علوم رایانش و فناوری اطلاعات
نشریه علمی انجمن کامپیوتر ایران
مجلد ۱۸، شماره ۱، ۱۳۹۹
صفحات ۷- ۱
مقاله عادی

رنگ‌آمیزی بدون تداخل برای مستطیل‌های متحرک

محمدعلی آبام[۱*]، لیلا بیابانی[۲]

[*]نویسنده مسئول، دریافت: ۹۹/۰۱/۲۲، بازنگری: ۹۹/۰۴/۲۳، پذیرش: ۹۹/۰۸/۱۹

[۱] استادیار، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران، ایران
[۲] دانش‌آموخته کارشناسی ارشد، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران، ایران

چکیده

مجموعه‌ی S متشکل از n ناحیه با نوعی ثابت (مانند دایره‌ها یا مستطیل‌های موازی محور) مفروض است. یک رنگ‌آمیزیِ بدون تداخل برای S اختصاص رنگ به ناحیه‌هاست به‌گونه‌ای که برای هر نقطه‌ی p که حداقل یک ناحیه از S شامل آن باشد، ناحیه‌ای وجود داشته باشد که رنگ آن در میان تمام ناحیه‌های مشمول p یکتا باشد. هدف ما در این مقاله، نگه‌داری رنگ‌آمیزی بدون تداخل برای مستطیل‌ها در حالت متحرک است. ناحیه‌ها در حال حرکت هستند، پس بعد از برخوردِ دو ناحیه، ممکن است رنگ‌آمیزی دیگر بدون تداخل نماند و نیاز به تغییر رنگ تعدادی از ناحیه‌ها باشد. هدف ما کمینه کردن تعداد رنگ‌های استفاده شده، تشخیص زمان‌های برخورد و همچنین کمینه کردن تعداد ناحیه‌هایی است که بعد از برخورد مجدداً رنگ‌آمیزی می‌شوند. در این پژوهش، مسئله‌ی رنگ‌آمیزی بدون تداخل را برای مستطیل‌های متحرک در صفحه و همچنین مستطیل‌های متحرک روی محور طول‌ها بررسی کرده‌ایم. الگوریتمی برای رنگ‌آمیزی بدون تداخل مستطیل‌های متحرک در صفحه ارائه می‌کنیم که از $O(\log^4 n)$ رنگ استفاده می‌کند و برای هر رویداد $O(\log n)$ رنگ‌آمیزی مجدد انجام می‌دهد. همچنین برای مستطیل‌های متحرک بر روی محور، الگوریتمی با استفاده از $O(\log^2 n)$ رنگ و $O(\log n)$ رنگ‌آمیزی مجدد برای هر برخورد ارائه می‌دهیم.

کلمات کلیدی: رنگ‌آمیزی بدون تداخل، ناحیه‌های متحرک، داده ساختار جنبشی

۱- مقدمه

۱-۱- رنگ‌آمیزی بدون تداخل[۱]

مسئله‌ی رنگ‌آمیزی بدون تداخل، اولین بار سال ۲۰۰۳ در [۱] و [۲] مطرح شد. انگیزه‌ی این مسئله، اختصاص فرکانس در شبکه‌های تلفن همراه است. شبکه‌های تلفن همراه، از دو نوع رأس تشکیل شده‌اند: *ایستگاه‌های پایه* (که مانند سرور عمل می‌کنند) و *کلاینت‌ها*. ارتباط بین کلاینت‌ها و ایستگاه‌های پایه، با پیوندهای رادیویی انجام می‌شود. به ایستگاه‌های پایه، برای پیوند آن‌ها به کلاینت‌ها، فرکانس‌های ثابتی اختصاص داده می‌شود. کلاینت‌ها نیز به‌طور مداوم در حال جستجوی فرکانس برای پیدا کردن یک ایستگاه پایه با دریافت خوب هستند. دریافت خوب تنها زمانی میسر می‌شود که ۱) ایستگاه پایه در محدوده‌ی دریافت باشد. ۲) به هیچ ایستگاه دریافتی در محدوده‌ی کلاینت، فرکانس مشابهی اختصاص داده نشده باشد (برای جلوگیری از تداخل). لذا اساس مسئله‌ی اختصاص فرکانس، اختصاص فرکانس به ایستگاه‌های پایه است طوری که هر کلاینت، بتواند در محدوده‌اش، ایستگاه پایه‌ای با فرکانس یکتا پیدا کند. به دلیل پرهزینه و محدود بودن طیف فرکانس‌ها، هدف به حداقل رساندن تعداد فرکانس‌های استفاده شده است.

