هوش مصنوعی

دانشکده مهندسی کامپیوتر

دکتر رهبان
بهار ۱۴۰۳

مهدی علی نژاد، ۴۰۱۱۰۶۲۶۶

تمرین ۱، تئوری

سوال ۱

۱. (۲۰ نمره، درجه سختی ۸) درستی یا نادرستی عبارت‌های زیر را با ذکر دلیلی مختصر یا بیان مثال نقض مشخص کنید:

(آ) در یک فضای متناهی، درخت جستجو می‌تواند نامتناهی باشد.

(ب) اگر f و g دو تابع قابل‌قبول [1] باشند، $۰/۷f + ۰/۲g$ نیز حتما قابل‌قبول است.

(ج) در local beam search روش انتخاب بهترین k فرزند، به انتخاب k فرزند تصادفی ارجحیت دارد و صرفا برای انتخاب سریع‌تر ممکن است از روش تصادفی استفاده شود.

(د) فضای حالتی وجود دارد که در آن iterative deepening search بدتر از روش depth-first search عمل کند. (مثلا پیچیدگی زمانی $O(n^۲)$ به‌جای $O(n)$)

(ه) برای اینکه الگوریتم BFS کامل باشد، لازم است که درجه انشعاب متناهی باشد اما برای کامل بودن الگوریتم IDS این شرط لازم نیست.

(و) جست‌وجوی عمق اول، حالتی از Best-First می‌باشد.(یعنی تابع هزینه‌ای وجود دارد که الگوریتم Best-First را معادل DFS کند.)

(ز) فرض کنید یک عامل [2] هوشمند برای بازی Tetris طراحی کرده‌ایم. محیط فعالیت [3] برای این عامل، fully observable ، single agent ، deterministic ، episodic و discrete است. (درستی یا نادرستی هرکدام از ویژگی‌های ذکر شده را بررسی کنید.)

(آ) درست است. زیرا در درخت جست و جو ما مکانیزمی برای جلوگیری از باز شدن دوباره ی نود هایی که دیده شدن نداریم و با تشکیل یک دور این درخت نامتناهی می شود.

(ب) درست است.

$$f \leq h^* \rightarrow ۰/۷f \leq ۰/۷h^*$$

$$g \leq h^* \rightarrow ۰/۲g \leq ۰/۲h^*$$

$$۰/۷f + ۰/۲g \leq ۰/۹h^* \leq h^*$$

(ج) نادرست است. ما می دانیم که گاهی حرکت در جهاتی که در حالت عادی حرکت بدی محسوب می شود، ممکن است مسیری بهینه تر را به ما نشان دهد و یا ما را از دام اکسترمم محلی نجات دهد. برای همین بهتر است بعضا این حرکات را انتخاب کنیم.

(د) درست است. درخت جست و جویی را فرض کنید که یک مسیر است. اگر هدف در عمق d از این درخت باشد، IDS در اردر زمانی $d^۲$ به آن می رسد ولی DFS در اردر زمانی d آن را پیدا می کند.

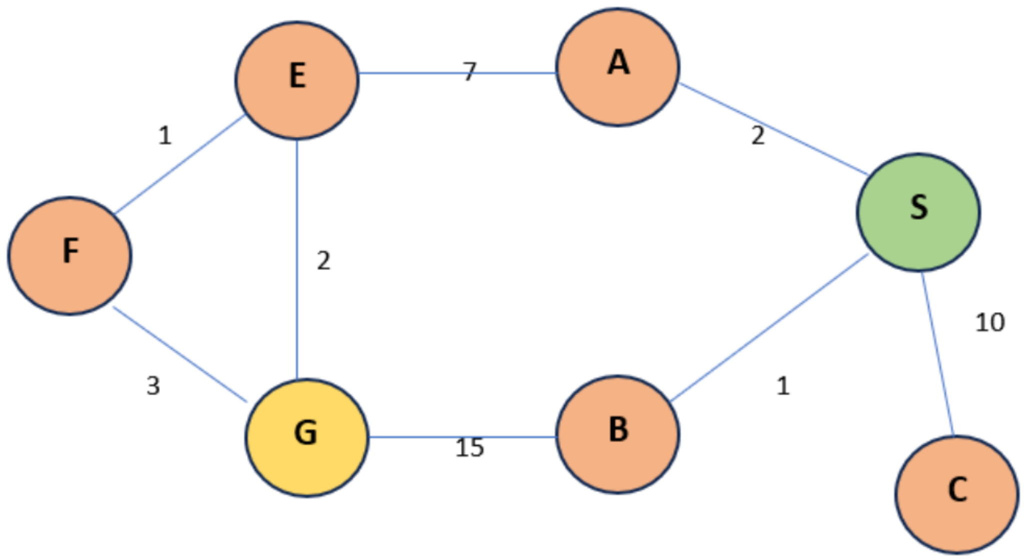
(ه) نادرست است. IDS با اینکه بخشی از آن DFS است ولی همچنان ماهیت های BFS را دارد و تا لایه ای تمام نشود، به عمق نمی افزاید و اگر درجه انشعاب نامتناهی باشد، این الگوریتم نیز به مشکل می خورد.

(و) درست است. با تعریف تابع اکتشاف به گونه ای که به راس های با عمق بیشتر عدد کوچکتری نسبت دهد، آن وقت best first مانند DFS عمل می کند. مثلا $h(n) = ۰$ یا $h(n) = \frac{1}{d_n}$

(ز)

- fully observable است، زیرا تمام محیط بازی دیده می شود.
- single agent هست زیرا با agent دیگری تعامل نخواهد داشت.
- deterministic نیست زیرا قطعات پیشرو به طور شانسی تشکیل می شوند.
- episodic نیست زیرا اکشن های قبلی در اکشن بعد ما تاثیر دارند.
- discrete است زیرا محیط بازی فضای پیوسته نیست.

