به نام خدا

هوش مصنوعی

تمرین کامپیوتری شماره 1

علی مهرانی - 810198542

# National Pizza Day

## ● نحوه مدل کردن مسئله

ابتدا دو کلاس به نام UniGraph و NPDState تعریف میکنیم که کلاس UniGraph شامل اطلاعات کلی گراف مسئله و NPDState نیزشامل یک شی از گراف و وضعیت ان است.

- کلاس UniGraph شامل
  - آرایه یال های گراف
  - تعداد رئوس گراف
  - آرایه یال های لق
  - وضعیت عبور از یال های لق
  - اطلاعات دانشجویان
  - اولویت بندی تحویل پیتزا
  - آرایه پیتزا فروشی ها
  - راس فعلی در گراف

- کلاس NPDState شامل
  - یک شی از گراف
  - مجموعه دانشجویانی که غذای خود را دریافت کرده اند
  - مسیر طی شده تا کنون (رئوسی که زیرج از ان عبور کرده)
  - پیتزا انتخاب شده برای تحویل
  - تعداد قدم های طی شده (معادل هزینه مسیر طی شده)
  - زمان صرف شده بر روی یال های لق

- Initial State

  حالت ابتدایی است که نقش ورودی مسیله را دارد که با نقطه شروع زیرج مشخص میشود.

- Goal State

  در state ای که تمامی پیتزاها یه دانشجویان تحویل داده شده باشند و موقعیت فعلی ما در گراف موقعیت یک دانشجو باشد که یعنی اخرین دانشجو هم پیتزا خود را در حال حاضر دریافت کرد.

- Actions

1. زیرج میتواند در هر مرحله به یکی از راسهای مجاور برود البته باید لق بودن یال ها را بررسی کند و بسته به آن عمل کند.

2. زیرج باید هر بار بعد از رد شدن از یک یال لق مدتی صبر کند تا بتواند دوباره از آن رد شود که این مدت برای یال های لق مشخص میشود.

3. زیرج میتواند در هر راسی که هست، یک ثانیه دیگر روی همان راس بماند به جای اینکه یک یال دیگر را طی کند.

4. اگر زیرج بخواهد از راسی پیتزا بگیرد، این کار زمانی نخواهد گرفت و میتواند بلافاصله بعد از رسیدن به راس مورد نظر پیتزا را دریافت کند.

5. بعد از رساندن پیتزا به مکان یک دانشجو، دانشجو بلافاصله پیتزا را تحویل میگیرد و این کار زمانی نمیگیرد.

6. اگر زیرج در حال حمل یک پیتزا باشد، تا وقتی که پیتزا را به دانشجوی مورد نظر نرساند، نمیتواند پیتزای جدیدی تهیه کند.

- Cost

1. گذر از یک یال در صورت لق نبودن آن 1 ثانیه زمان میبرد.

2. گذر از یک یال لق در صورتی که قبلا از آن عبور نشده باشد همان 1 ثانیه زمان را احتیاج دارد اما در غیر اینصورت باید به مقدار ذکر شده متناسب هر یال صبر کرد تا بتوان دوباره از آن عبور کرد.

- Transition model

1. با هر بار رفتن از یک راس به راس دیگر زیرج به یاد دارد که کدام پیتزا ها را تحویل داده و اولویت دانشجویان را نیز به یاد دارد.

2. مکان فعلی زیرج با هر بار رفتن از یک راس به راس دیگر update میشود.

- در کلاس UniGraph اطلاعات فایل خوانده میشود و مقادیر مربوط به گراف تنظیم میشود و در کلاس NPDState وضعیت هر گراف و باقی اطلاعات ذکر شده ذخیره میشود.

- **الگوریتم های پیاده سازی شده**

- **BFS**

این الگوریتم یک الگوریتم uniformed است که در آن رئوس بر اساس ارتفاعشان پیمایش میشوند.
زمانی که تمام رئوس یک ارتفاع پیمایش شدند و به goal state نرسیدیم ، پیمایش رئوس ارتفاع بعدی آغاز میشود و این کار تا رسیدن به هدف یا goal state ادامه می یابد.
همچنین در طول اجرای این الگوریتم رئوس بازدید شده ذخیره میشوند تا دوباره بررسی نشوند و حالت تکراری ایجاد نشود.
این الگوریتم کامل است و سرعت آن در مقایسه با IDS بیشتر است اما در مقایسه با *A سرعت کم تری دارد.

