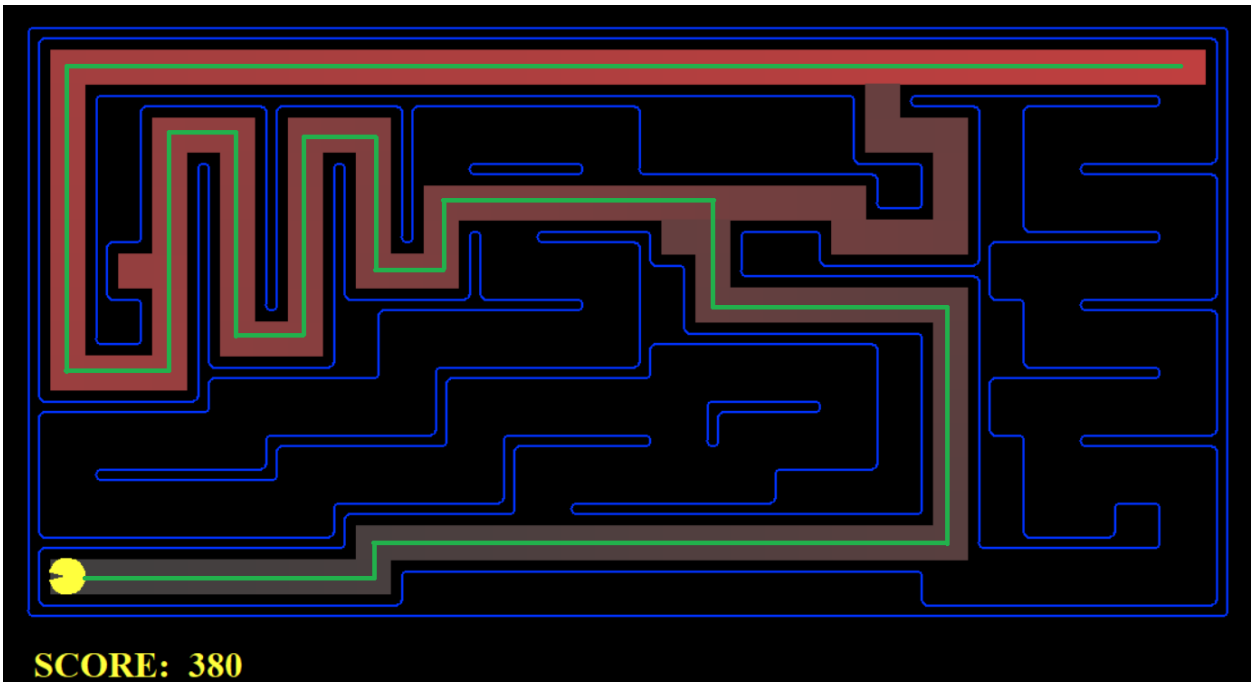
موژان میرجلیلی - 9831140

**1. سوال:** آیا ترتیب کاوش همان ترتیبی بود که انتظار داشتید؟ آیا پکمن در راه رسیدن به هدف، به همه مربع های کاوش شده می رود؟

بله، مشاهده می‌شود که ابتدا عمق جست‌وجو شده است درست مانند الگوریتم DFS. (بهینه نیست)

خیر،



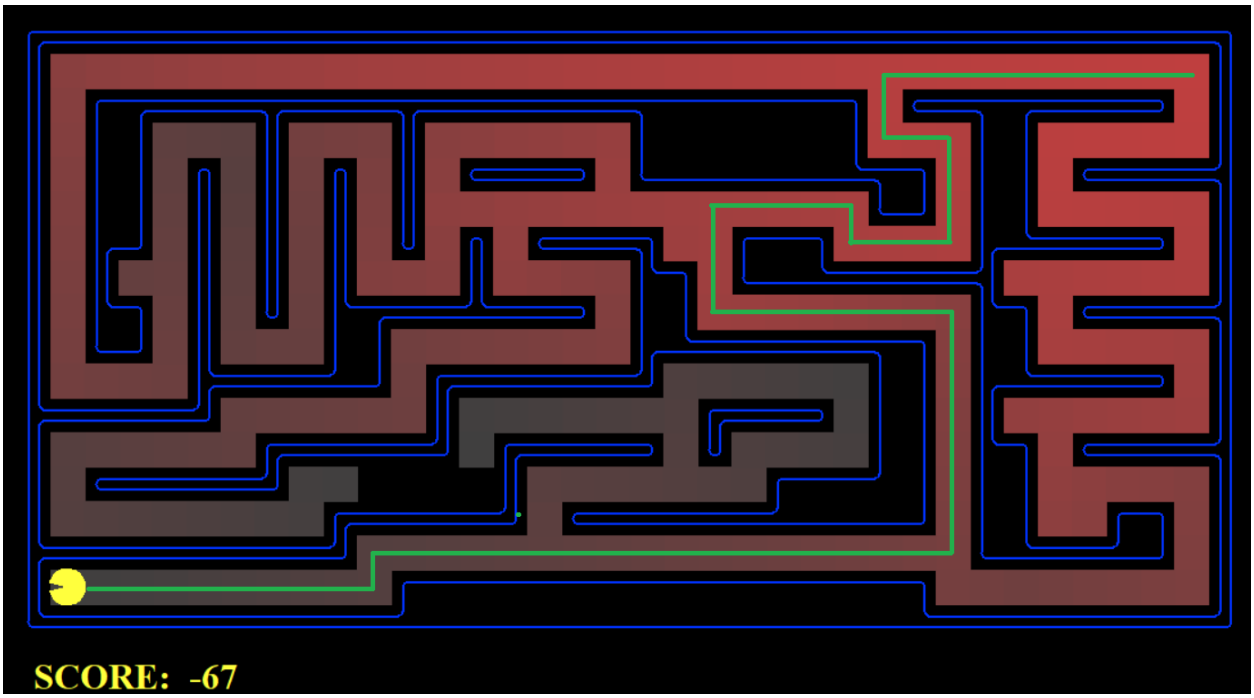

```
C:\Users\Moujan\Desktop\AI_P1\search>python pacman.py -l mediumMaze -p SearchAgent
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 130 in 0.0 seconds
Search nodes expanded: 146
Pacman emerges victorious! Score: 380
Average Score: 380.0
Scores:        380.0
Win Rate:      1/1 (1.00)
Record:        Win
```

