گزارش پروژه BFS و DFS

محمد مهدی حجازی

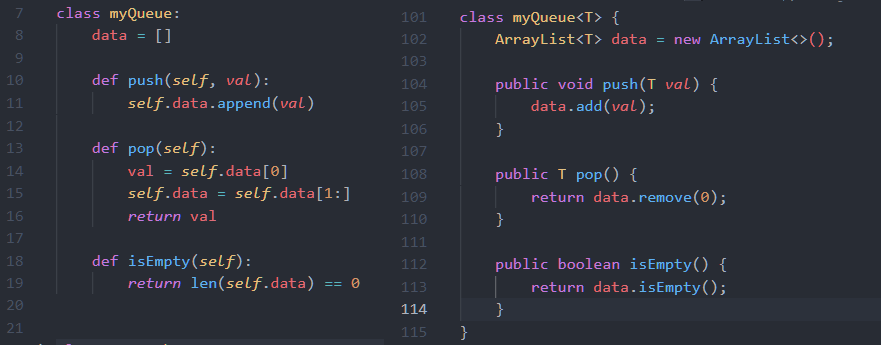
مبانی و آشنایی با هوش مصنوعی

در این پروژه ما یک گراف 10 راسی در اختیار داریم و با دریافت یک فایل test.txt، یال های این گراف را از فایل خوانده و رئوس مجاور را به هم وصل میکنیم.

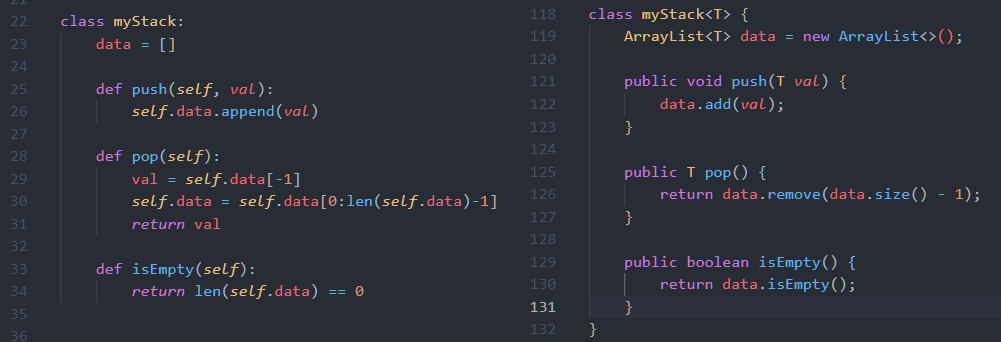
این پروژه به دو زبان جاوا و پایتون پیاده سازی شده است. ولی ساختار هردو به یک صورت هستند.

**کلاس ها**

ما برای پیاده سازی این دو الگوریتم، به ساختار های مختلفی نیاز داریم. یکی از این ساختار ها Queue یا صف که برای پیاده سازی BFS موردنیاز است. صف به این صورت است که ما رئوس را به آن اضافه میکنیم و هر بار اولین راسی که اضافه شده را دریافت میکنیم. (First in First out, FIFO). این ساختار را به صورت کلاس به این صورت پیاده سازی میکنیم:



یکی دیگر از این ساختار ها Stack یا پشته که برای پیاده سازی DFS موردنیاز است. پشته به این صورت است که ما رئوس را به آن اضافه میکنیم و هر بار اخرین راسی که اضافه شده را دریافت میکنیم. (Last in First out, LIFO). این ساختار را به صورت کلاس به این صورت پیاده سازی میکنیم:



**الگوریتم BFS**

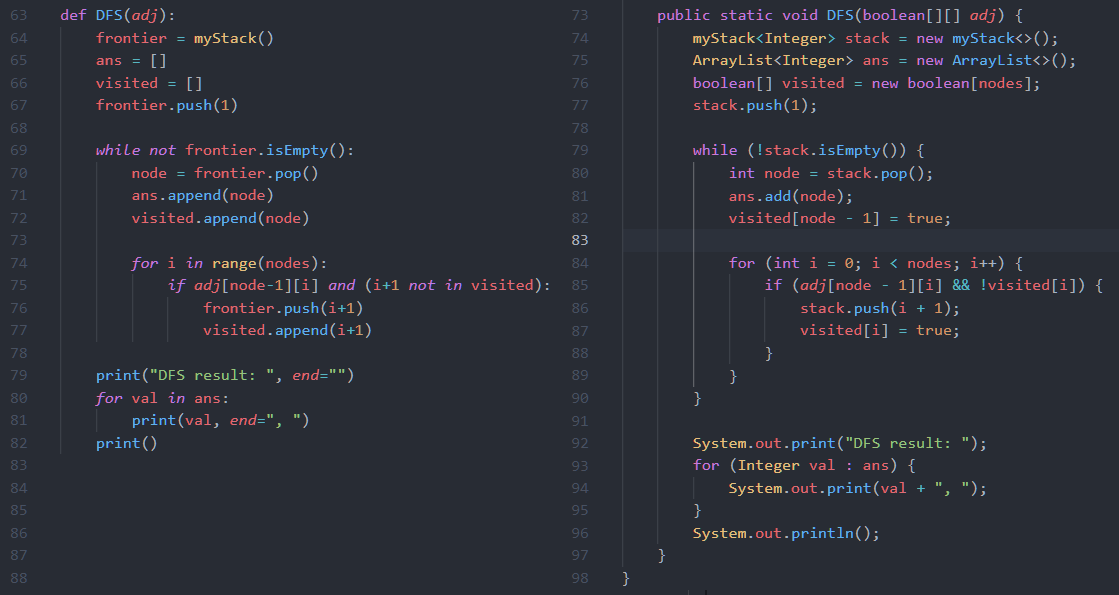
حال با استفاده از ساختار صف که در کلاس myQueue پیاده سازی شده است، الگوریتم BFS را مینویسیم:



در هردو پیاده سازی از راس 1 شروع میکنیم و تا زمانی که صف ما خالی نشده است، همسایه های پیمایش نشده را پیمایش میکنیم و رئوس را به لیست ans اضافه میکنیم. در نهایت مقادیر موجود در داخل ans را به ترتیب چاپ میکنیم که نشانگر پیمایش BFS ماست.

**الگوریتم DFS**

حال با استفاده از ساختار پشته که در کلاس myStack پیاده سازی شده است، الگوریتم DFS را مینویسیم:



در هردو پیاده سازی از راس 1 شروع میکنیم و تا زمانی که پشته ما خالی نشده است، همسایه های پیمایش نشده را پیمایش میکنیم و رئوس را به لیست ans اضافه میکنیم. در نهایت مقادیر موجود در داخل ans را به ترتیب چاپ میکنیم که نشانگر پیمایش DFS ماست.

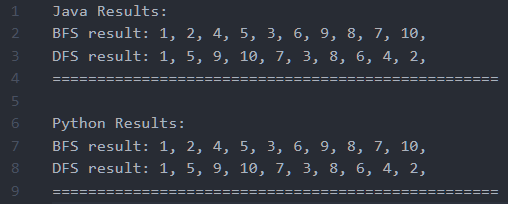
نکته حائز اهمیت در اینجا این است که ما میتوانیم برای چک کردن همسایگی میتوانیم از راس دهم تا راس اول چک میکنیم (خط 84 – خط 74) تا ترتیب پیمایش به ترتیب شماره رئوس باشد.

(یعنی تعویض خط 84 جاوا به : for(int i=nodes-1; i>=0; i--) و تعویض خط 74 پایتون به for i in range(nodes-1, -1, -1): )

**نتایج**

در کنار فایل های پیاده سازی، یک فایل RUN.bat قرار داده شده که با اجرای این فایل، خروجی ها در صفحه و در فایل results.txt نشان داده میشود.

خروجی الگوریتم:



\*نکته: اگر ترتیب چک کردن همسایگی را در DFS برعکس کنیم، پیمایش به صورت زیر درمی آید:

