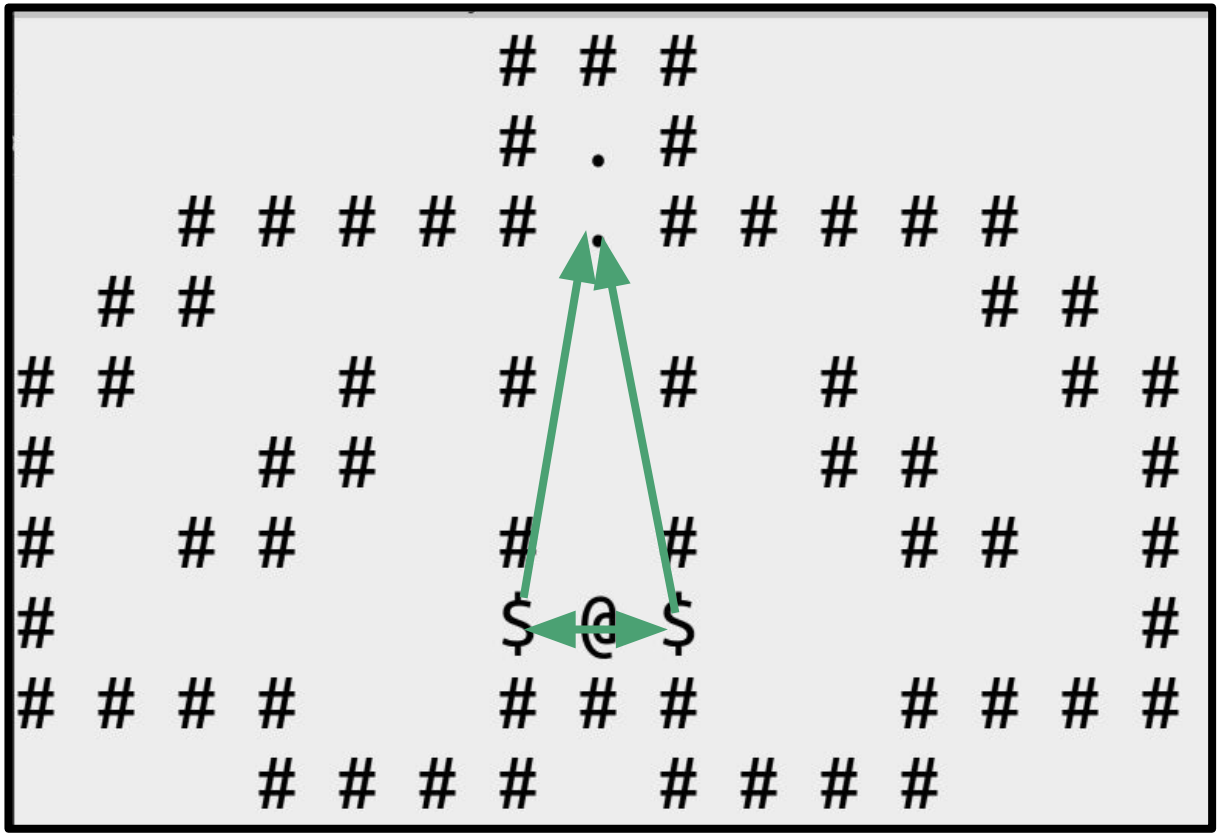
- Toma la distancia Manhattan de cada **$** hacía él ● más cercano al mismo, y las suma.

ADMISIBLE

Heurística 0

```
def h(state):
    sum = 0
    for box in state.boxes:
        mini = sys.maxsize
        for goal in goalStates:
            md = abs(goal[0]-box[0]) + abs(goal[1]-box[1])
            if md < mini:
                mini = md
        sum += mini
    return sum
```

Heurística 1



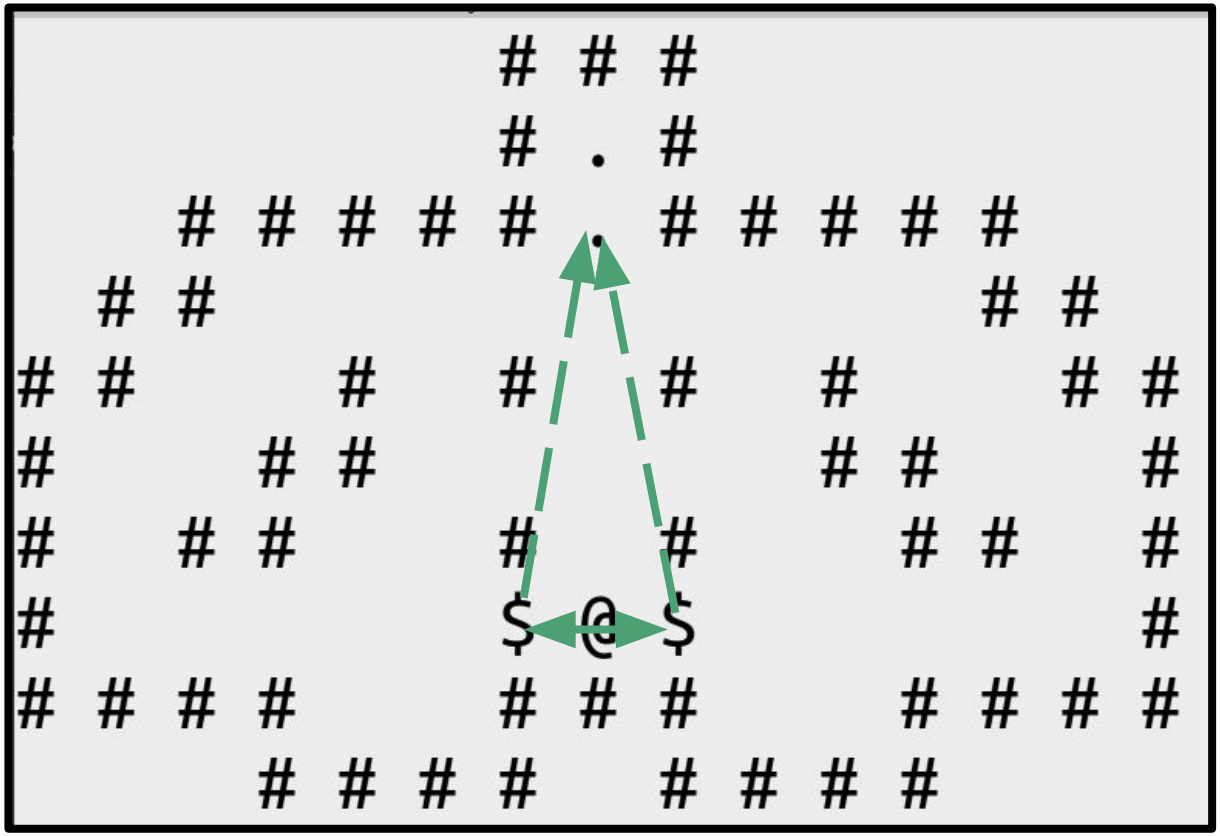
- Toma la Distancia Manhattan de cada **$** hacía él ● más cercano al mismo, y las suma.
- Le suma la Distancia Manhattan más chica entre @ y algún **$**

ADMISIBLE

Heurística 1

```
def h(state):
    sum = 0
    for box in state.boxes:
        mini = sys.maxsize
        for g in gS:
            md =  abs(g[0]-box[0]) + abs(g[1]-box[1])
            if md < mini:
                mini = md
        sum += mini
    sum += min([abs(b[0]-state.user[0]) + abs(b[1]-state.user[1])
for b in state.boxes])
    return sum
```

Heurística 2



- Calcula para cada **$** la distancia lineal a algún ●.
- Suma esas distancias.
- Suma la distancia máxima entre @ y algún ●

NO ADMISIBLE

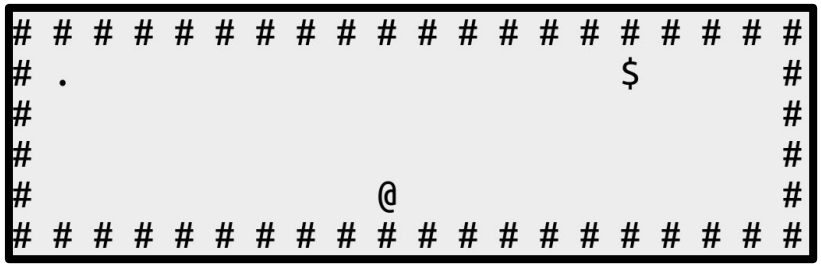
Heurística 2

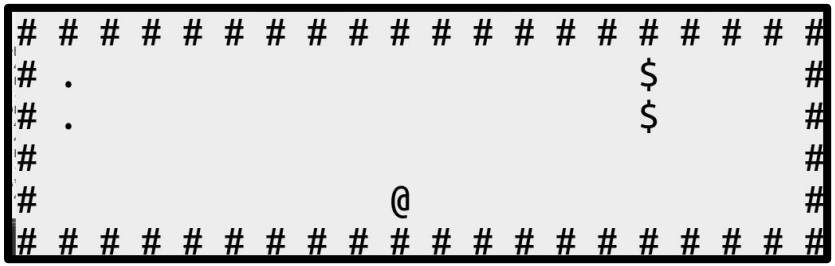
```
            def h(state):
                for box in state.boxes:
                    max = 0
                    for g in gS:
                        md =  abs(g[0]-box[0])**2 + abs(g[1]-box[1])**2
                        if md > max:
                            max = md
                    sum += max
                sum += max([abs(b[0]-state.user[0]) + abs(b[1]-state.user[1])
for b in state.boxes])**2
                return sum
        self.h = h
```

