هوش مصنوعی **تمرین شماره ۱**

مدرسین: **دکتر فدایی و دکتر یعقوب‌زاده**

طراحان: پاشا براهیمی، کسری حاجی‌حیدری، محمدصادق ابوفاضلی

مهلت تحویل: چهارشنبه ۱۶ اسفند ۱۴۰۲، ساعت ۲۳:۵۹

بخش کتبی

Agents

جدول زیر را پر کنید.

Environment	CS-GO	Sudoku
Fully/Partially Observable		
Deterministic/Stochastic		
Episodic/Sequential		
Static/Dynamic/Semi-Dynamic		
Discrete/Continuous		
Single-agent/Multi-agent		

بازی CS-GO (Counter-Strike) یک بازی شوتر اول شخص است که در آن تعدادی عامل (بازیکن) در یک map قرار گرفته و به بازی می‌پردازند.

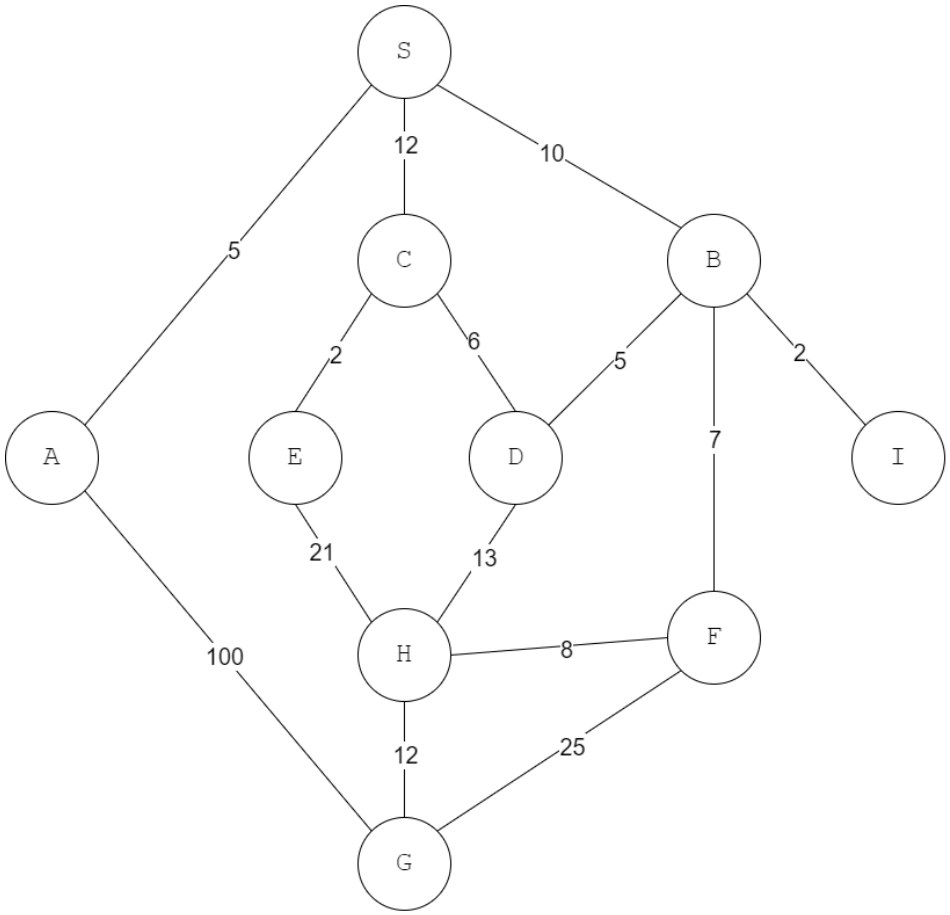
پاسخ:

Environment	CSGO	Sudoku
Fully observable/Partially	Partially	Fully
Deterministic/Stochastic	Stochastic	Deterministic
Episodic/Sequential	Sequential	Episodic
Static/Dynamic/Semi Dynamic	Semi Dynamic	Static
Discrete/Continuous	Continuous	Discrete
Single-agent/Multi-agent	Multi-agent	Single-agent

Search

سوال اول

الف) گراف زیر را در نظر بگیرید. آیا استفاده از الگوریتم های bfs و dfs برای این گراف، جواب بهینه را به ما می‌دهد؟ توضیح دهید در چه صورتی استفاده از این دو الگوریتم پیشنهاد می‌شود.
ب) حداقل هزینه برای رسیدن از راس S به G را با استفاده از الگوریتم Uniform Cost Search محاسبه کنید. به ازای تمام state های دیده شده، راسی که روی آن قرار دارید، مسیر طی شده، هزینه صرف شده و مجموعه‌های Explored و Frontier را به صورت یک جدول بنویسید.
اگر در هر مرحله چند انتخاب داشتید، راسی که از لحاظ ترتیب الفبایی کوچکتر است را انتخاب کنید.



