به نام خدا

کوئیز پنجم درس ساختمان داده‌ها و الگوریتم‌ها

مدرس: مهدی صفرنژاد ۲۳ دی ۱۴۰۰

زمان پاسخگویی:۳۰ دقیقه زمان آپلود: ۱۵ دقیقه

سوال ۱:

نئون می‌خواهد میهمانی‌ای برگزار و دوستان ماده‌اش را دعوت کند، ولی به هیچ عنوان دوست ندارد در میهمانی‌اش واکنشی رخ دهد. از آنجا که خانه‌ی او دو بشر بیشتر ندارد، او حداکثر می‌تواند میهمانان را به دو گروه تقسیم کند. به شما لیستی به طول n داده شده که شامل تمام دوتایی‌های دوستان نئون است که با هم واکنش می‌دهند. الگوریتمی با پیچیدگی زمانی $O(n)$ ارائه دهید که به نئون بگوید اصلا می‌تواند میهمانی را با این شرایط برگزار کند یا خیر و اگر می‌تواند، تقسیم‌بندی دوستانش باید به چه شکل باشد.

پاسخ

گراف G را به طوری تشکیل می‌دهیم که راس‌های آن دوستان واکنش‌دهنده‌ی نئون باشند و بین هر جفت ماده‌ای که ممکن است با یکدیگر واکنش دهند یک یال قرار می‌دهیم. حال می‌خواهیم راس‌های گراف را به دو بخش تقسیم کنیم به گونه‌ای که داخل هر بخش هیچ یالی وجود نداشته باشد. این همان تعریف گراف دو بخشی است.
برای تشخیص دوبخشی بودن گراف می‌توانیم روی هر یک از مولفه‌های همبندی گراف، BFS اجرا کنیم و روی هر راس فاصله‌اش تا راس شروع را ذخیره می‌کنیم(فاصله در درخت BFS). در انتها اگر برای هر یال زوجیت عدد روی دو راس متصل به آن متفاوت بود، آن گراف دو بخشی است و تمام راس‌های با عدد زوج را در یک گروه و بقیه‌ی راس‌ها را در گروه دیگر قرار می‌دهیم.
از آنجا که n یال داریم، حداکثر $2n$ راس داریم(دقت کنید راس‌هایی که در لیست نیستند نیازی به محاسبات ندارند). پس پیچیدگی زمانی ساخت گراف، BFS و چک کردن نهایی یال‌ها همگی از $O(n)$ هستند پس پیچیدگی زمانی کل الگوریتم از $O(n)$ است.

سوال ۲:

الگوریتمی ارائه کنید که در زمان $O(V+E)$ وجود دور در یک گراف جهت‌دار را بررسی کند.

پاسخ

فرض کنید از راس 1 الگوریتم DFS را شروع می‌کنیم. گراف در صورتی دارای دور است که در اجرای این الگوریتم به راسی برسیم که قبل از آن در DFS از راس 1 آن را دیده باشیم. در نتیجه آرایه‌ای به نام $currentDfsTraversal$ تعریف کرده و بعد از رسیدن به راس i ، $currentDfsTraversal[i]$ در صورتی که $False$ باشد برابر $True$ قرار می‌دهیم. در صورتی که مقدار $currentDfsTraversal[i]$ برابر $True$ بود، گراف حاوی دور است. از آنجایی که گراف جهت دار است، ممکن است با یکبار اجرای الگوریتم DFS تمامی رئوس دیده نشوند. در نتیجه نیاز است تا دیده شدن تمامی رئوس الگوریتم اداامه پیدا کند. برای اینکار آرایه‌ای به نام $visited$ تعریف می‌کنیم. دقت کنید در صورتی که $currentDfsTraversal[i]$ برابر $False$ و $visited[i]$ برابر $True$ باشد، راس i در DFS های قبلی بررسی شده و دیگر نیازی به ادامه‌ی DFS از آن راس نیست.