Testing

Mapas

easy



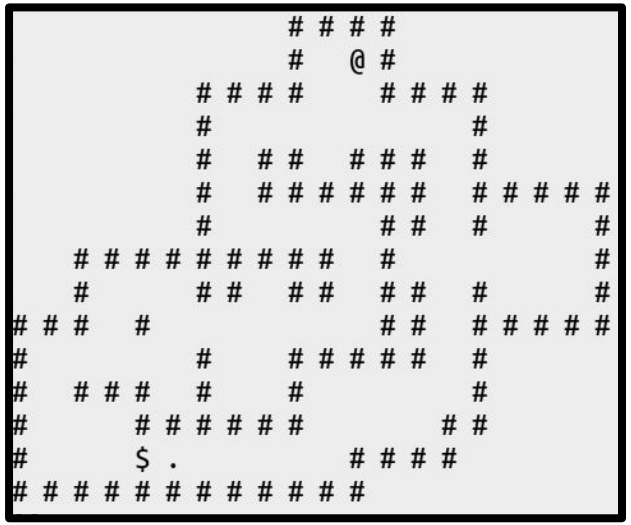


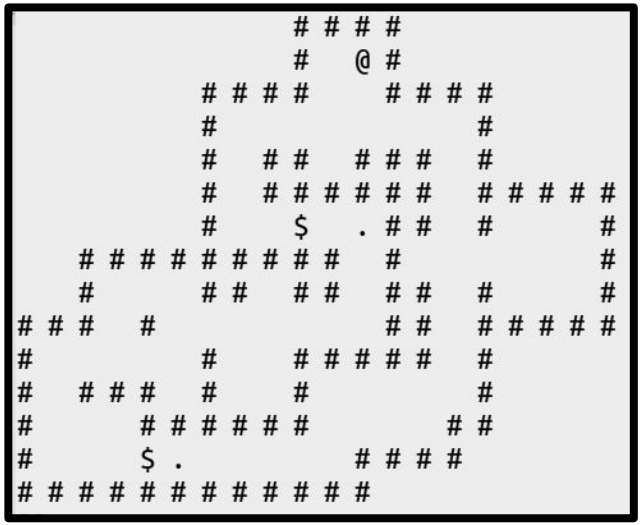
- Mayor libertad de movimiento para @, en la mayoría de las posiciones se puede mover en las 4 direcciones
- Poco recorrido de **$** en **.**
- Menor cantidad de steps

Mapas

long

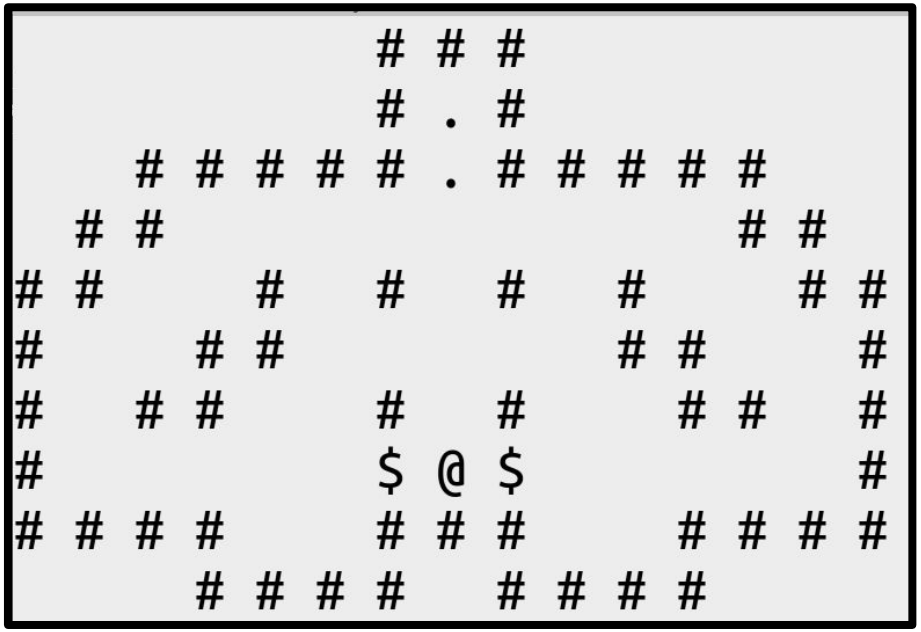
- Menor libertad de movimiento para **@**
- Dificultad media para colocar **$** en **.**
- Mayor cantidad de steps