۲. (۱۰ نمره، درجه سختی ۳) فرض کنید شکل زیر یک فضای جست‌وجو بوده و وضعیت شروع حالت S و وضعیت پایان G باشد. اعداد نوشته شده روی یال‌ها هزینه مسیر هستند.
برای هر کدام از الگوریتم‌های زیر مشخص نمایید چه مسیری طی می‌شود.

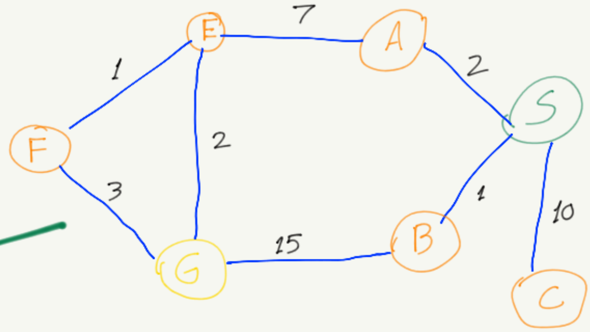


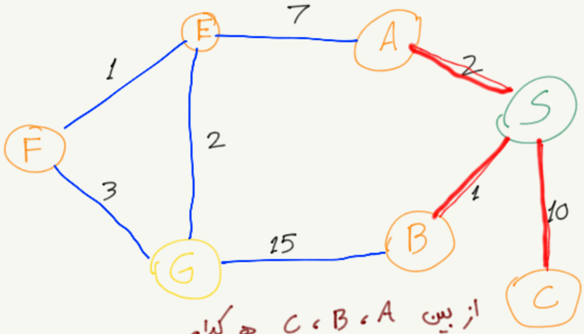
G	F	E	C	B	A	S	States
۰	۱	۱	۷	۱۰	۷	۹	h(s)

- DFS
- greedy
- UCS
- A*
- BFS

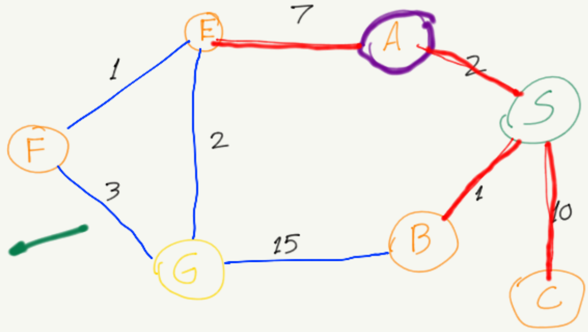
DFS

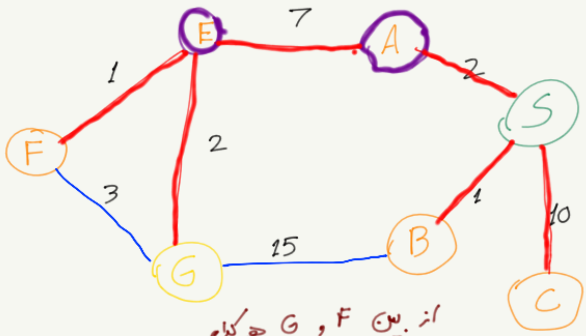
States	S	A	B	C	E	F	G
h(s)	9	7	10	7	1	1	0



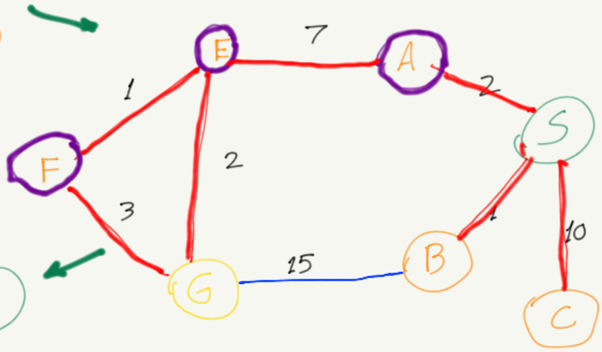


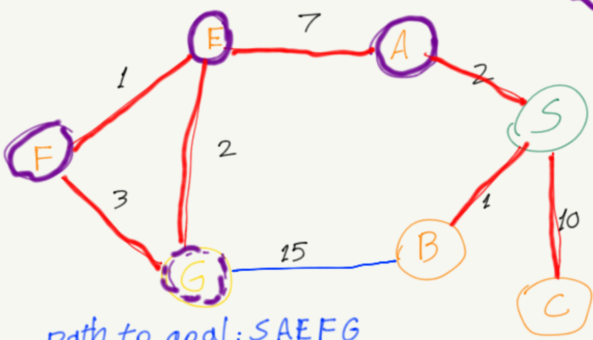
از بین A، B، C هرکدام
می‌توانستند انتخاب شوند