رنگ‌آمیزی بدون تداخل در حالت هندسی، به دو صورت زیر که دوگان یک‌دیگر هستند، بررسی می‌شود:

- **رنگ‌آمیزی ناحیه‌ها**[۲]: خانواده‌ی متناهی S از n ناحیه با نوعی ثابت (مانند دایره‌ها یا مستطیل‌های موازی محور) داده شده است. می‌خواهیم کوچک‌ترین عدد طبیعی k را بیابیم که بتوان با استفاده از k رنگ، به هر ناحیه از S یک رنگ اختصاص داد، به‌طوری‌که در رنگ‌آمیزی حاصل، شرط پیشِ رو برقرار باشد: برای هر نقطه‌ی $p \in \cup_{b \in S} b$ حداقل یک ناحیه‌ی $b \in S$ وجود داشته باشد که p داخل آن باشد و رنگ آن در بین تمام ناحیه‌های مشمول p یکتا باشد. این رنگ‌آمیزی را رنگ‌آمیزی بدون تداخل S می‌نامیم.

- **رنگ‌آمیزی فضای محدوده**[۳]: مجموعه‌ی P از n نقطه در $\mathbb{R}^d$ و مجموعه‌ی R از محدوده‌ها (به‌عنوان‌مثال مجموعه‌ی تمام دایره‌های صفحه) داده شده است که آن را فضای محدوده‌ی (P, R) می‌نامیم. هدف یافتن کوچک‌ترین عدد طبیعی k است، به‌طوری‌که رنگ‌آمیزی نقاط با k رنگ وجود داشته باشد، به شرطی که برای هر $r \in R$ که $P \cap r \neq \emptyset$ نقطه‌ی $q \in P \cap r$ وجود داشته باشد که رنگ اختصاص داده شده به آن، در میان تمام نقاط $P \cap r$ یکتا باشد. این رنگ‌آمیزی را رنگ‌آمیزی بدون تداخل فضای محدوده‌ی (P, R) (و یا رنگ‌آمیزی بدون تداخل P باتوجه به R) می‌نامیم.

مسئله‌ی رنگ‌آمیزی بدون تداخل برای اولین بار در سال ۲۰۱۴ در [۳] برای حالت جنبشی بررسی شد؛ در آن پژوهش، رنگ‌آمیزی بازه‌های متحرک و رنگ‌آمیزی نقاط متحرک با توجه به دایره‌ها مطالعه شد. در حالت جنبشی نیز مانند حالت ساکن، هدف کمینه کردن تعداد رنگ‌های استفاده شده است. همچنین به دلیل حرکت اشیاء، ممکن است در زمان‌هایی نیاز به ایجاد تغییر در رنگ‌آمیزی و رنگ‌آمیزی مجدد تعدادی از اشیاء داشته باشیم. فرض می‌کنیم رنگ‌آمیزی مجدد عملی پرهزینه است، پس هدف دیگر مسئله، کمینه کردن تعداد رنگ‌آمیزی‌های مجدد پس از هر رویداد[۴] است.

