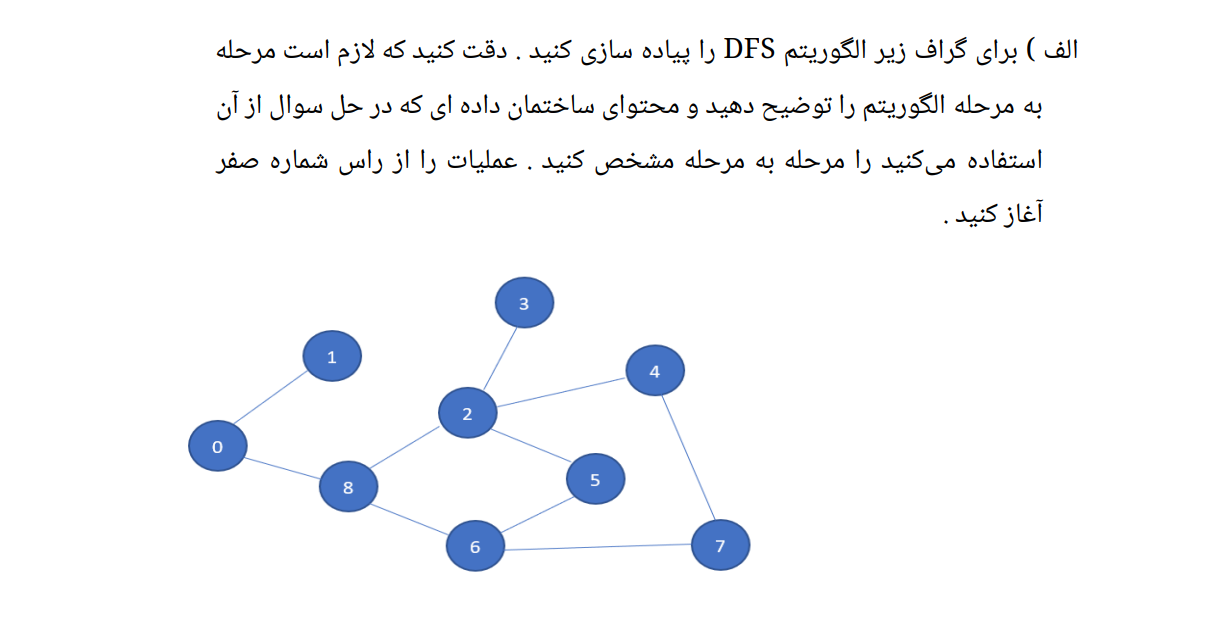
Amirali Farazmand

99522329



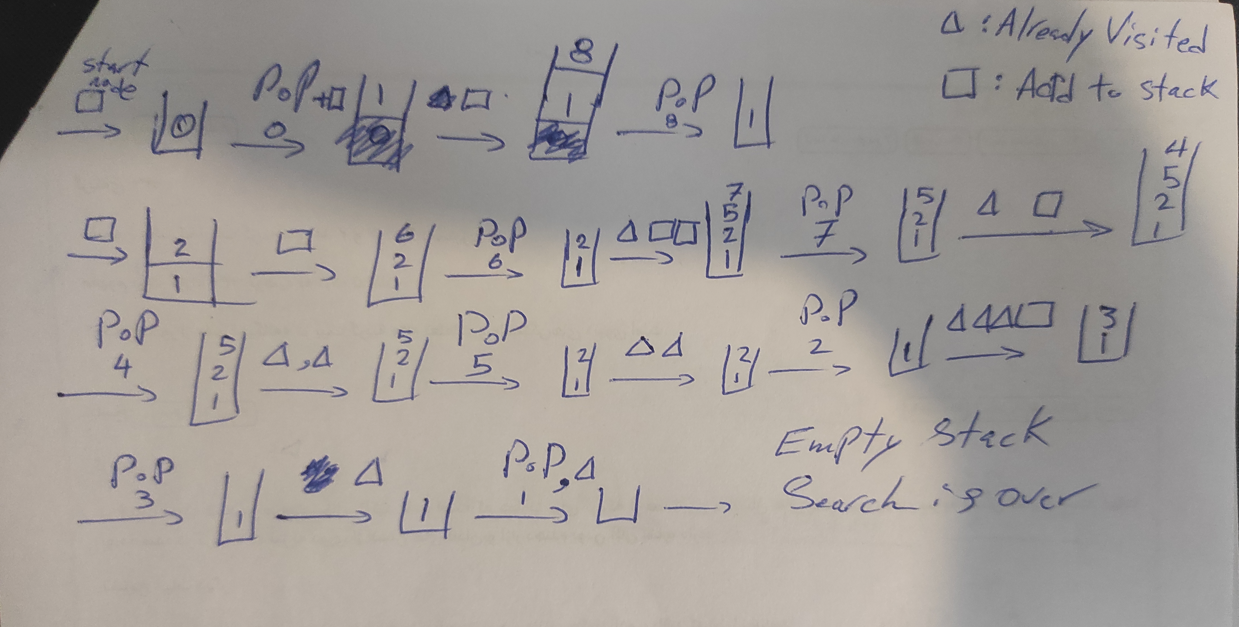
از iterative DFS استفاده شده است که در آن 0 را در استک خود میگذاریم.همچنین آرایه ای بطول نود هایمان داریم که اگر نودی را ویزیت کردیم 1 میگذاریم وگرنه 0.