از بین F و G هرکدام
می‌توانستند انتخاب شوند

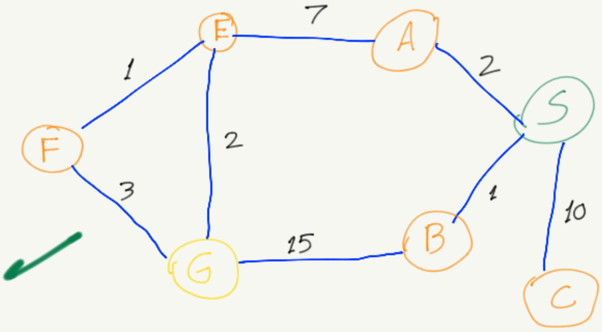


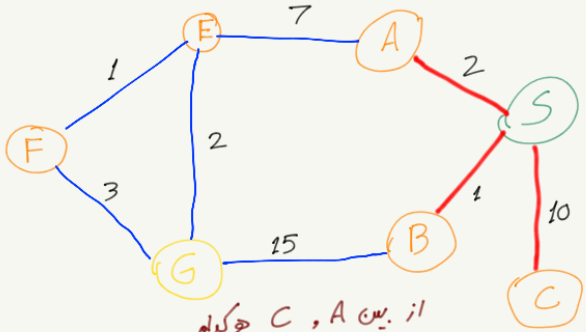


path to goal: SAEFG

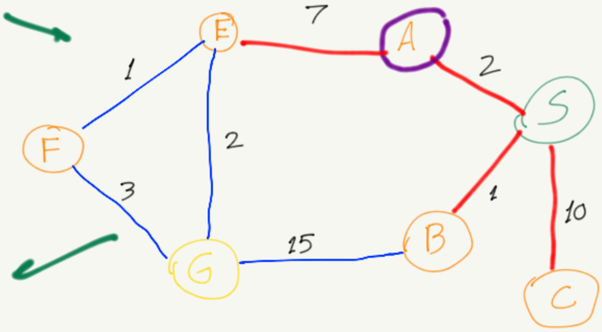
greedy

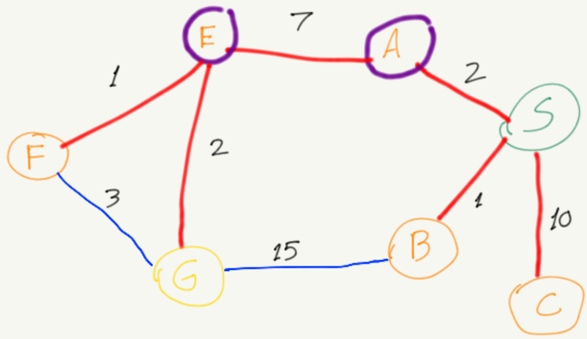
States	S	A	B	C	E	F	G
h(s)	9	7	10	7	1	1	0