- **IDS**

این الگوریتم نیز مانند BFS یک الگوریتم uninformed است و از لحاظ زمانی از BFS کندتر است اما به طور کلی حافظه کمتری نسبت به BFS نیاز دارد.
در اجرای این الگوریتم برای پیمایش گراف در هر عمق از الگوریتم DFS استفاده میشود.

- **A***

این الگوریتم یک الگوریتم informed است که در آن رئوس با هزینه کمتر را پیمایش میکنیم که برای تعیین این هزینه از یک تابع به نام heuristic استفاده میکنیم که مقدار این تابع برای رای n به صورت h(n) بیان میشود که در واقع تخمینی برای هزینه رسیدن به goal از راس n است.
پس از محاسبه heuristic آن را با تابع دیگری به نام g(n) جمع میکنیم و حاصل جمع آن را evaluation function می نامیم که با f(n) آن را نمایش میدهیم و از این تابع به عنوان هزینه برای انتخاب راس با هزینه کم تر برای پیمایش استفاده میکنیم.

- **توضیحات heuristic**

الگوریتم A* زمانی به ما نتیجه بهینه را ارائه میدهد که تابع heuristic مطرح شده آن consistent باشد.
heuristic ما برای داشتن این ویژگی باید میزان هزینه پیش بینی شده بین دو راس توسط آن کمتر مساوی هزینه واقعی بین دو راس باشد.
در این برنامه ما heuristic در هر state را به عنوان تعداد پیتزا های تحویل داده نشده در آن وضعیت بیان میکنیم و چون هزینه واقعی برای ان راس تا رسیدن به هدف همواره بیش تر از این مقدار است پس این تابع consistent است و جواب بهینه تولید میکند.

- **Weighted A***

اجرای این الگوریتم همانند A* است اما برای تعیین f(n) ابتدا مقدار heuristic را در یک ضریب α ضرب میکنیم و مقدار evaluation function به صورت زیر میشود.

$$f(n) = h(n) * \alpha + g(n)$$

در واقع میتوان گفت که A* اصلی یک weighted A* با α=1 است.

- **توضیحات اجرای الگوریتم ها**

الگوریتم ها را بر روی 6 تست اجرا میکنیم که تست 1 تا 3 در کنار صورت پروژه آپلود شده اند و تست 0 نیز در صورت پروژه اورده شده ، تست 4 و 5 نیز دست ساز هستند.

- **نتایج اجرای الگوریتم**
  - در تست هایی که مدت زمان اجرا ان ها طولانی بود از - استفاده شده.

**Test 0**

| | پاسخ مسئله | میانگین زمان اجرا | تعداد استیت های دیده شده | مسیر پاسخ |
|---|---|---|---|---|
| BFS | 9 | 0.0093 | 47 | [1, 2, 3, 4, 4, 4, 3, 2, 3, 5] |
| IDS | 9 | 0.0531 | 281 | [1, 2, 3, 4, 4, 4, 3, 2, 3, 5] |
| A* | 9 | 0.0083 | 41 | [1, 2, 3, 4, 4, 4, 3, 2, 3, 5] |
| A* weighted | 9 | 0.00631 | 34 | [1, 2, 3, 4, 4, 4, 3, 2, 3, 5] |

| | | | | |
|---|---|---|---|---|
| 1 -> 1.5 | | | | |
| A* weighted 2 -> 8 | 9 | 0.00397 | 21 | [1, 2, 3, 4, 4, 4, 3, 2, 3, 5] |

## Test 1

| | پاسخ مسئله | میانگین زمان اجرا | تعداد استیت های دیده شده | مسیر پاسخ |
|---|---|---|---|---|
| BFS | 10 | 1.6368 | 984 | [6, 4, 10, 5, 3, 8, 9, 1, 7, 9, 2] |
| IDS | 10 | 20.185 | 7498 | [6, 4, 10, 9, 7, 9, 2, 9, 1, 3, 8] |
| A* | 10 | 1.3114 | 686 | [6, 4, 10, 9, 1, 3, 8, 9, 7, 9, 2] |
| A* weighted 1 -> 1.5 | 10 | 0.6370 | 408 | [6, 4, 10, 9, 1, 7, 9, 2, 8, 3, 8] |
| A* weighted 2 -> 8 | 10 | 0.0511 | 44 | [1, 2, 3, 4, 4, 4, 3, 2, 3, 5] |

