پروژه سرچ هوش

سپهر آزردار 810199357

:Initial sate

همان استیت شروع است که در ابتدا مسئله به عنوان ورودی داده میشود.

مدل کردن آن به این صورت است که در ابتدا تمام متغیر هایی که در پایین گفته شده است، با مقادیر اولیه(همان صفر یا set خالی) مقدار دهی میشوند. زیرا ما در ابتدا کار، هیچ دستور پخت و مریدی رو ندیده ایم.

:Goal state

استیت ای است که در آن تمام مریدان دیزی خود را دریافت کردن. گل استیت ما، نود خاصی در گراف نیست. در واقع یکی از مریدان در لحظه مناسب(در واقع وقتی اخرین مریدی باشد که دیزی اش را میگیرد) نود goal ما میباشد. و پارامتر های استیت تعیین میکنند که آن نود آیا گل هست یا خیر.

:Action

در هر استیت، ما میتوانیم به همسایه ها خود(نود ها مجاور در گراف داده شده) برویم و آن ها next state ها ما هستند.

و هر استیت ما یه سری پارامتر دارد. که وضعیت سرچ در آن لحظه را نشان میدهد. در واقع نود های ما در گراف مقادیر مختلفی برای پارامترهای میگیرند و همین باعث میشود که ما چندین استیت به ازای یک نود داشته باشیم.

توضیح :

```
class State:
  def __init__(self,pos,path,cur_time,wait,hard_
    self.position=pos
    self.path=path
    self.current_time=cur_time
    self.wait=wait
    self.hard_nodes=hard_nodes
    self.collected_recipes=collected_recipes
    self.done_morids=done_morids
```

هر استیت مقادیر روبه رو را دارد:

که

Pos: نودی که الان در آن قرار داریم.

Path: مسیری که تا الان طی کردیم(شامل خود pos)

Current_time: مدت زمانی که تا الان صرف کردیم

Wait: اگر در نود ها صعب العبور گیر کنیم مجبور هستیم که صبر کنیم بنابراین با این متغییر، این مدت زمان را نشان میدهم.

Hard_nodes: نود های صعب العبور

Collected_recipes: دستور پخت های سری که تا الان جمع کرده ایم

Done_morid: مریدانی که دیزی شان را گرفته اند.

فرایند کلی:

الگوریتم ها بهم دیگه خیلی نزدیک هستند، و تفاوت ها جزئی در اولویت و نحوه ی پیاده سازی دارند.

الگوریتم کلی در bfs:

ما درخت فضای نمونه ما را به صورت سطر به سطر طی میکنیم .الگوریتم به نحوی است که نود هایی که در یک سطر(ارتفاع) از درخت هستند، هزینه ی برابر دارند. اگر همه یال ها در این درخت هزینه برابر داشته باشند این اتفاق میفتد. و بدین شکل الگوریتم بهینه هم هست. برای پیاده سازی از صف(FIFO) استفاده میکنیم. برای wait هم به اندازه wait صبر میکنیم در واقع به این اندازه فرایند expand کردن برای این نود رو به تاخیر میندازیم. تا موقعی که بچه هایش وارد صف میشود هزینه اش با بقیه next state

هایی که وارد شده اند و خواهند شد یکی باشد. در واقع در bfs هر سطر از درخت سطر پایینی اش رو میسازد و همان طور گفته شد با تمام نود هایی که در یک سطر وجود دارند هزینه برابری داشته باشند.

:IDS

باید dfs بزنیم. برای اینکار از یک استک استفاده میکنیم و یک استیت را از بالای آن پاپ میکنیم در صورتی که جواب باشد که برمیگردیم. و اگر باید صبر کنیم به، همان اندازه صبر میکنیم و زمان رو به همان اندازه افزایش میدهیم و در صورتی که wait = 0 بود بچه ها استیت را(nest state) در استک پوش میکنیم و تا عمق گفته شده فرایند رو ادامه میدهیم. عمق رو با cur_time اندازه میگیریم از انجایی هر عمق معادل یک واحد زمانی میباشد. بدین شکل این الگوریتم هم همانند بالایی بهینه است

:Astar

فرایند مشابه بالا است فقط از یک min-heap برای پیاده سازی اولویت استفاده میکنیم. این الگوریتم هم بهینه است چون از یه heuristic که کانسیستنت میباشد استفاده میکنیم. هر نود رو بررسی میکنیم، next_state ها رو میسازیم و پارامتر هاشون رو مقدار دهی میکنیم و به هر استیت یه f(n) نیز نسبت میدهیم و با توجه به آن(و با کمک min-heap) در هرمرحله استیتی که کمترین H را دارد رو انتخاب میکنیم.

Weighted: این الگوریتم optimslity رو دیگر تضمین نمیکند. ولی سریع تر است چون با ضرب یه عدد بزرگ تر از 1 در h(n) (به ازای نود ها مختلف) مسیر ها به سمت گل اولویت بیشتر پیدا میکنند(اختلاف بین مسیر ها در جهت گل و مخالف آن زیاد میشود).

:Heuristic

برای هر استیت بدین شکل تعریف میشود:

حاصل جمع تعداد مرید ها باقی مانده + دستور پخت های خوانده نشده – تعداد نود ها مشترک
در واقع نود هایی یک از یان دو خاصیت رو دارا بودند رو در یک set میریزیم تا یم نود رو دوبار نشماریم.

و consistent هست چون هر فرزندی نسبت به پدر خود هزینه بیشتر مساوی رو تخمین میزند. در بهترین حالت به ازای یک حرکت هم یه مرید دیزی میگیره هم یه دستور پخت خونده میشه که در این صورت یکی به cur_time اضافه و از h کم میشود پس child، بزرگ تر مساوی پدر خود هزینه دارد. پس consistent است.

به ازای تست کیس 3

میانگین	تعداد استیت	پاسخ مسئله	
0.623	2066	13, 11, 10, 3, 2, 6, 12, 5, 9, 4, 1, 13, 11, 10	BFS
3.8	3069	13, 11, 10, 3, 2, 6, 12, 5, 9, 4, 1, 13, 11, 10	IDS
0.075	376	13, 11, 10, 3, 2, 6, 12, 5, 9, 4, 1, 13, 11, 0.10	A*
0.004	30	13, 1, 4, 9, 5, 12, 6, 2, 3, 10, 11, 10, 5, 9, 4	W1و2
0.02	134	13, 11, 10, 3, 2, 6, 12, 5, 9, 4, 1, 13, 11, 10	W2و1.5

A* و weighted نود های کمتری بسط میدهند چون آگاهانه هستند