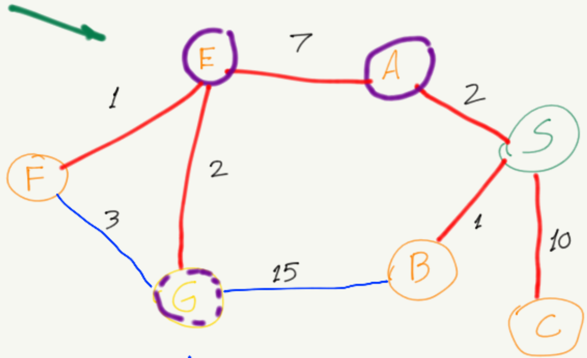




از بین A و C هرکدام
می‌توانستند انتخاب شوند



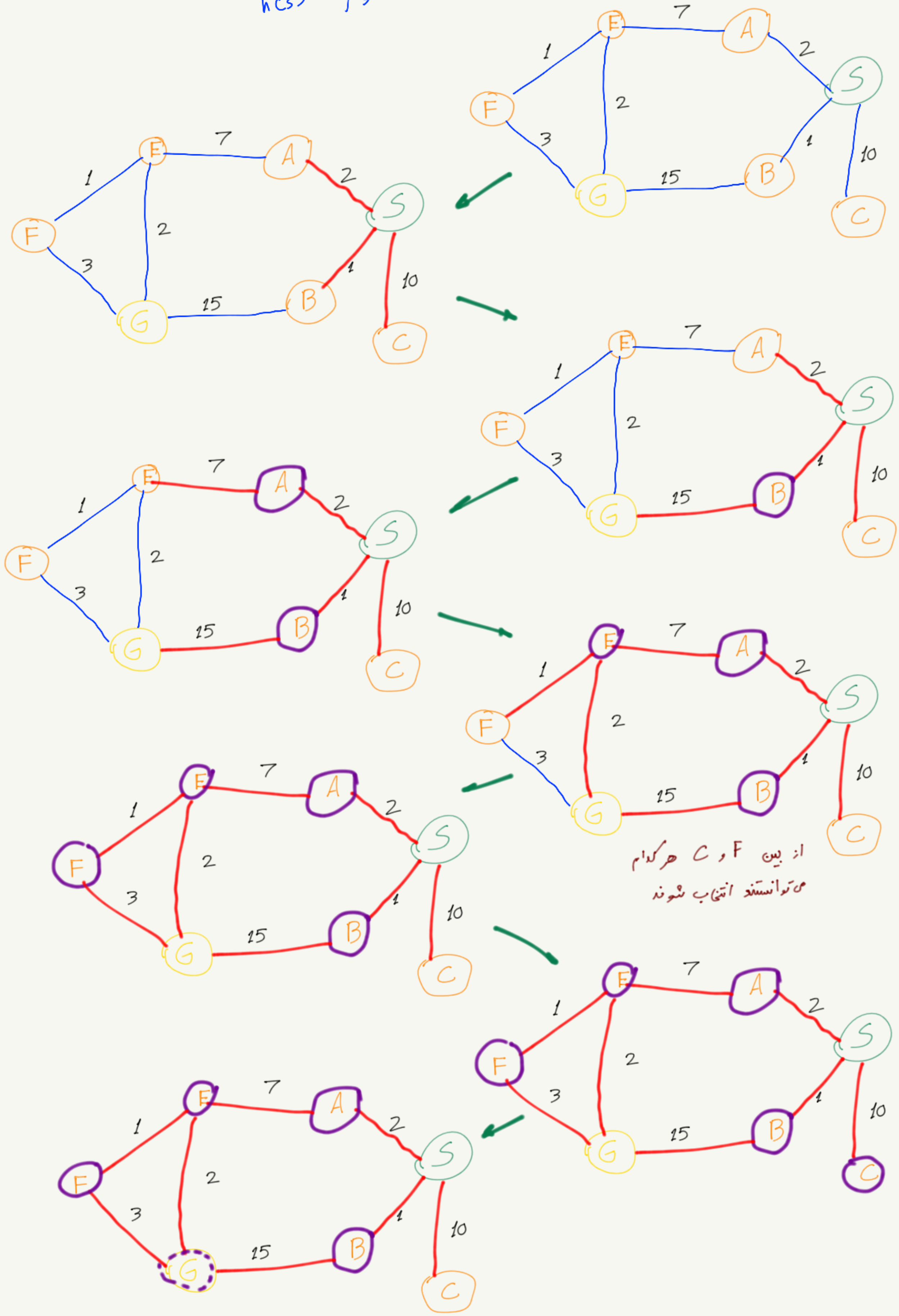




path to goal : SAEG

UCS

States	S	A	B	C	E	F	G
h(s)	9	7	10	7	1	1	0

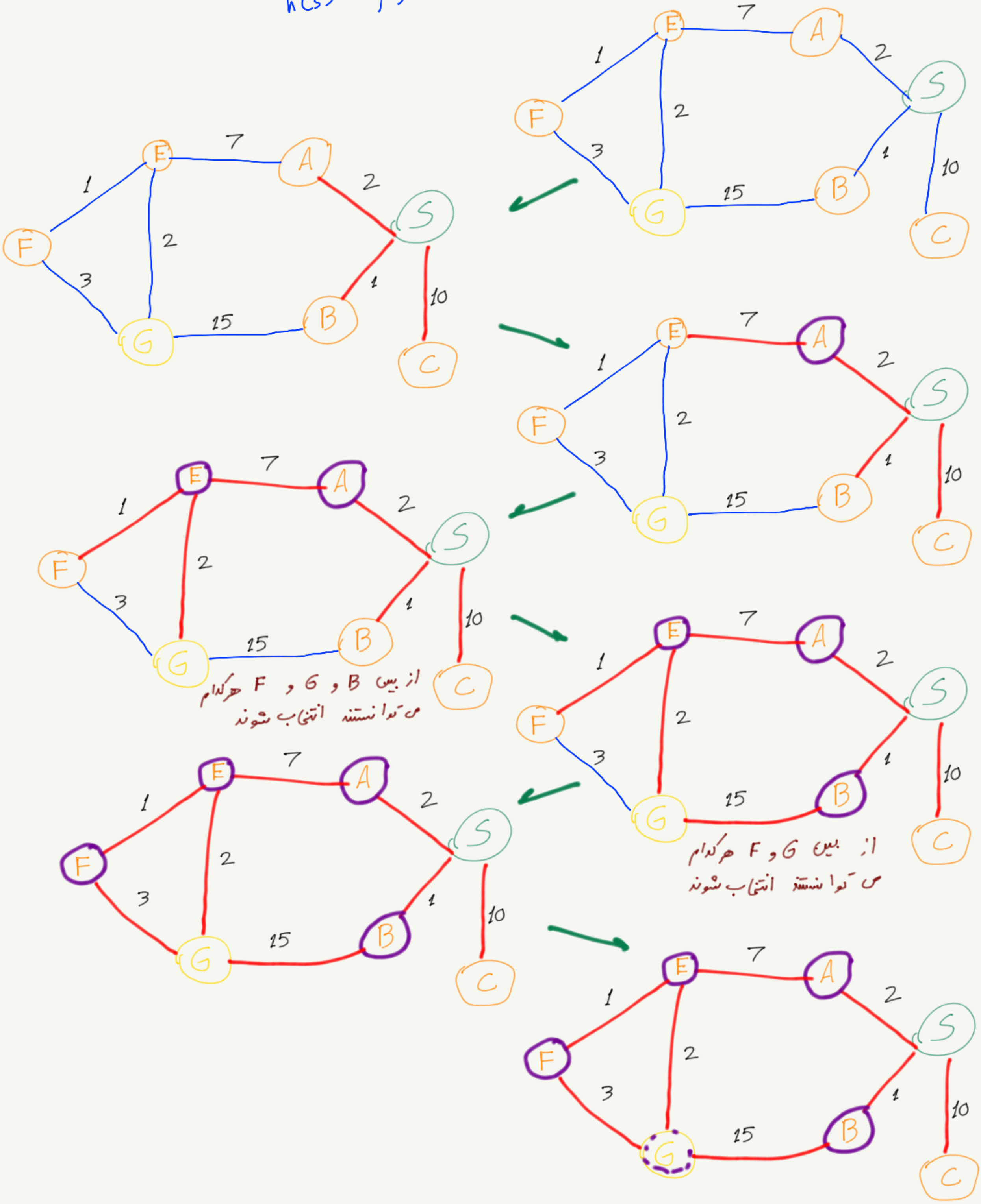


از بین F و C هرکدام
می‌توانستند انتخاب شوند

path to goal: SAEG

A^*

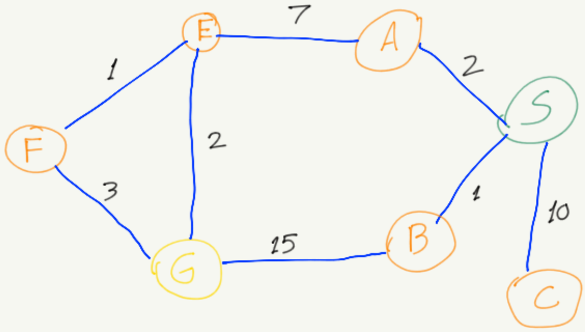
States	S	A	B	C	E	F	G
h(s)	9	7	10	7	1	1	0