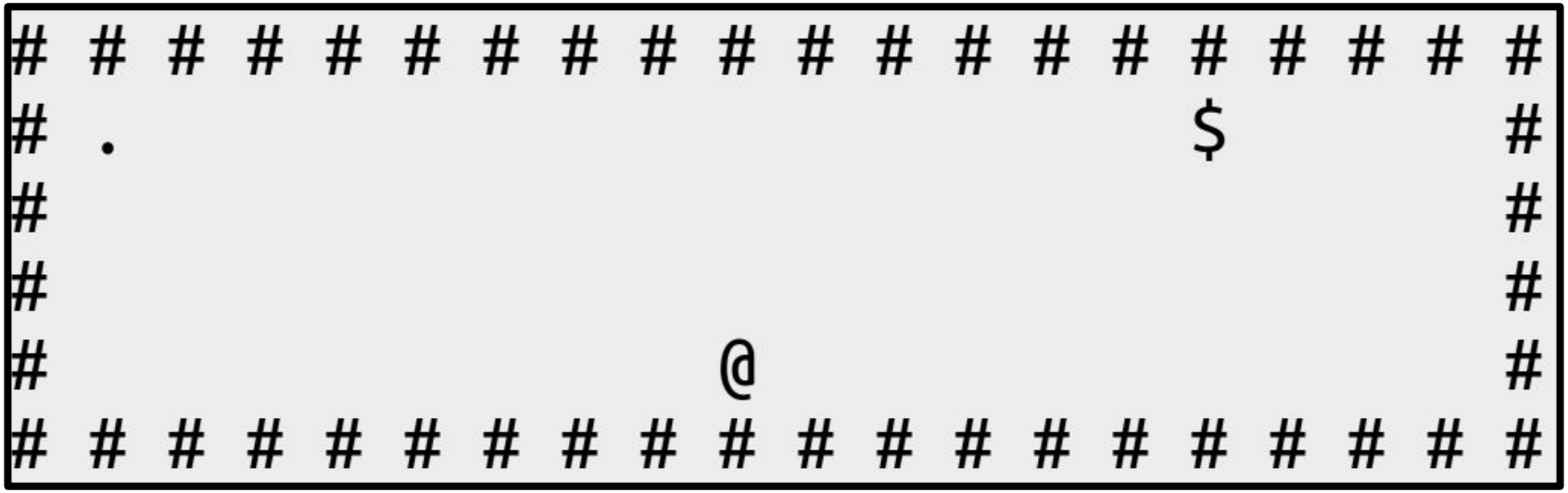


Mapas

map1



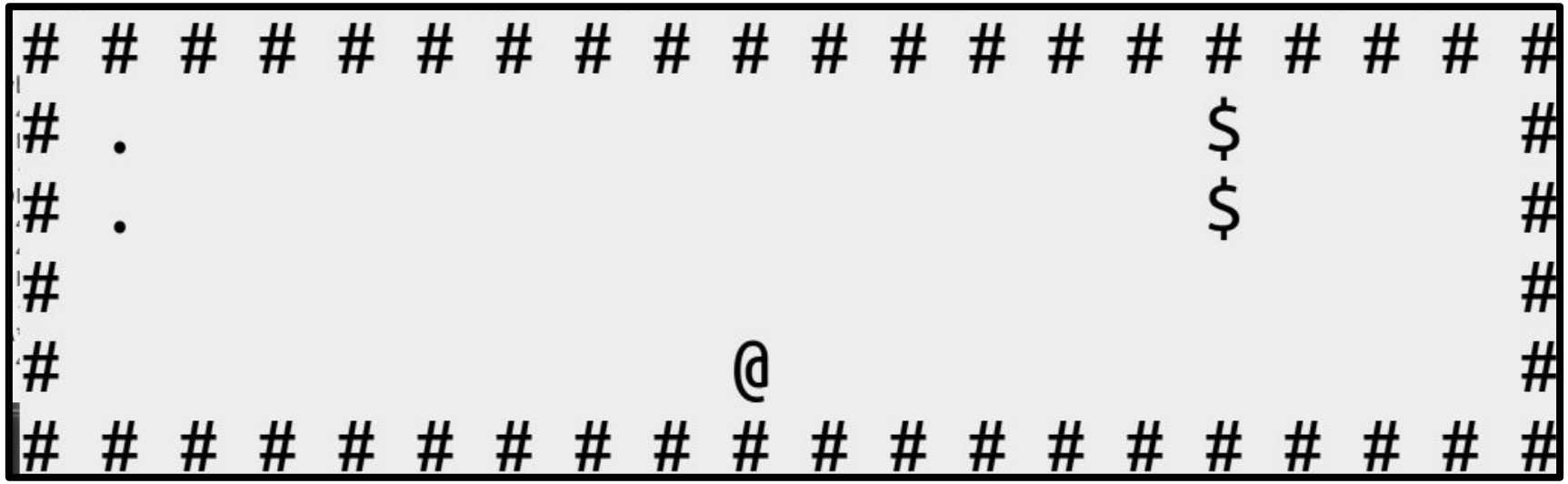
- Mapa provisto por la cátedra
- Intermedio entre *easy* y *long*