یک رنگ‌آمیزی بدون تداخل بدیهی، اختصاص رنگی یکتا برای هر شیء است. در این صورت، چون رنگ هیچ دو شیء ای یکسان نیست، هیچ‌گاه نیاز به رنگ‌آمیزی مجدد نخواهیم داشت، اما تعداد رنگ‌های استفاده شده n است که بسیار زیاد است. یک روش بدیهی دیگر این است که پس از هر رویداد، یک رنگ‌آمیزی جدید را برای اشیا در آن لحظه به دست آوریم. در این روش، تعداد رنگ‌های مورد نیاز مانند حالت ساکن است ولی ممکن است بعد از هر رویداد، نیاز به رنگ‌آمیزی مجدد n شیء داشته باشیم. لذا در حالت جنبشی، هدف یافتن الگوریتمی ست که بین تعداد رنگ‌های مورد نیاز و تعداد رنگ‌آمیزی‌های مجدد بعد از هر رویداد مصالحه داشته باشد.

۱-۲- داده ساختارهای جنبشی[۵]

الگوریتم‌هایی که با اشیای در حال حرکت مواجه هستند، به‌طور سنتی در زمان‌های گسسته نمونه‌گیری می‌کنند و بر اساس وضعیت اشیاء در این زمان‌ها، ساختار مورد استفاده‌شان را به‌روزرسانی می‌کنند. انتخاب طول فاصله‌ی مناسب برای نمونه‌گیری‌ها، یک مسئله‌ی جدی در این رویکرد است. مطلوب برای رفع این مشکل این است که بدانیم دقیقاً در چه نقطه‌ای نیاز به انجام عملیات داریم. در واقع مختصات یک شئ باید تنها درصورتی‌که موجب یک تغییر واقعی در داده ساختار می‌شود، مورد توجه قرار گیرد. داده ساختارهای جنبشی که توسط بوش و همکاران [۴،۵] معرفی شد، دقیقاً چنین کاری می‌کنند: علاوه بر ساختار، اطلاعاتی نگه‌داری می‌شود که به کمک آن‌ها می‌توان به دست آورد که چه زمانی در ساختار، یک تغییر جدی رخ می‌دهد.

داده ساختار جنبشی، ساختاری است که ویژگی[۶] خاصی از مجموعه‌ی اشیای در حال حرکت (به‌عنوان‌مثال پوسته‌ی محدب نقاط) را نگه‌داری می‌کند، و از دو بخش تشکیل شده است: توصیفی ترکیبی از ویژگی مورد نظر مسئله و مجموعه‌ای از گواه‌ها[۷] که تست‌های ابتدایی بر روی اشیای ورودی هستند. خاصیت گواه‌ها این است که مادامی‌که نتیجه‌ی آن‌ها تغییری نکند، ویژگی مورد نظر مسئله تغییر نمی‌کند. به‌عبارت‌دیگر، مجموعه‌ی گواه‌ها یک اثبات را تشکیل می‌دهند که توصیف ترکیبی فعلی از ویژگی مورد نظر هنوز درست است. فرض بر این است که هر شئ معادله‌ی حرکت شناخته شده‌ای دارد، لذا زمان ابطال[۸] هر گواه را می‌توان محاسبه کرد. هرگاه یک گواه باطل می‌شود - که آن را یک *رویداد* می‌نامیم - داده ساختار جنبشی نیاز به به‌روزرسانی دارد. برای دانستن زمان ابطال گواه بعدی، زمان‌های ابطال در یک صف رویداد با اولویت زمان ذخیره می‌شوند.

کارایی یک داده‌ساختار جنبشی بر اساس چهار معیار زیر سنجیده می‌شود [۶].