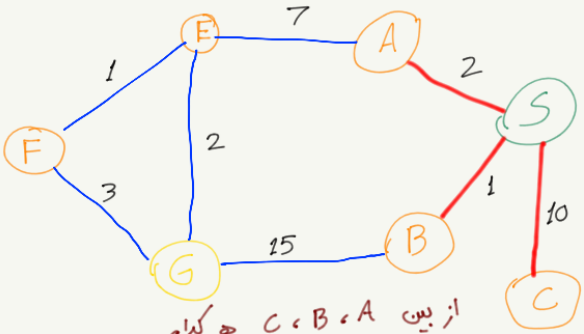


path to goal : S A E G

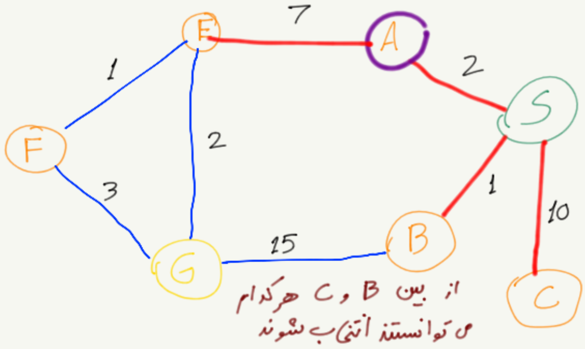
A^* •

States	S	A	B	C	E	F	G
h(s)	9	7	10	7	1	1	0

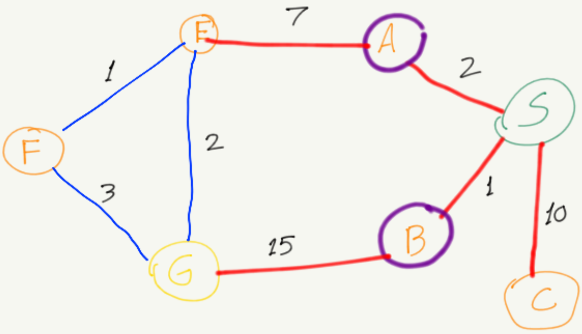


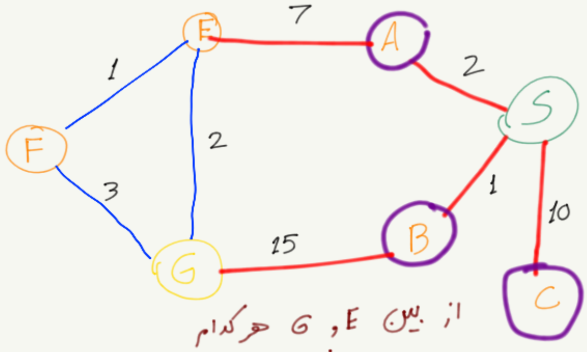


از بین A، B، C هرکدام
می‌توانستند انتخاب شوند

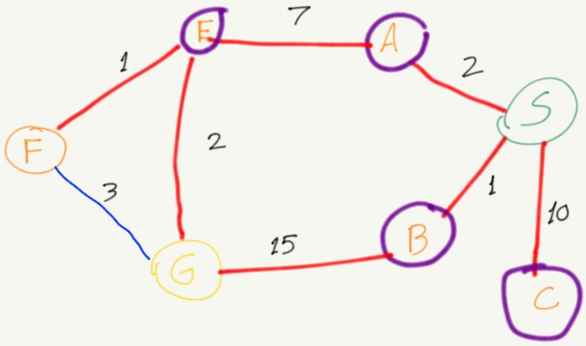


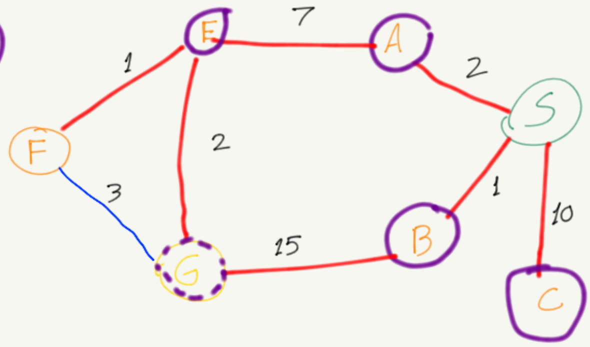
از بین B و C هرکدام
می‌توانستند انتخاب شوند





از بین E و G هرکدام
می‌توانستند انتخاب شوند





path to goal: SBG

۳. (۱۵ نمره، درجه سختی ۴) الگوریتم ژنتیک را در نظر بگیرید که در آن از کروموزوم‌هایی با طول ثابت ۸ و به فرم $x = \overline{ABCDEFGH}$ استفاده می‌کنیم. هر ژن می‌تواند عددی بین ۰ تا ۹ را بگیرد. تابع fitness را برای هر کروموزوم x به صورت زیر تعریف می‌کنیم:

$$f(x) = ۳ * A + ۲ * B + C - D - E + F + ۲ * G + ۳ * H$$

جمعیت اولیه از ۴ کروموزوم زیر تشکیل شده است:

$$x_۱ = ۲۱۸۳۲۹۶۵$$
$$x_۲ = ۱۹۴۷۶۳۲۰$$
$$x_۳ = ۸۹۲۳۴۵۴۸$$
$$x_۴ = ۳۷۹۱۰۹۷۷$$

آ) مقدار fitness نمونه‌های داده‌شده را به دست آورید.

ب) حال عملیات crossover را بر روی دو نمونه با بیشترین fitness انجام دهید، به صورت تک نقطه‌ای که محل crossover نقطه میانی کروموزوم باشد. همچنین عملیات crossover را به صورت دو نقطه‌ای روی دومین و سومین کروموزوم از لحاظ بیشتر بودن fitness انجام دهید. (روش این نوع crossover به این صورت است که $CDEF$ از یک کروموزوم و بقیه ژن‌ها از یک کروموزوم دیگر بدست می‌آیند.)

ج) فرض کنید دستگاهی داریم که می‌تواند یک عدد طبیعی دو رقمی از ۱۰ تا ۸۹ را به صورت تصادفی خروجی بدهد. چهار بار از دستگاه استفاده کردیم و به اعداد زیر رسیدیم:

$$۷۲, ۴۷, ۱۵, ۷۵$$

حال می‌خواهیم با کمک این اعداد عملیات mutation را انجام دهیم. اگر $\overline{xy}$ عدد دو رقمی تولید شده باشد، مقدار x امین حرف کروموزوم از سمت چپ را y واحد در پیمانه ۱۰ اضافه می‌کنیم. عملیات mutation را برای جمعیت بدست آمده در قسمت ب با توجه به اعداد دستگاه، به ترتیب از چپ به راست اعمال کنید.

د) آیا fitness جمعیت جدید نسبت به جمعیت اولیه بهتر شده است؟

ه) طبق جمعیت اولیه، آیا بدون انجام عملیات mutation امکان رسیدن به جواب بهینه (جواب با بیشترین fitness ممکن) وجود دارد؟ اگر بله، مسیر رسیدن به آن را بگویید یا ثابت کنید ممکن نیست.

(آ)

$$f(x_۱) = ۴۷, f(x_۲) = ۱۹, f(x_۳) = ۷۴, f(x_۴) = ۷۵$$

(ب)

$$crossover(x_۴, x_۳) = ۳۷۹۱۴۵۴۸(y_۱), ۸۹۲۳۰۹۷۷(y_۲)$$
$$crossover(x_۳, x_۱) = ۲۱۲۳۴۵۶۵(y_۳), ۸۹۸۳۲۹۴۸(y_۴)$$

(ج)

$$mutation(۷۲, y_۱) = ۳۷۹۱۴۵۶۸$$
$$mutation(۴۷, y_۲) = ۸۹۲۰۰۹۷۷$$
$$mutation(۱۵, y_۳) = ۷۱۲۳۴۵۶۵$$
$$mutation(۷۵, y_۴) = ۸۹۸۳۲۹۹۸$$

(د)

$$f(y_۱) = ۶۸, f(y_۲) = ۸۸, f(y_۳) = ۵۰, f(y_۴) = ۹۶$$

بله fitness متوسط جامعه نسبت به حالت اولیه بیشتر شده است.

(ه) خیر، بیشترین fitness ممکن برای عدد ۹۹۹۰۰۹۹۹ اتفاق می افتد. با crossover ها، جای ارقام اول هر جامعه تغییر می کند و مقدار آنها تغییری ندارد. طبق همین استدلال هرگز رقم ۹ در خانه های A و G و H و هیچگاه رقم ۰ در خانه ی D قرار نمی گیرد.

سوال ۴

۴. (۱۵ نمره، درجه سختی ۴) فرض کنید n مهره در یک صفحه $n * n$ داریم. شما می‌توانید تمام n مهره را همزمان کنترل کنید. چند مهره می‌توانند در یک خانه قرار بگیرند و هر مهره در هر لحظه از زمان می‌تواند به جهت شمال، جنوب، غرب یا شرق حرکت کند و یا این که سر جای خود ثابت باقی بماند. جدول حاوی تعدادی خانه مانع است که هیچ‌کدام از مهره‌ها نمی‌توانند در این خانه‌ها قرار بگیرند. هدف شما این است که تمام مهره‌ها با کمترین مرحله ممکن در یک خانه قرار بگیرند. (در یک مرحله‌ی زمانی تمام مهره‌ها می‌توانند همزمان با هم حرکت کنند.) این مسئله را به عنوان یک مسئله جستجو در نظر بگیرید:

الف) فرض کنید از الگوریتم A^* با کمک تابع اکتشافی[۴] برای حل این مسئله جستجو استفاده کنید. در هر مرحله زمانی $p_i = (x_i, y_i)$ موقعیت مهره i ام در صفحه است. برای هر کدام از توابع اکتشافی زیر، قابل‌قبول[۵] بودن و یک‌نوا[۶] بودن آن را مشخص کنید:

- تعداد جفت مهره‌هایی که در یک مکان یکسان قرار ندارند.
- اگر نزدیک‌ترین مهره‌ها فاصله حداقل ۴ داشته باشند، ۲ و در غیر اینصورت ۰
- $\frac{۱}{۲}max_{i,j}|x_i - x_j| + \frac{۱}{۲}max_{i,j}|y_i - y_j|$
- $\frac{۱}{۲}max(max_{i,j}|x_i - x_j|, max_{i,j}|y_i - y_j|)$

ب) فرض کنید طبق توضیحات بخش قبل، یک تابع قابل قبول و یک‌نوای h_a برای حل مسئله داریم. حال مسئله تغییر می‌کند و شما $m \leq n$ مهره دارید که از سایر مهره‌ها سنگین‌ترند و بعد از هر مرحله‌ای که حرکت کنند، نیاز به ۱ واحد زمانی استراحت دارند (استراحت به این صورت است که سر جای خود می‌مانند). آیا می‌توان گفت که برای مسئله جدید همچنان h_a قابل قبول است؟ یک‌نوایی چطور؟

(آ)
- admissible نیست، سناریویی را فرض کنید که $n - ۱$ مهره در یک خانه باشند و یک مهره در فاصله ی یک خانه ای این خانه. در این صورت تابع اکتشاف عدد $n - ۱$ را برمی گرداند ولی تنها یک حرکت نیاز است تا به goal برسیم. همین مثال نقضی برای monotonic بودن نیز هست. زیرا نامساوی $h(a) \leq ab + h(b)$ را نقض می کند.
- admissible است زیرا اگر نزدیک ترین مهره ها فاصله ی ۴ داشته باشند، حداقل ۴ هزینه برای رسیدن آنها به هم است. monotonic نیست، فرض کنید نزدیک ترین مهره ها ۴ واحد فاصله داشته باشند. اگر در مرحله ی بعدی فاصله ی آنها از ۴ کمتر شود، نامساوی یکتا بودن نقض می شود.
- admissible نیست. مثالی را در نظر بگیرید که ۵ مهره به فرم علامت + در صفحه قرار داشته باشند و در یک مرحله ی دیگر به استیت goal می رسیم ولی در این حالت این تابع به ما عدد ۲ را می دهد. monotonic نیز نیست. همین حرکت مثال نقض آن نیز هست.
- admissible است زیرا در هر مرحله با فرض اینکه هیچ محدودیتی نداشته باشیم و این مهره به سمت هم حرکت کنند، باید آنها را به سمت وسط مستطیلی که توسط ۴ پرت ترین مهره تشکیل می شود ببریم و در حرکت طول و عرض این مستطیل را ۲ واحد می توانیم کوچک کنیم که این به این معنی است که حداقل $\left\lceil \frac{\max(\max_{ij}(|x_i-x_j|),\max_{ij}(|y_i-y_j|))}{۲} \right\rceil$ حرکت لازم است. monotonic نیز است، همانطور که در بالا گفته شد، در هر مرحله حداکثر یک واحد این تابع کاهش پیدا می کند و هر حرکت نیز هزینه ی یک دارد پس نامساوی $h(a) \leq ab + h(b)$ همواره برقرار است.

(ب) admissible بودن آن همچنان پا برجا می ماند، زیرا ما داریم قید به مسئله اضافه می کنیم و باعث پیچیدگی بیشتر آن می شویم. به صورت اثبات نیز می توان گفت که در حالت $m = ۰$ که قابل قبولیم. با اضافه شدن این مهره های سنگین، در بهترین حالت باعث تغییر هزینه نمی شوند و در حالت های دیگر ممکن است هزینه را افزایش دهند پس تابع ما همچنان admissible است. در مورد monotonic بودن نیز همینطور است، در بهترین حالت که گفتیم تغییری ایجاد نمی کنند و در بعضی حالات وضعیت را بدتر نیز می کنند و باعث افزایش هزینه ی بین دو استیت همسایه می شوند که این به یکنوا بودن کمک می کند.

سوال ۵

۵. (۱۵ نمره، درجه سختی ۵) یک جدول $n*n$ داریم که هر خانه‌ی آن سیاه یا سفید است. در هر مرحله می‌توانیم یک زیرجدول از آن را انتخاب کنیم و رنگ خانه‌های درون آن را معکوس کنیم (از سیاه به سفید و بالعکس). می‌خواهیم با کمترین تعداد مرحله کاری کنیم که جدول کاملا سفید شود.

الف) فضای مساله را گونه‌ای پیکربندی کنید به نحوی که حالت‌ها، کنش‌ها، ضریب انشعاب و حالت اولیه و نهایی مساله واضح باشد و آن‌ها را مشخص کنید.

ب) اندازه فضای مساله را به فرمت O بزرگ و بر حسب n بدست آورید.

ج) یک تابع اکتشافی نابدیهی برای حل این مساله ارائه دهید و این تابع را از لحاظ یک‌نوایی و قابل‌قبول بودن بررسی کنید.

(آ)
- حالت ها را هرکدام با یک ماتریس n * n متشکل از ۱ و ۱− مدل می کنیم
- کنش های ما را می توان به یک ماتریس m * k از ۱− تشبیه کرد که در یک قسمتی از ماتریس حالت ضرب نظیر به نظیر می شود.
- ضریب انشعاب برابر است با:

$$\sum_{i=۰}^{n-۱}\sum_{j=۰}^{n-۱}\sum_{k=۰}^{n-i}\sum_{m=۰}^{n-j} ۱ = \sum_{i=۰}^{n-۱}\sum_{j=۰}^{n-۱}(n-i)(n-j) = \left(\sum_{i=۰}^{n-۱}(n-i)\right)\left(\sum_{j=۰}^{n-۱}(n-j)\right) = \left(\frac{n(n+۱)}{۲}\right)^۲$$

- حالت اولیه ماتریسی پر شده از ۱ و ۱− است
- حالت نهایی حالتی است که تمام ماتریس با ۱ پر شده باشد.