**2. سوال:** آیا این راه حل کمترین هزینه را دارد؟ اگر نه فکر کنید که جستجوی اول عمق چه کاری را اشتباه انجام می دهد.

خیر، DFS ابتدا عمق را جست‌وجو می‌کند و اگر جوابی(غیر بهینه) را در عمق زودتر بیابد و به هدف برسید، همان را استفاده می‌کند که بهینه نیست.

**3. سوال:** آیا الگوریتم جستجوی اول سطح راه حل با کمترین هزینه را پیدا می کند؟ اگر نه پیاده سازی خود را چک کنید.

بله، BFS در سطح (کمترین عمق) جواب را می‌یابد که کمترین هزینه را نیز خواهد داشت.




```
C:\Users\Moujan\Desktop\AI_P1\search>python pacman.py -l mediumMaze -p SearchAgent -a fn=bfs
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 269
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win
```

**4. سوال:** الگوریتم های جستجویی که تا به این مرحله پیاده سازی کرده اید را روی openMaze اجرا کنید و توضیح دهید چه اتفاقی می افتد.

DFS: تمام مسیر به دلیل جست‌وجو عمقی به صورت طبقه‌ای کاوش می‌شود که بهینه نیست و در یافتن هدف تاثیری ندارد.

BFS: به دلیل این که سطحی جست‌وجو می‌کند و هدف در عمقی‌ترین لایه وجود دارد، کل مسیر پیمایش می‌شود و هزینه بیش‌تری دارد.

UCS: مشابه BFS است چون هزینه همه خانه‌ها یک است.

*A: مشابه 2 مورد قبلی با این تفاوت که خانه‌های کمتری برای رسیدن به هدف کاوش می‌شوند. (به دلیل استفاده از هیوریستیک) در نتیجه بهترین عملکرد را دارد.

**5. سوال:** هیوریستیک خود را توضیح دهید و سازگاری آن را استدلال کنید.

بررسی هیوریستیک گوشه ها: در این جا با استفاده از فاصله منهتن، فاصله تا 4 گوشه را یافته و هزینه را برای رسیدن به نزدیک‌ترین را بررسی می‌کنیم و به نقاط بعدی می‌رویم.

این هیورسیتیک به دلیل این که در هر مرحله از مسیر رسیدن ما به گوشه کمتر است، سازگار است. زیرا هزینه تخمینی که با استفاده از منهتن است قطعا از هزینه واقعی کمتر است.

**6. سوال:** هیوریستیک خود را توضیح دهید و سازگاری آن را استدلال کنید.

بررسی هیوریستیک غذا: فاصله عامل تا تمامی غذاها را با استفاده از mazeDistance به دست می‌آوریم و در لیستی قرار می‌دهیم. mazeDistance از BFS استفاده می‌کند و اینگونه وجود دیوارها نادیده گرفته نمی‌شود. حال ماکسیمم فاصله تا غذا (دورترین) را برمی‌گردانیم.

این هیورسیتیک به دلیل این که همیشه فاصله را تا یک غذا (دورترین) را برمی‌گرداند و هزینه واقعی ما هزینه خوردن همه غذاها است پس قطعا همیشه کمتر یا مساوی (یک غذا یا همه غذاها در راستای فاصله دورترین غذایی که برمی‌گردانیم باشند) است و در نتیجه سازگار است.

**7. سوال:** ClosestDotSearchAgent شما، همیشه کوتاه‌ترین مسیر ممکن در ماز را پیدا نخواهد کرد. مطمئن شوید که دلیل آن را درک کرده اید و سعی کنید یک مثال کوچک بیاورید که در آن رفتن مکرر به نزدیک‌ترین نقطه منجر به یافتن کوتاه‌ترین مسیر برای خوردن تمام نقاط نمی شود.

با توجه به الگوریتم agent ما همیشه به نزدیک‌ترین نقطه می‌رود و در نتیجه کوتاه‌ترین مسیر ممکن است به دست نیاید و هزینه افزایش یابد.

مثال، در این جا به جای اینکه ابتدا نقطه بالا سمت راست را بخورد سمت نقاط نزدیک تر سمت چپ می‌رود و در نهایت دوباره مجبوریم به آن نقطه برگردیم که باعث می‌شود مسیر اضافه‌ای طی شود.