- **پاسخگویی**[۹]: زمان مورد نیاز برای به‌روزرسانی بعد از ابطال هر گواه، یک معیار بسیار مهم کارایی است که زمان پاسخگویی نامیده می‌شود. اگر زمان پاسخگویی پلی‌لگاریتمیک باشد، داده ساختار جنبشی پاسخگو اطلاق می‌شود.
- **فشردگی**[۱۰]: فشردگی یک داده ساختار جنبشی، تعداد گواه‌هایی است که ذخیره می‌شود. داده ساختار فشرده است اگر تعداد گواه‌ها همواره نزدیک خطی[۱۱] باشد.
- **محلی بودن**[۱۲]: بیشینه تعداد گواه‌هایی که هر شئ درگیر آن است، محلیت داده ساختار جنبشی نامیده می‌شود. این معیار مهم است زیرا هنگامی‌که برنامه‌ی حرکت یک شئ تغییر می‌کند، نیاز است زمان ابطال همه‌ی گواه‌هایی که در آن‌ها درگیر بوده محاسبه‌ی مجدد شده و همچنین صف رویداد به‌روزرسانی شود. داده ساختار جنبشی محلی است اگر محلیت آن پلی‌لگاریتمیک باشد.
- **کارآمدی**[۱۳]: این معیار با تعداد رویدادهایی که باید پردازش شود ارتباط دارد. ابطال یک گواه لزوماً منجر به تغییر در ویژگی مورد نظر نمی‌شود؛ ممکن است تنها یک تغییر داخلی در داده ساختار جنبشی ایجاد شود. رویدادهایی که باعث تغییر در ویژگی موردنظر می‌شوند، *رویداد خارجی* و دیگر رویدادها، *رویداد داخلی* نامیده می‌شود. کارآمدی یک داده ساختار جنبشی، نسبت بیشینه تعداد رویدادهای داخلی و خارجی به بیشینه تعداد رویدادهای خارجی است. ازآنجاکه این مقدار وابسته به نوع حرکت اشیاء است، برای محاسبه فرض می‌شود حرکت اشیاء خطی و یا چندجمله‌ای با درجه‌ی ثابت است. داده ساختار جنبشی کارآمد است اگر کارآمدی آن پلی‌لگاریتمیک باشد.

معیار کارآمدی در مسئله‌ی رنگ‌آمیزی بدون تداخلِ ناحیه‌های متحرک، به دلیل ماهیت مسئله پیچیده است. در مسئله‌ای مانند سمت راست‌ترین نقطه، رویداد خارجی مستقل از الگوریتم است، اما در این مسئله، رویدادی خارجی تابعی از رنگ‌آمیزی ناحیه‌ها در مراحل قبلی است. به دلیل این پیچیدگی، برای الگوریتم‌های ارائه شده در این مقاله، کارآمدی را بررسی نخواهیم کرد.

۱-۳- مسئله‌ی مورد بررسی در این پژوهش

هدف این پژوهش بررسی رنگ‌آمیزی بدون تداخل برای ناحیه‌های متحرک است که در مسئله‌ی ۱ به‌صورت دقیق بیان شده است.

مسئله‌ی ۱. مجموعه‌ی S، شامل n ناحیه در صفحه با نوعی ثابت (مثلاً مستطیل‌های موازی محور) مفروض است. همچنین معادله‌ی حرکت هر ناحیه که تابعی پیوسته است، داده شده است؛ هرچند ممکن است ناحیه‌ها معادله‌ی حرکت خود را در زمان‌هایی تغییر دهند. تخصیص رنگ به ناحیه‌های عضو S، رنگ‌آمیزی بدون تداخل است اگر برای هر نقطه‌ی $p \in \bigcup_{b \in S} b$ حداقل یک ناحیه‌ی $b \in S$ وجود داشته باشد که p داخل آن باشد و رنگ آن در بین تمام ناحیه‌های مشمول p یکتا باشد. هدف پیدا کردن یک الگوریتم برای نگه‌داری رنگ‌آمیزی بدون تداخل برای S و تغییر رنگ‌ها در زمان‌هایی است که رنگ‌آمیزی دچار تداخل می‌شود، طوری که تعداد رنگ‌های مورد نیاز و تعداد ناحیه‌هایی که پس از هر برخورد رنگ‌آمیزی مجدد می‌شوند، کمینه باشد.

در بخش ۴ به مستطیل‌های موازی محورِ متحرک در صفحه می‌پردازیم و الگوریتمی برای رنگ‌آمیزی مستطیل‌های موازی محور ارائه می‌کنیم که از $O(\log^4 n)$ رنگ استفاده می‌کند و پس از هر رویداد، $O(\log n)$ رنگ‌آمیزی مجدد انجام می‌دهد، و هر رویداد در زمان $O(\log n)$ پردازش می‌شود.