پاسخ:

الف)

در گراف داده شده به علت اینکه هزینه میان نودها یکسان نیست، الگوریتم bfs نمی‌تواند پاسخ بهینه را به ما بدهد. همچنین الگوریتم dfs نیز به صورت کلی پاسخ بهینه را نمی‌دهد. در حالتی استفاده از الگوریتم bfs توصیه می‌شود که هزینه میان نودها یکسان باشد و در این صورت پاسخ بهینه خواهد بود. برای الگوریتم dfs زمانی که تعداد پاسخ‌ها زیاد باشد، استفاده از این الگوریتم می‌تواند مناسب باشد و به سرعت به جواب خواهیم رسید.

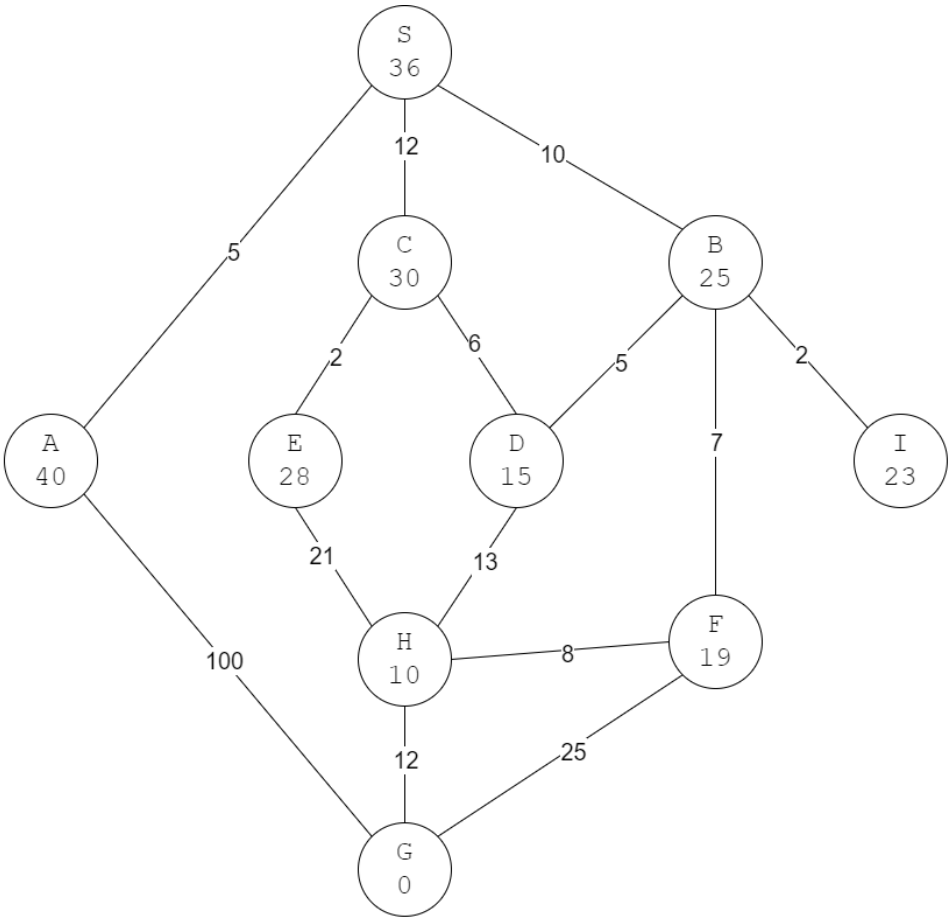
ب)

