به نام خدا

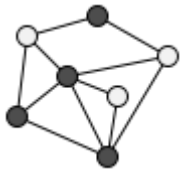
دل آ را فرقانی 9531064

••گزارش کار مربوط به رنگ آمیزی گراف به روش **Weak Coloring**••

توضیحی در ارتباط با الگوریتم weak coloring :

در نظریه گراف ، weak coloring یک روش خاص برای برچسب زدن به راس های یک گراف است.
برای مثال k-weak coloring برای یک گراف مانند G(V,E) ، یک رنگ به هر راس گراف ($v \in V$ $c(v) \in \{1, 2, ..., k\}$) اختصاص می دهد . به این ترتیب هر یک از راس های غیرایزوله با حداقل یک رأس با رنگ متفاوت مجاور هستند.
در نماد ریاضی ، برای هر $v \in V$ غیر ایزوله ، یک راس $u \in V$ به طوری که $\{u,v\} \in E$ و $c(v) \neq c(u)$ وجود دارد.

شکل زیر یک 2-weak coloring از یک گراف را نشان می دهد. هر رأس مشکی در مجاورت حداقل یک رأس سفید قرار گرفته است و بالعکس.
این الگوریتم به دلیل این که صرفا چک می کند در مجاورت هرراس حداقل یک راس با رنگ متمایز وجود داشته باشد ، در حالت کلی برای گراف رنگ آمیزی درستی را ارایه نمی دهد و ممکن است دو راس مجاور دارای رنگ یکسانی باشند .



الگوریتم پیاده سازی شده در این پروژه :

در الگوریتم پیاده سازی شده در این پروژه برای رسیدن به k-weak coloring ابتدا گراف را به صورت یک لیست همسایگی ذخیره می کنیم ، هر راس به صورت یک شی به نام Node ذخیره می شود که دارای ویژگی هایی مانند شماره راس ، رنگ راس و عمق راس در درخت ساخته شده از گراف ، توسط الگوریتم BFS است .

برای رنگ آمیزی گراف نیز ابتدا از یک راس دلخواه (که این راس در الگوریتم پیاده سازی شده اولین راس موجود در لیست همسایگی است) شروع میکنیم و با الگوریتم BFS گراف را طی میکنیم . برای هر راس که از آن می گذریم عمق آن را به اندازه ی یک واحد افزایش نسبت به عمق parent آن قرار میدهیم .

در نهایت تمام راس های موجود در لیست همسایگی دارای عمق مشخص هستند . پس از آن این راس ها را نسبت به عمقشان توسط Insertion sort مرتب می کنیم . با این کار ارتفاع درخت مشخص می شود .

سپس تعداد رنگ های موجود که با آن گراف را رنگ آمیزی می کنیم مشخص می کنیم . اگر تعداد رنگ ها از ارتفاع درخت بیش تر بود به ترتیب راس های موجود در هر سطح را با رنگ های متمایز رنگ آمیزی می کنیم و اگر تعداد رنگ ها از ارتفاع درخت کمتر بود تا جایی که رنگ ها تمام نشده اند ، راس های هر سطح را با رنگ های متمایز رنگ می کنیم و بعد از آن تا ارتفاع درخت تمام نشده است رنگ ها در سطح ها تکرار می شود .

ورودی های داده شده به برنامه :

Input :	output:
First Example: 1 2 1 3 1 4 1 5 2 1 2 3 2 4 2 5 3 1 3 2 3 4 3 5 4 1 4 2 4 3 4 5 5 1 5 2 5 3 5 4	1 : 0 2 : 1 3 : 1 4 : 1 5 : 1
Second Example: 1 2 3 1 3 7 2 6 2 7 7 6 1 8 4 8 8 7 4 6	1 : 0 5 : 0 2 : 1 3 : 1 8 : 1 7 : 2 6 : 2 4 : 2
Third Example: 1 2 2 4 4 1 7 6 7 4 6 4 3 4 3 5 4 5	1 : 0 2 : 1 4 : 1 7 : 2 6 : 2 3 : 2 5 : 2

پیچیدگی زمانی الگوریتم :

هزینه ی ساخت گراف و ساخت لیست همسایگی از روی فایل csv : O(n^2)

هزینه ی Breath First Search در گراف : O(VE)

هزینه ی Insertion Sort : O(n^2)

هزینه ی رنگ آمیزی گراف با الگوریتم رنگ آمیزی ضعیف : O(n)

نقاط قوت و ضعف مربوط به الگوریتم پیاده سازی شده :

نقطه ی ضعف این الگوریتم این است که گراف درست رنگ آمیزی نمی شود و این الگوریتم improper است .