mapa: easy

Algoritmo	Costo Total	Profundidad	Nodos Explorados	Frontera	Tiempo (seg)
No Informados					
BFS	**34**	25	781	85	0,3312
DFS	42	29	1138	25	0,7565
IDDFS^	42	29	1042	25	0,6486
Informados					
Heuristica 0					
A*	**34**	25	121	35	0,0150
GG	**34**	25	106	48	0,0158
IDA*	**34**	25	121	9	0,0579
Heuristica 1					
A*	**34**	25	100	38	0,0101
GG	**38**	27	73	44	0,0048
IDA*	**34**	25	100	3	0,0507
Heuristica 2					
A*	**34**	25	121	35	0,0148
GG	**34**	25	106	48	0,0103
IDA*	**34**	25	121	9	0,0561
^ Profundidad por iteracion = 50					

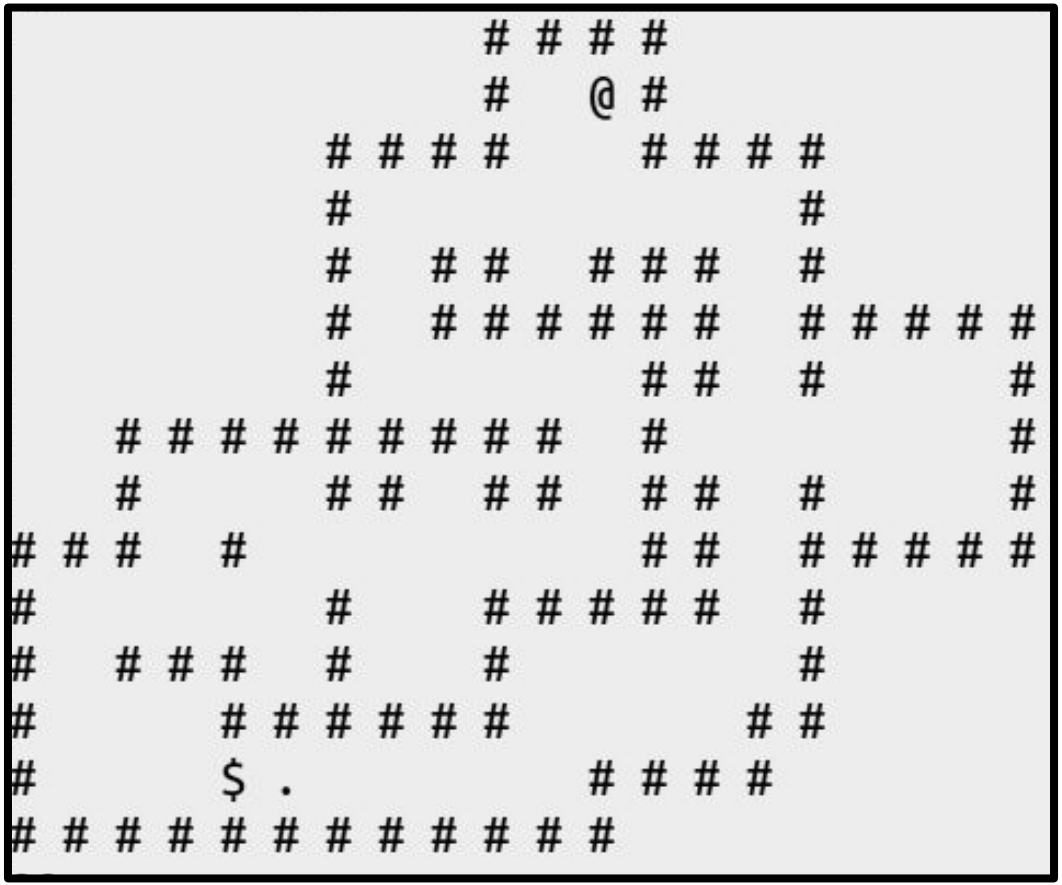
mapa: easy - servidor: pampero



mapa: easy2

Algoritmo	Costo Total	Profundidad	Nodos Explorados	Frontera	Tiempo (seg)
No Informados					
BFS	**76**	53	67201	729	2979,9317
DFS	1760	908	14206	833	**102,7325**
IDDFS^	**102**	66	15500	58	**104,8851**
Informados					
Heuristica 0					
A*	**76**	53	50796	4649	**1394,6843**
GG	**80**	55	1205	100	**0,9484**
IDA*	none	none	none	none	>3600
Heuristica 1					
A*	**76**	53	50796	4649	**1393,8097**
GG	**88**	59	9424	988	**46,6149**
IDA*	none	none	none	none	>3600
Heuristica 2					
A*	**76**	53	50796	4649	**1397,1097**
GG	**80**	55	1205	100	**1,0248**
IDA*	none	none	none	none	>3600
^ Profundidad por iteracion = 50					

