به نام خدا

درس طراحی الگوریتم

تمرین سری سوم

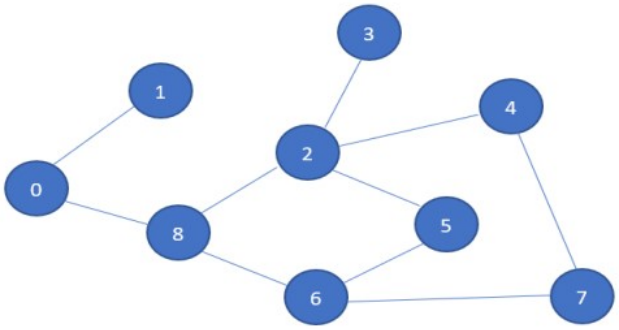
مدرس درس:
سرکار خانم دکتر ملکی

تهیه شده توسط:
الناز رضایی ۹۸۴۱۱۳۸۷

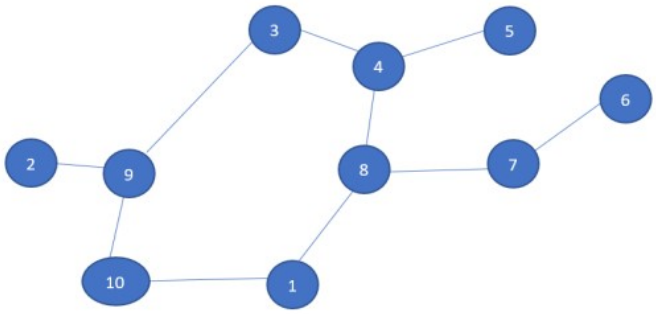
تاریخ ارسال: ۱۴۰۲/۰۲/۰۳

سوال ۱ :

الف) برای گراف زیر الگوریتم DFS را پیاده‌سازی کنید. دقت کنید که لازم است مرحله به مرحله الگوریتم را توضیح دهید و محتوای ساختمان داده‌ای که در حل سوال از آن استفاده می‌کنید را مرحله به مرحله مشخص کنید . عملیات را از راس شماره صفر آغاز کنید.



ب) این بار برای گراف زیر الگوریتم BFS را پیاده‌سازی کنید. تمامی نکاتی که در بخش قبل گفته شده بود، برای این بخش نیز باید رعایت شوند. عملیات را از راس شماره ۱ آغاز کنید.



ج) پیچیدگی زمانی و پیچیدگی حافظه دو الگوریتم بالا را بیان کنید. همچنین تفاوت این الگوریتم‌ها از نظر ساختمان داده برای پیاده‌سازی و دلیل آن (برای مثال اگر از صف استفاده می‌شود ، چرا از پشته استفاده نمی‌شود و ...) بیان کنید. اگر بدانیم راس نهایی که دنبال آن می‌گردیم به راس شروع نزدیکتر است، استفاده از کدام روش را پیشنهاد می‌کنید؟ در حالت برعکس چه طور (یعنی بدانیم راس نهایی در فاصله نسبتا زیادی نسبت به راس اولیه قرار دارد) ؟ دلیل خود را بیان کنید.

پاسخ ۱.الف:

یک لیست برای ذخیره راس‌های ویزیت شده (visited) و یک پشته (stack) برای راس‌های مجاور تعریف می‌کنیم. از راس ۰ شروع می‌کنیم و آن را در لیست visited قرار می‌دهیم. همسایه‌های آن را نیز که شامل رئوس ۱ و ۸ هستند را به ترتیب در stack قرار می‌دهیم. بنابراین داریم:

visited	0								
stack	1	8							

حال باید به سراغ عنصر stack top که در اینجا ۱ می‌باشد برویم و آن را visit کنیم. ۱ همسایه ویزیت نشده ندارد. بنابراین در stack، تنها ۸ باقی می‌ماند.

visited	0	1							
stack	8								

حال به سراغ راس ۸ می‌رویم و آن را ویزیت می‌کنیم. همسایه‌های ویزیت نشده ۸، ۲ و ۶ می‌باشند.

visited	0	1	8						
stack	2	6							

اکنون راس ۲ را انتخاب کرده و به visited آن را اضافه و همسایه‌هایش را در stack قرار می‌دهیم.

visited	0	1	8	2					
stack	3	4	5	6					

راس ۳ را از stack برداشته و به visited اضافه می‌کنیم و چون همسایه‌ای ندارد، به stack چیزی اضافه نمی‌شود.

visited	0	1	8	2	3				
stack	4	5	6						

در ادامه به سراغ عنصر top پشته که ۴ است می‌رویم و همسایه‌های ویزیت نشده‌اش که شامل ۷ است را به پشته اضافه می‌نماییم.

visited	0	1	8	2	3	4			
stack	7	5	6						

به سراغ راس ۷ رفته و تنها همسایه‌اش ۶ می‌باشد.

visited	0	1	8	2	3	4	7		
stack	6	5							

حال به سراغ عنصر top پشته که ۶ است می‌رویم و چون همسایه‌ای ندارد، به stack چیزی اضافه نمی‌شود.