Frontier	Explored	Cost	Path	Current Node
A,B,C	S	0	S	S
B,C,G	S,A	5	S,A	A

C,D,F,G,I	S,A,B	10	S,B	B
D,E,F,G,I	S,A,B,C	12	S,C	C
D,E,F,G	S,A,B,C,I	12	S,B,I	I
D,F,G,H	S,A,B,C,I,E	14	S,C,E	E
F,G,H	S,A,B,C,I,E,D	15	S,B,D	D
G,H	S,A,B,C,I,E,D,F	17	S,B,F	F
G	S,A,B,C,I,E,D,F,H	25	S,B,F,H	H
-	S,A,B,C,I,E,D,F,H,G	37	S,B,F,H,G	G

سوال دوم

الف) در این مرحله به هر راس یک مقدار هیوریستیک نسبت داده شده است. الگوریتم A* را روی گراف اجرا کنید و حداقل هزینه برای رسیدن از راس S به G را به دست آورید. به ازای تمام state های دیده شده، راسی که روی آن قرار دارید، مسیر طی شده، هزینه صرف شده، مجموع هیوریستیک و هزینه صرف شده و مجموعه‌های Explored و Frontier را به صورت یک جدول بنویسید.
هر state را تنها یک بار بررسی کنید. یعنی در صورت وجود راسی در مجموعه Explored، دوباره آن راس را بررسی نکنید (حتی اگر مسیر کوتاه‌تری برای آن یافت شده باشد). اگر در هر مرحله چند انتخاب داشتید، راسی که از لحاظ ترتیب الفبایی کوچکتر است را انتخاب کنید.
ب) هیوریستیک را از لحاظ Admissible و Consistent بودن بررسی کنید.



پاسخ:

الف)

Frontier	Explored	Cost + h	Cost	Path	Current Node
A,B,C	S	36	0	S	S
A,C,D,F,I	S,B	35	10	S,B	B
A,C,F,I,H	S,B,D	30	15	S,B,D	D
A,C,F,H	S,B,D,I	35	12	S,B,I	I
A,C,H,G	S,B,D,I,F	36	17	S,B,F	F
A,C,G,E	S,B,D,I,F,H	35	25	S,B,F,H	H
A,C,E	S,B,D,I,F,H,G	37	37	S,B,F,H,G	G

ب)

به ازای هر گره، هیوریستیک در نظر گرفته شده از هزینه واقعی مسیر آن گره تا گره مقصد کوچکتر است، در نتیجه هیوریستیک Admissible است.

تفاوت هیوریستیک گره‌های S و B برابر 11 است که از هزینه واقعی انتقال از S به B بیشتر است. در نتیجه در تعریف شرط

$$h(N) \leq c(N, P) + h(P)$$

نمی‌گنجد، بنابراین این هیوریستیک Consistent نیست.

سوال سوم

الف) در چه زمانی رویکرد local search نسبت به سایر روش های جستجو برتری دارد؟

ب) یکی از الگوریتم های local search، الگوریتم hill climbing می‌باشد. یکی از مشکلات اصلی این الگوریتم احتمال پیدا کردن local maximum به عنوان جواب نهایی می‌باشد. دو راه حل برای این مشکل ارائه دهید.

پاسخ:

الف)

زمانی که فضای جستجوی ما بسیار بزرگ باشد، استفاده از الگوریتم‌های local search پیشنهاد می‌شود. به این دلیل که تمرکز این الگوریتم‌ها بر همسایگی‌های یک نقطه از فضا و نه کل فضا است و به همین علت می‌توانند با محاسبات و حافظه کمتر به پاسخ برسند. هر چند باید توجه داشت که این الگوریتم‌ها لزوما جواب optimal را نمی‌دهند و به عبارتی یک trade-off بین الگوریتم‌های local search و الگوریتم‌های جستجوی دیگر مانند bfs و dfs وجود دارد. اگر منابع محاسباتی محدود باشند (یا فضای جستجو بسیار بزرگ باشد) و تاکیدی بر رسیدن به global maximum نداشته باشیم، الگوریتم های local search مناسب هستند.

ب)

- برای اینکه از گیر افتادن در local maximum ها جلوگیری شود می‌توان با یک احتمالی (که در مراحل ابتدایی الگوریتم مقدار بیشتری دارد) اجازه داد که حرکت در جهت منفی هم صورت بگیرد. مثال این راه حل، الگوریتم Simulated Annealing می‌باشد که بر اساس متغیر دما احتمال انجام حرکت در جهت منفی را محاسبه می‌کند.
- اجرا کردن الگوریتم در چند نقطه شروع متفاوت و در نهایت انتخاب بهترین جواب میان آن ها می‌تواند منجر به پیدا کردن global maximum شود.