## Test 2

| | پاسخ مسئله | میانگین زمان اجرا | تعداد استیت های دیده شده | مسیر پاسخ |
|---|---|---|---|---|
| BFS | 20 | 17.593 | 3752 | [14, 9, 4, 15, 12, 3, 12, 6, 14, 1, 14, 12, 3, 8, 6, 4, 10, 11, 8, 3, 13] |
| IDS | - | - | - | - |
| A* | 20 | 17.332 | 3138 | [14, 9, 4, 15, 12, 3, 8, 6, 14, 1, 14, 12, 6, 8, 6, 4, 10, 11, 5, 3, 13] |
| A* weighted 1 -> 1.5 | 20 | 13.873 | 2780 | [14, 9, 4, 15, 12, 3, 8, 6, 14, 1, 14, 12, 6, 8, 6, 4, 10, 11, 8, 3, 13] |
| A* weighted 2 -> 8 | 21 | 0.726 | 323 | [14, 9, 4, 15, 12, 3, 8, 6, 8, 6, 4, 6, 14, 1, 14, 12, 3, 8, 11, 10, 4, 13] |

**Test 3**

| | پاسخ مسئله | میانگین زمان اجرا | تعداد استیت های دیده شده | مسیر پاسخ |
|---|---|---|---|---|
| BFS | - | - | - | - |

| | | | | |
|---|---|---|---|---|
| IDS | - | - | - | - |
| A* | - | - | - | - |
| A* weighted 1 -> 1.5 | - | - | - | - |
| A* weighted 2 -> 8 | 33 | 2.1654 | 511 | [15, 14, 8, 20, 13, 10, 18, 10, 8, 9, 20, 8, 20, 2, 20, 5, 16, 12, 3, 19, 11, 19, 13, 6, 13, 1, 13, 19, 17, 8, 18, 4, 14, 16] |

**Test 4**

| | پاسخ مسئله | میانگین زمان اجرا | تعداد استیت های دیده شده | مسیر پاسخ |
|---|---|---|---|---|
| BFS | 15 | 0.0315 | 115 | [1, 2, 5, 3, 1, 2, 6, 2, 1, 3, 4, 7, 4, 3, 1, 2] |
| IDS | 13 | 0.4624 | 1267 | [1, 3, 5, 2, 6, 2, 1, 3, 4, 7, 4, 3, 1, 2] |
| A* | 15 | 0.0438 | 108 | [1, 2, 5, 3, 1, 2, 6, 2, 1, 3, 4, 7, 4, 3, 1, 2] |
| A* weighted 1 -> 1.5 | 15 | 0.0322 | 106 | [1, 2, 5, 3, 1, 2, 6, 2, 1, 3, 4, |

| | | | | |
|---|---|---|---|---|
| | | | | 7, 4, 3, 1, 2] |
| A* weighted 2 -> 8 | 15 | 0.00397 | 79 | [1, 2, 5, 3, 1, 2, 6, 2, 1, 3, 4, 7, 4, 3, 1, 2] |

**Test 5**

| | پاسخ مسئله | میانگین زمان اجرا | تعداد استیت های دیده شده | مسیر پاسخ |
|---|---|---|---|---|
| BFS | 12 | 0.1632 | 304 | [1, 2, 6, 8, 7, 4, 3, 5, 2, 6, 7, 6, 2] |
| IDS | 12 | 2.372 | 2855 | [1, 3, 5, 2, 6, 7, 6, 2, 6, 8, 7, 4, 3] |
| A* | 12 | 0.1313 | 214 | [1, 2, 6, 8, 7, 4, 3, 5, 2, 6, 7, 6, 2] |
| A* weighted 1 -> 1.5 | 12 | 0.0944 | 164 | [1, 2, 6, 8, 7, 4, 3, 5, 2, 6, 7, 6, 2] |
| A* weighted 2 -> 8 | 12 | 0.0219 | 53 | [1, 2, 6, 8, 7, 4, 3, 5, 2, 6, 7, 6, 2] |