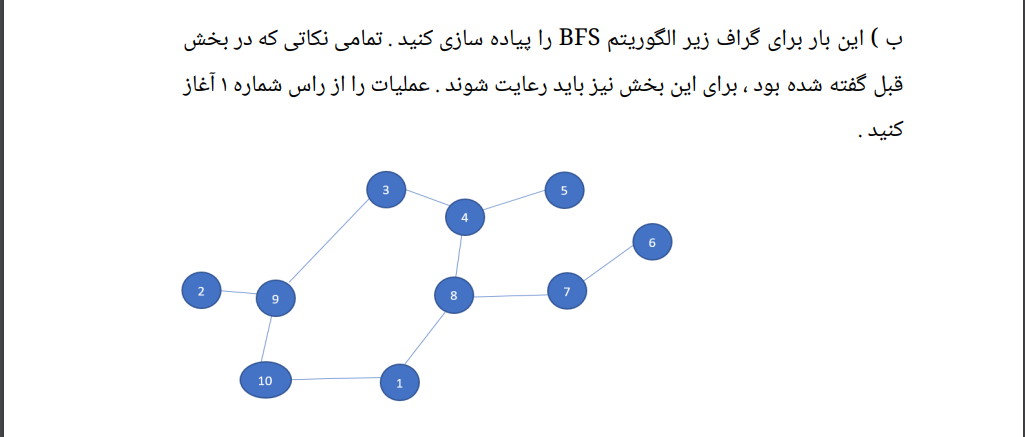
در یک لوپ باید از استک نود ورداریم.ویزیت مربوط به آن را 1 میکنیم و از همه همسایه هایش اگر تا حالا ویزیت نکرده باشیمش در استک میگذاریم وگرنه کاری رویشان انجام نمیدهیم. هر وقت که همه همسایه های نودی را طی کردیم به سراغ نود بعدی داخل استک میرویم. لوپ تا موقعی که استک خالی نشده باشد باید ادامه پیدا کند.

مرحله به مرحله در عکس مشخص کردم.