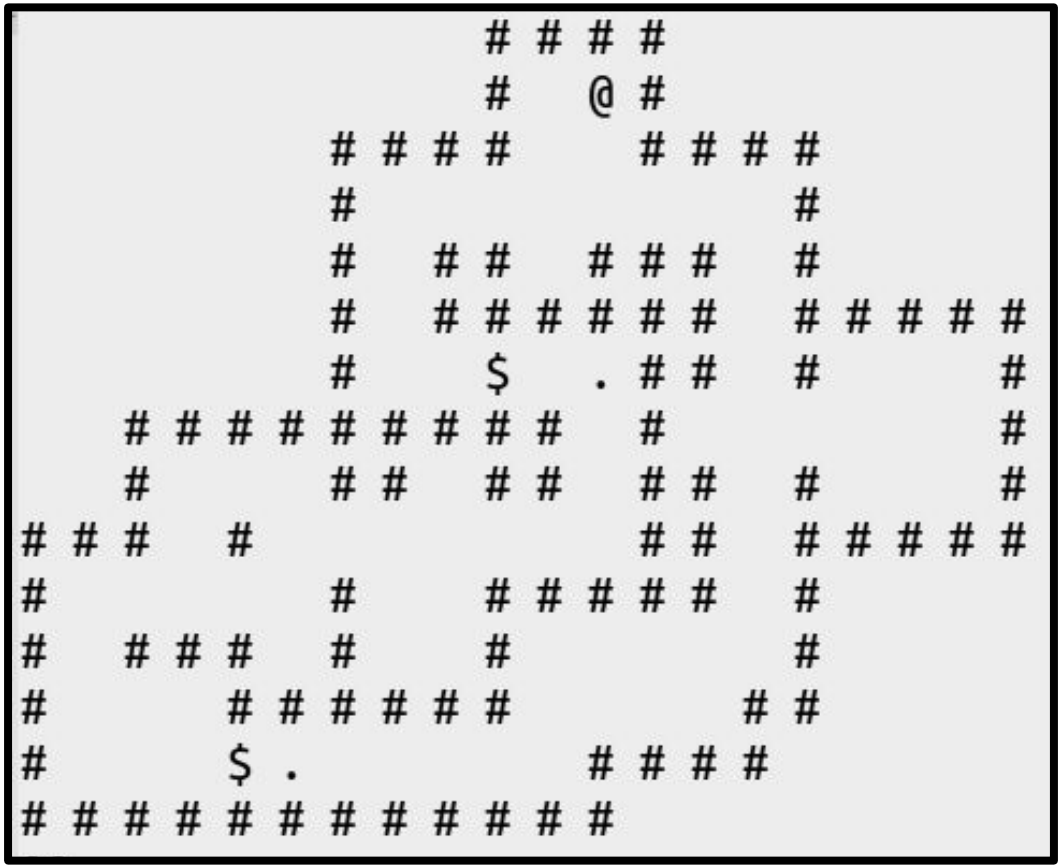
mapa: easy2 - servidor: pampero



mapa: long - servidor: pampero

Algoritmo	Costo Total	Profundidad	Nodos Explorados	Frontera	Tiempo (seg)
No Informados					
BFS	**65**	34	117	6	0,0072
DFS	159	82	127	22	**0,0071**
IDDFS^	**65**	36	135	11	0,0080
Informados					
Heuristica 0					
A*	**61**	34	108	7	0,0062
GG	**65**	34	86	1	0,0052
IDA*	**61**	34	108	4	0,0152
Heuristica 1					
A*	**61**	34	101	8	0,0058
GG	**65**	34	50	6	0,0021
IDA*	**61**	34	101	6	0,0178
Heuristica 2					
A*	**61**	34	108	7	0,0063
GG	**65**	34	86	1	0,0049
IDA*	**61**	34	108	4	0,0152
^ Profundidad por iteracion = 50					

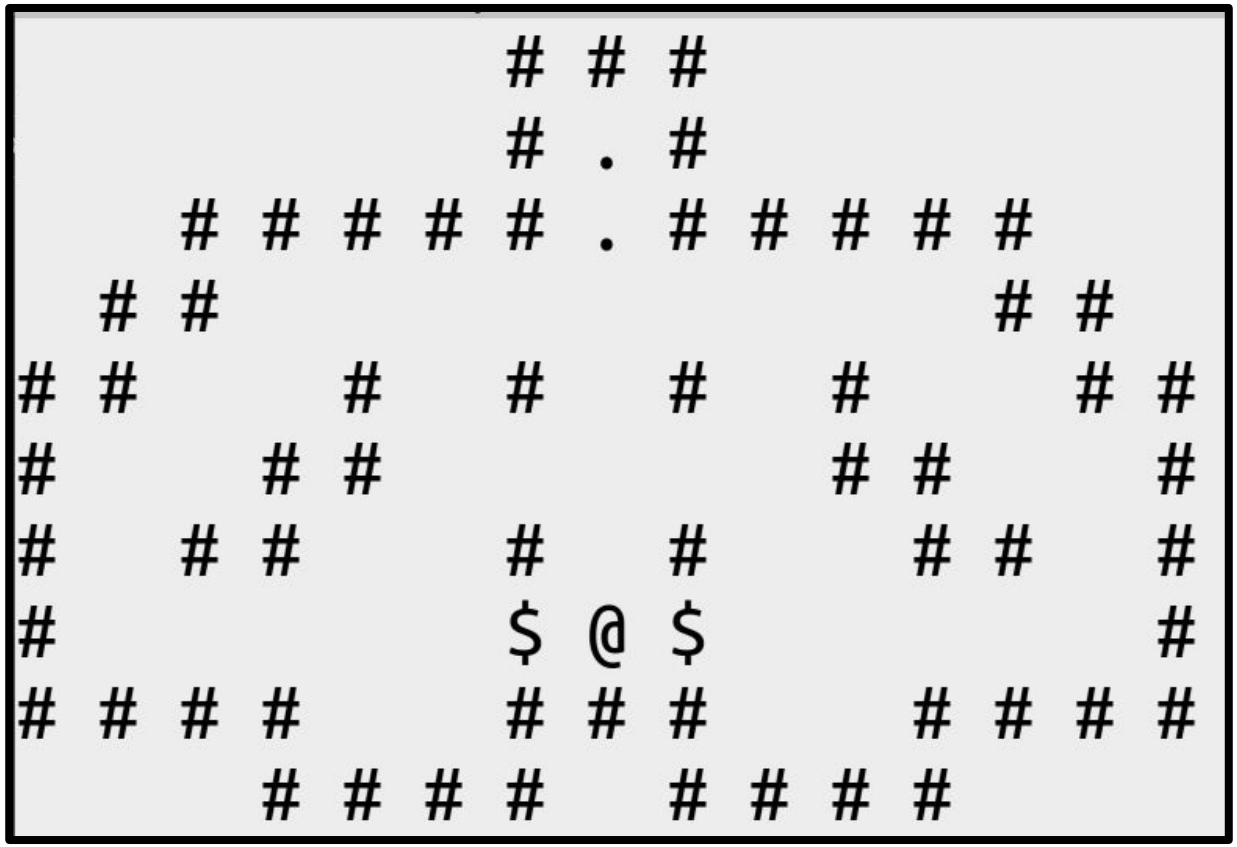
mapa: long - servidor: pampero



mapa: long2 - servidor: pampero

Algoritmo	Costo Total	Profundidad	Nodos Explorados	Frontera	Tiempo (seg)
No Informados					
BFS	**99**	54	617	35	0,1635
DFS	195	**102**	214	25	0,0226
IDDFS^	**99**	54	1123	14	0,5395
Informados					
Heuristica 0					
A*	**99**	54	522	29	0,1201
GG	**99**	54	259	17	0,0350
IDA*	**99**	54	537	32	0,2858
Heuristica 1					
A*	**99**	54	474	35	0,1007
GG	**99**	54	105	11	0,0076
IDA*	**99**	54	474	33	0,3389
Heuristica 2					
A*	**99**	54	522	29	0,1158
GG	**99**	54	259	17	0,0335
IDA*	**99**	54	537	32	0,2799
^ Profundidad por iteracion = 50					

mapa: long2 - servidor: pampero



mapa: map1 - servidor: pampero

Algoritmo	Costo Total	Profundidad	Nodos Explorados	Frontera	Tiempo (seg)
No Informados					
BFS	**139**	79	15906	19	100,9906
DFS	2491	1297	4526	522	**6,5951**
IDDFS^	303	165	15921	58	**114,6886**
Informados					
Heuristica 0					
A*	139	79	15888	31	**104,9507**
GG	**201**	111	822	75	**0,2760**
IDA*	none	none	none	none	>3600
Heuristica 1					
A*	**139**	79	15818	46	100,6220
GG	**205**	115	1970	366	**1,2844**
IDA*	none	none	none	none	>3600
Heuristica 2					
A*	139	79	15888	31	**101,1652**
GG	**201**	111	822	75	**0,2764**
IDA*	none	none	none	none	>3600
^ Profundidad por iteracion = 50					

mapa: map1 - servidor: pampero

Datos

Incremento en una caja

Algoritmo	Tiempo - easy (seg)	Tiempo - easy2 (seg)	Incremento(veces)
No Informados			
BFS	0,3312	2979,9317	8996,2647
DFS	0,7565	**102,7325**	135,8081
IDDFS^	0,6486	**104,8851**	161,7210
Informados			
Heuristica 0			
A*	0,0150	**1394,6843**	93246,2571
GG	0,0158	**0,9484**	59,9809
IDA*	0,0579	>3600	none
Heuristica 1			
A*	0,0101	**1393,8097**	138329,6645
GG	0,0048	**46,6149**	9623,2234
IDA*	0,0507	>3600	none
Heuristica 2			
A*	0,0148	**1397,1097**	94151,2053
GG	0,0103	**1,0248**	99,1739
IDA*	0,0561	>3600	none