visited	0	1	8	2	3	4	7	6	
stack	5								

حال تنها عنصر پشته که ۶ است را به عناصر visited اضافه می‌کنیم و چون همسایه‌ای ندارد، چیزی به پشته اضافه نمی‌شود. از طرفی چون دیگر عنصری در stack نداریم، یعنی DFS ما به پایان رسیده است و راس visit نشده‌ای نداریم.

visited	0	1	8	2	3	4	7	6	5
stack									

پاسخ ۱.ب:

برای BFS، از صف (queue) استفاده می‌کنیم. ب ه‌این‌صورت که از راس ۱ شروع کرده و آن را در visited قرار داده و همسایه‌هایش که ۸ و ۱۰ هستند را به صف اضافه می‌کنیم.

visited	1									
queue	8	10								

اکنون عنصر ابتدایی صف که ۸ است را برمی‌داریم و همسایه‌های ویزیت نشده‌اش که ۴ و ۷ هستند را به صف اضافه می‌کنیم.

visited	1	8								
queue	10	4	7							

مراحل قبل را تکرار کرده و ۱۰ را از queue برداشته و به visited اضافه و همسایه‌های ویزیت نشده‌اش (۹) را به queue اضافه می‌کنیم.

visited	1	8	10							
queue	4	7	9							

حال به سراغ ۴ رفته و آن را به visited و همسایه‌های ویزیت نشده‌اش (۳ و ۵) را به queue اضافه می‌کنیم.

visited	1	8	10	4						
queue	7	9	3	5						

حال به سراغ ۷ رفته و آن را به visited و همسایه‌های ویزیت نشده‌اش (۶) را به queue اضافه می‌کنیم.

visited	1	8	10	4	7					
queue	9	3	5	6						

اکنون عنصر ابتدایی صف که ۹ است را برمی‌داریم و همسایه ویزیت نشده‌اش که ۲ هست را به صف اضافه می‌کنیم.

visited	1	8	10	4	7	9				
queue	3	5	6	2						

مراحل قبل را تکرار کرده و ۳ را از queue برداشته و به visited اضافه می‌کنیم و چون همسایه ویزیت نشده‌ای ندارد، چیزی به صف اضافه نمی‌شود.

visited	1	8	10	4	7	9	3			
queue	5	6	2							

در ادامه ۵ را از queue برداشته و به visited اضافه می‌کنیم و چون همسایه ویزیت نشده‌ای ندارد، چیزی به صف اضافه نمی‌شود.

visited	1	8	10	4	7	9	3	5		
queue	6	2								

حال ۶ را از queue برداشته و به visited اضافه می‌کنیم و چون همسایه ویزیت نشده‌ای ندارد، چیزی به صف اضافه نمی‌شود.

visited	1	8	10	4	7	9	3	5	6	
queue	2									

در انتها نیز عنصر ۲ را از صف برداشته و به visited می‌افزاییم. از طرفی چون همسایه‌ای ندارد، چیزی به صف اضافه نمی‌شود و صف خالی می‌ماند که نشان می‌دهد BFS نیز به پایان رسیده و تمامی راس‌ها visit شده‌اند.

visited	1	8	10	4	7	9	3	5	6	2
queue										

پاسخ ۱.ج:

پیچیدگی زمانی:

- BFS: $O(V + E)$
- DFS: $O(V + E)$

پیچیدگی حافظه:

- BFS: BFS پیچیدگی حافظه بالاتری نسبت به DFS دارد زیرا باید تمام گره‌ها را در هر سطح از نمودار در یک صف ذخیره کند. پیچیدگی حافظه BFS $O(V)$ است.
- DFS: DFS پیچیدگی حافظه کمتری نسبت به BFS دارد زیرا فقط باید مسیر فعلی را در یک پشته ذخیره کند. پیچیدگی حافظه DFS $O(H)$ است که در آن H ارتفاع درخت است.

ساختار داده برای پیاده‌سازی:

- BFS: BFS از یک ساختار داده صف برای پیاده‌سازی استفاده می‌کند زیرا قبل از رفتن به سطح بعدی، از تمام گره‌ها در هر سطح از نمودار بازدید می‌کند.
- DFS: DFS از یک ساختار داده پشته برای پیاده‌سازی استفاده می‌کند زیرا قبل از عقب‌نشینی، نمودار را در طول یک مسیر تا آنجا که ممکن است بررسی می‌کند.

اگر راس نهایی موردنظر ما به راس شروع نزدیک‌تر باشد، باید از BFS استفاده کنیم زیرا تضمین می‌کند که کوتاه‌ترین مسیر بین دو گره را پیدا کند. با این حال، اگر راس نهایی در فاصله نسبتا زیادی از راس اولیه قرار داشته باشد، DFS ترجیح داده می‌شود زیرا نسبت به BFS حافظه کمتری دارد و ممکن است در برخی موارد زمان اجرای سریع‌تری داشته باشد.