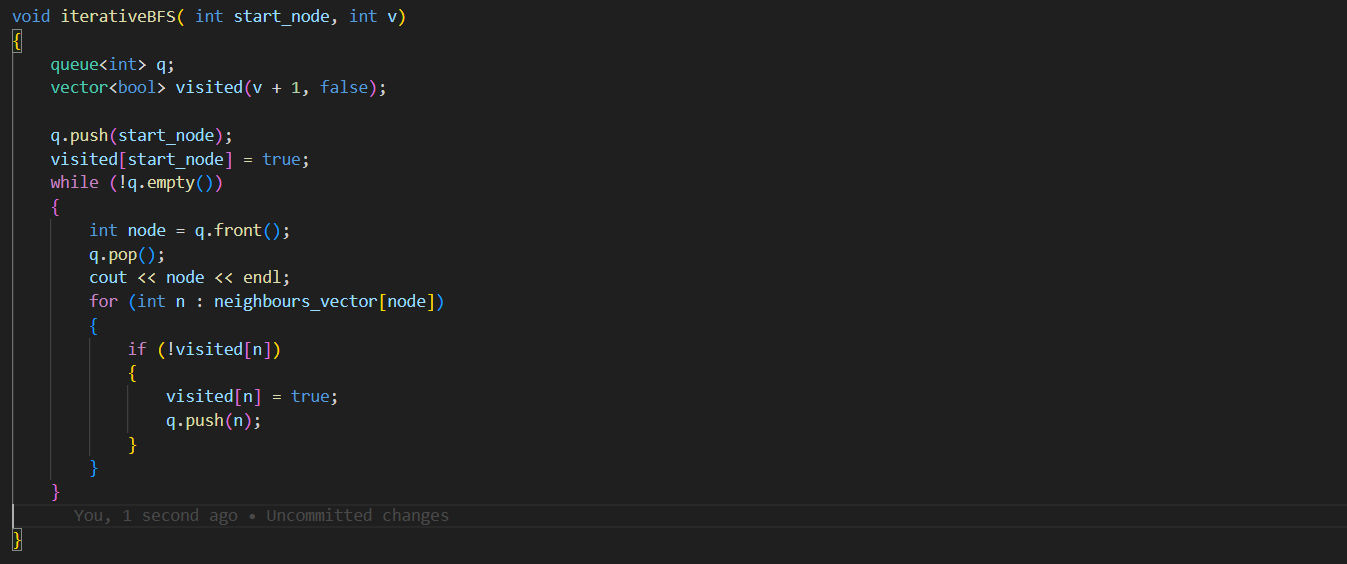


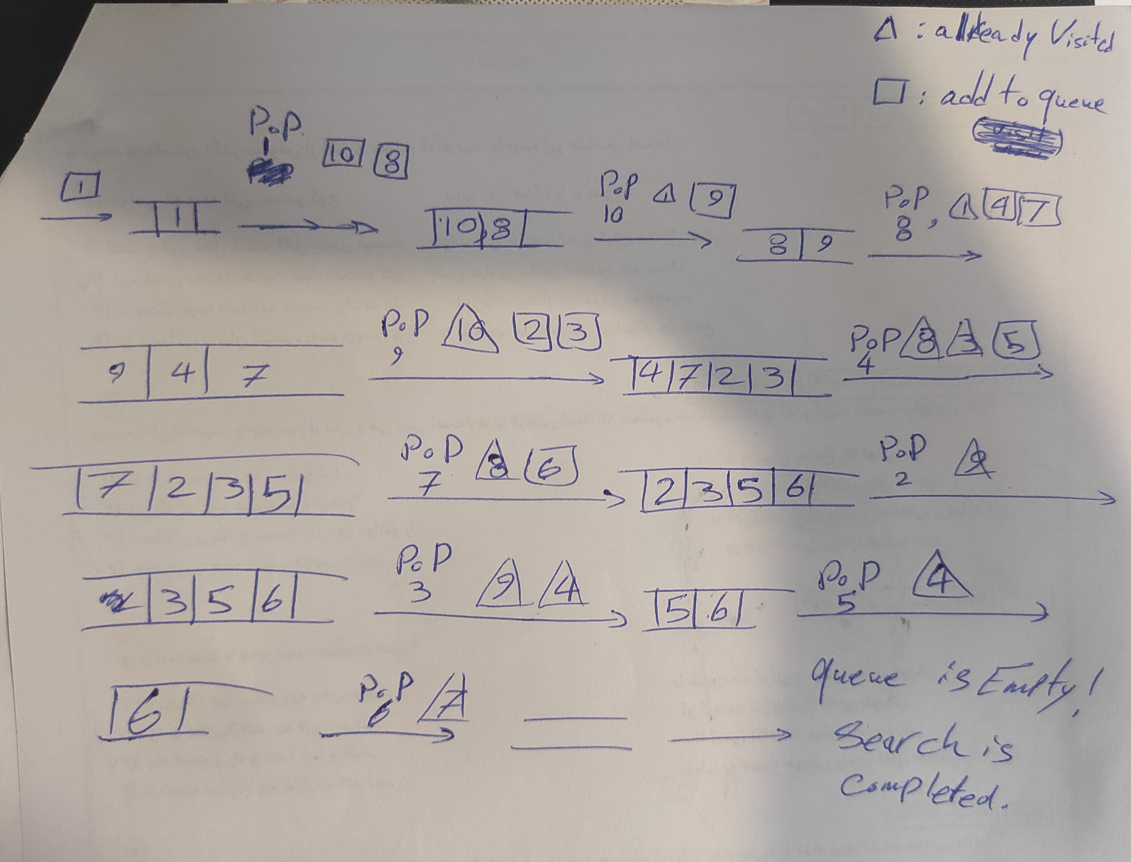


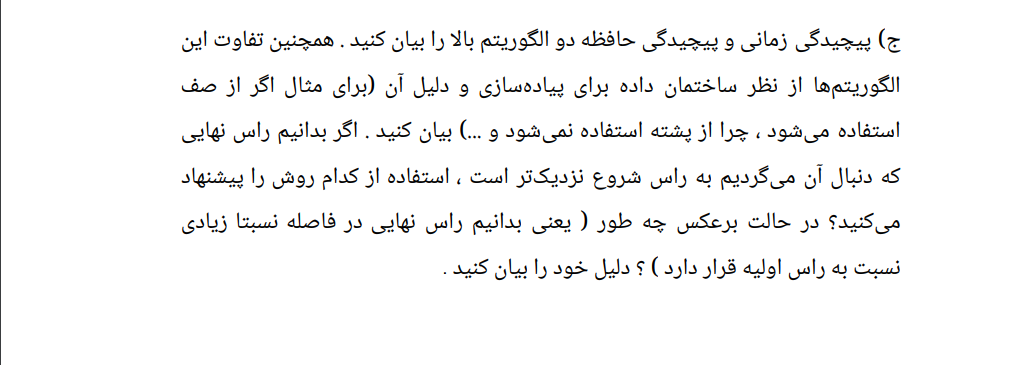


در این بخش از iterative BFS استفاده شده است. از ساختار صف استفاده میکنیم.مثل بخش قبل در آرایه ای بولین ذخیره میکنیم که نود ها را ویزیت کردیم یا نه. نود 1 را داخل صف میگذاریم.در یک لوپ در هر مرحله نود را از صف بیرون میکشیم و برای همه ی همسایه هایش چک میکنیم که اگر قبلا ویزیت شدند کاریشان نمیکنیم ولی اگر نشده باشند خانه مربوطه آن را در آرایه درست میکنیم و به صف اضافه اش میکنیم. این کار را تا وقتی صف خالی شود ادامه میدهیم، وقتی صف خالی باشد یعنی ما همه نود ها را بررسی کردیم.

در شکل مراحل این مثال نشان داده شده اند.







اگر از صف استفاده شود یعنی ما نودی که زودتر دیده ایم را زودتر میرویم سراغش و همسایه های او را بررسی میکنیم. اما در استک وقتی ما نودی را زودتر از همسایه های دیگرش میبینیم دیرتر می اییم سراغش و وقتی کار همه ی آنها تمام شد این نود بررسی میشود. در مثال گفته شده که راس نهایی نزدیک راس شروع است بهتر است bfs و در حالت مخالفش بهتر است dfs زد چرا که bfs نود ها را گویا به ترتیب عمق نسبت به نقطه شروع میپیماید، برخلافش در dfs از نود اولیه تا جایی که عمق دارد هم چه بسا پیش میرود.

مرتبه زمانی هردو برابر با هم است چرا که ما نود ها را ذخیره میکنیم.

O(V) .

همچنین مرتبه زمانی هردو برابر با O(V+E) هست به ازای V برابر تعداد راس و E برابر تعداد یال.