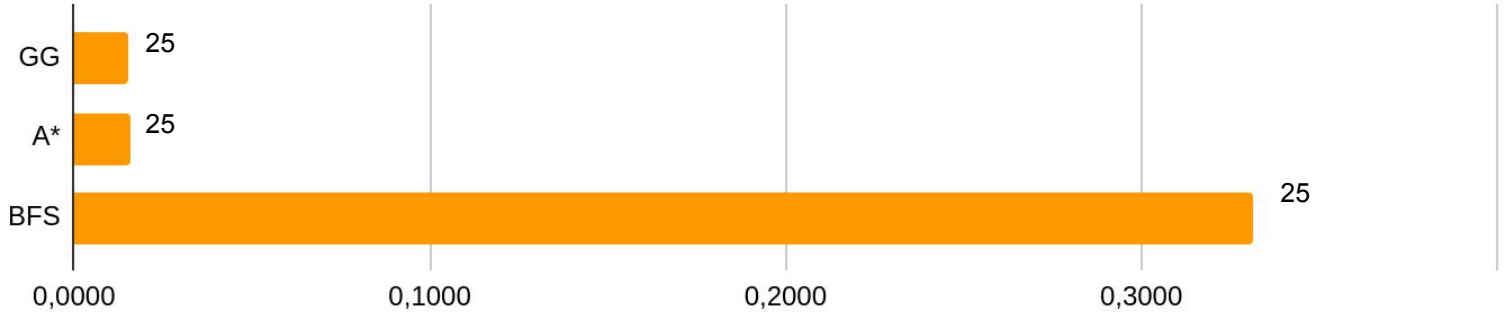
Datos

Incremento en una caja

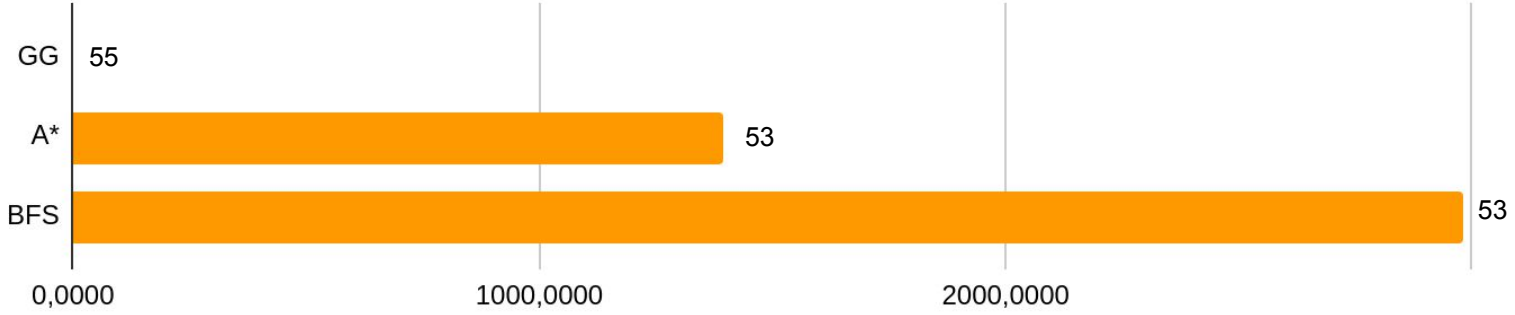
Algoritmo	Tiempo - long (seg)	Tiempo -long2 (seg)	Incremento(veces)
No Informados			
BFS	0,0072	0,1635	22,5615
DFS	**0,0071**	0,0226	3,1838
IDDFS^	0,0080	0,5395	67,6473
Informados			
Heuristica 0			
A*	0,0062	0,1201	19,3185
GG	0,0052	0,0350	6,7938
IDA*	0,0152	0,2858	18,7760
Heuristica 1			
A*	0,0058	0,1007	17,4868
GG	0,0021	0,0076	3,5670
IDA*	0,0178	0,3389	19,0853
Heuristica 2			
A*	0,0063	0,1158	18,2721
GG	0,0049	0,0335	6,8586
IDA*	0,0152	0,2799	18,4671

Conclusión

Conclusión



mapa: easy



mapa: easy2

Conclusión

Performance de GG - Costo

easy	+0%
easy2	+3,8%
long	+0%
long2	+0%
map1	+40,5%

Conclusión

A*

- Mejor opción si lo prioritario es el costo

GG

- Mejor opción si lo prioritario es el tiempo