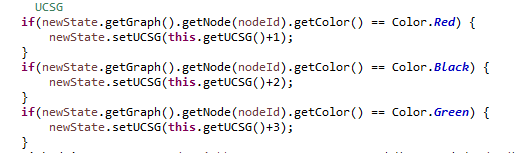
سارا صفایی 9622762204 پروژه search

# DFS

پیاده سازی این نوع از جستجو مشابه جستجوی اول سطح میباشد و تفاوت در این است که برای نگه داری لیست fringe از ساختمان داده ی stack استفاده شده است و چون این جستجو در هر مرحله برای پیدا کردن مسیرش فقط به گره های والدش نیاز دارد و شاخه های قبلی پردازش شده نیاز نمیباشد از حافظه ی کمتری استفاده میکند.

# UCS

برای پیاده سازی جستجوی UCS مطابق با صورت پروژه هزینه ی انتخاب هر گره در نظر گرفته شده و بعد از هر بار ایجاد گره ها هزینه اش محاسبه میشود.

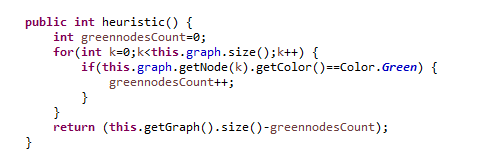


برای نگه داری لیست fringe از صف الویت استفاده شده است و در این صف هر گره با توجه به هزینه ای که دارد مرتب شده است که به ترتیب هزینه از صف خارج و پردازش میشوند.

# GBFS

برای این الگوریتم نیاز است که تابع ابتکاری داشته باشیم که مقدار h(n) را برای هرگره محاسبه کنه و به طور حریصانه ابتدا گره های با h(n) کمتر را بسط دهیم که مشابه UCS از صف الویت استفاده شده است.

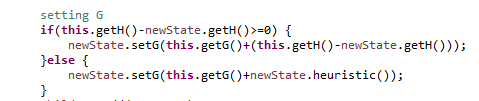
تابع ابتکاری زیر :



در این تابع برای هر حالت تعداد گره هایی که سبز نیستند در گراف مسئله را محاسبه میکنیم پس هرچه این مقدار کمتر باشد الویت انتخاب گره بیشتر است این تابع سازگار و قابل قبول است.

قابل قبول است زیر مقدار h هر گره از مقدار هزینه ی واقعی n تا گره ی هدف کمتر یا مساوی است . h(n) <= h\*(n)چون هزینه ی مسیر جوری محاسبه میشود که بین هر دو گره این هزینه به قدری باشد که برای رنگ کردن گراف هزینه شده است و نه کمتر یعنی اگر نود های سبز کمتر یا بیشتر شوند هزینه به همان درهردو صورت میزان افزایش میابد.

و این تابع سازگار است زیرا h هر گره مقداری کمتر یا برابر مقدار h گره بعد از خودش بعلاوه ی هزینه ی واقعی مسیر از خودش تا گره بعد از خودش دارد.



# Astar

در این الگوریتم مشابه الگوریتم های قبلی که از صف الویت برای مقایسه حالات استفاده شد استفاده کرده ایم اما مقایسه هر گره با توجه به مقدار F(n) هر گره صورت میگیرد که برابر با جمع هزینه ی واقعی یعنی G(n)و هزینه ی تابع ابتکاری یعنی H(n) می باشد.