(ب) هر خانه ی این ماتریس ۲ حالت می تواند داشته باشد و $n^۲$ تا خانه داریم پس مسئله $۲^{n^۲}$ حالت دارد و از اردر $O(۲^{n^۲})$ است.

(ج) تابع heuristic را اینگونه تعریف می کنیم که خانه های ۱− ناپیوسته را در نظر بگیرد و گوشه های آنها را بشمارد، و عدد $\left\lceil \frac{\text{گوشه ها}}{۴} \right\rceil$ را به ما برگرداند.
برای اینکه با حالت goal برسیم باید هرکدام از این گروه خانه هارا قرینه کنیم و اینکار اگر به فرم مستطیل باشند در یک عمل، در غیر این صورت در تعداد بیشتری عمل انجام می شود، پس این عدد کران پایینی برای هزینه ی واقعی تا هدف است و admissible است. monotonic نیز هست زیرا در هر مرحله که یک هزینه پرداخت می کنیم، تابع heuristic حداکثر می تواند یک واحد کمتر شود. یعنی ما تعداد کل گوشه ها را نمی توانیم در یک مرحله بیشتر از ۴ واحد کاهش دهیم پس تابع اکتشافمون همواره یکنواست.

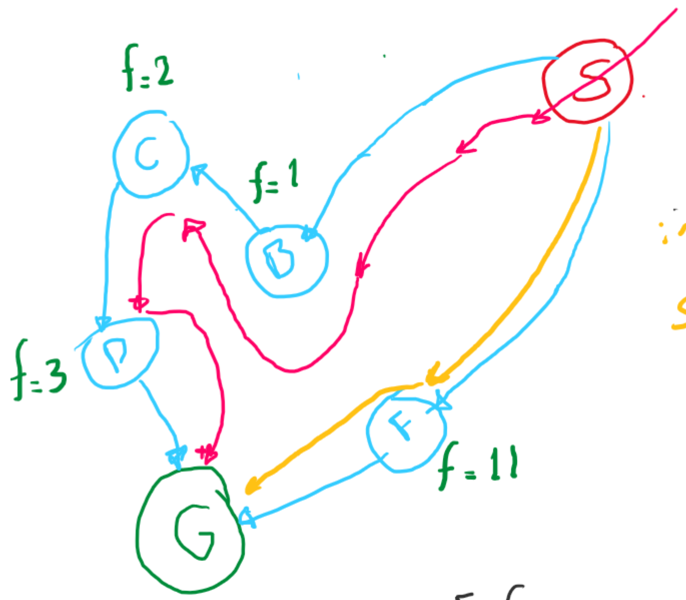
سوال ۶

۶. (۱۵ نمره، درجه سختی ۶) به سوالات زیر درباره الگوریتم A^* پاسخ دهید:

- ۱. یک مثال از یک گراف جهت‌دار و یک تابع heuristic نه لزوما قابل‌قبول بزنید که در آن الگوریتم A^* مسیر بهینه را پیدا نکند. هزینه تمامی یال‌ها باید مثبت باشد و گراف حداکثر ۶ گره داشته باشد. گره‌های شروع و پایان را مشخص کنید و برای گره‌های باقی‌مانده مقدار تابع heuristic را بنویسید.
  - آ) مسیر بهینه را مشخص نمایید.
  - ب) مسیری که A^* پیدا می‌کند را مشخص نمایید.
- ۲. سوال بالا را برای گراف جهت‌دار بدون دور و یک تابع heuristic قابل‌قبول و نه لزوما یکنوا با حداکثر ۸ گره حل نمایید.

(آ) .

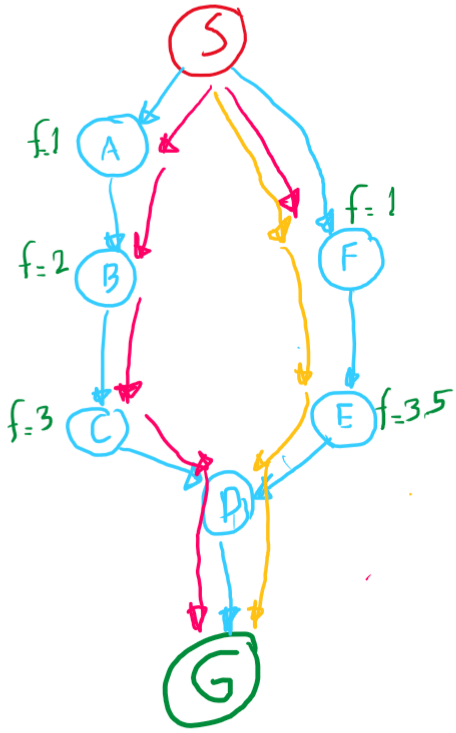
مسیر A^* : SBCDG



مسیر بهینه: SFG

n	B	C	D	F	G
h(n)	0	0	0	10	0

(ب) .



n	h(n)
A	0
B	0
C	0
D	0
E	1.5
F	0

مسیر A^*: SABCDG

مسیر بهینه: SFEDG