در بخش ۵ مسئله را برای مستطیل‌های متحرک بر روی محور طول‌ها بررسی می‌کنیم. به‌عبارت‌دیگر، ناحیه‌های مورد بررسی، مستطیل‌های موازی محور در صفحه که ضلع پایینی‌شان روی محور طول‌ها قرار دارد هستند. الگوریتم پیشنهادی ما با استفاده از $O(\log^2 n)$ رنگ و $O(\log n)$ رنگ‌آمیزی مجدد پس از هر برخورد، رنگ‌آمیزی بدون تداخل را نگه‌داری می‌کند، همچنین هر برخورد در زمان $O(\log n)$ پردازش می‌شود.

۲- کارهای مرتبط پیشین

در این بخش در مورد چند نمونه از کارهای پیشین صحبت می‌کنیم. در [۲] الگوریتمی برای رنگ‌آمیزی نقاط با توجه به دایره‌ها با $O(\log n)$ رنگ معرفی شد و در [۷] ثابت شد که هر رنگ‌آمیزی بدون تداخلِ نقاط با توجه به دایره‌ها، برای هر وضعیتی از نقاط در صفحه از $\Omega(\log n)$ رنگ استفاده می‌کند. در [۸] الگوریتمی برای رنگ‌آمیزی ناحیه‌های با پیچیدگی اجتماع کم ارائه شد و به کمک این الگوریتم، الگوریتمی برای رنگ‌آمیزی مستطیل‌ها با $O(\log^2 n)$ رنگ مطرح شد. همچنین در این مرجع برای رنگ‌آمیزی نقاط با توجه به مستطیل‌ها، روشی ساده با$O(\sqrt{n})$ رنگ ارائه شد و بهبود $O(\sqrt{n}/\sqrt{\log n})$ نیز ذکر شد. در [۹] رنگ‌آمیزی شبه بدون تداخل معرفی شد و به کمک آن، الگوریتمی ارائه شد که با افزودن $O(n^{1-\epsilon})$ نقطه به مجموعه‌ی نقاط ورودی، از $\tilde{O}(n^{\frac{3}{8}+\epsilon})$ رنگ استفاده می‌کند. در [۱۰] به کمک رنگ‌آمیزی شبه بدون تداخل، الگوریتمی ارائه شد که نقاط را با $\tilde{O}(n^{0.382})$ رنگ، با توجه به مستطیل‌ها رنگ می‌کند.

برای اولین بار در [۳] مسئله‌ی رنگ‌آمیزی بدون تداخل در حالت جنبشی بررسی شد؛ الگوریتمی برای رنگ‌آمیزی بازه‌های متحرک با چهار رنگ و رنگ‌آمیزی مجدد $O(1)$ بازه پس از هر رویداد، و الگوریتمی برای رنگ‌آمیزی نقاط متحرک با توجه به دایره‌ها با $O(\log n)$ رنگ و رنگ‌آمیزی دوباره‌ی سرشکن $O(\log n)$ نقطه بعد از هر رویداد، مطرح شد. برای بازه‌های متحرک در [۱۱] تعداد رنگ‌آمیزی‌های مجدد بعد از هر رویداد به سه بهبود یافت، همچنین اشاره شد که روش در صورت انقباض و انبساط بازه‌ها در طول زمان نیز کار می‌کند.

دی برگ و همکاران [۱۲] در سال ۲۰۱۹، مسئله‌ی رنگ‌آمیزی متحرک در حالت پویا را برای مربع‌های واحد و چند حالت خاص از مستطیل‌ها بررسی کردند که در بخش ۳-۱ به تشریح آن می‌پردازیم. در حالت پویا ناحیه‌ها امکان درج و حذف دارند.

۳- پیش‌نیازها

در این بخش به بررسی و توضیح مختصر چند نمونه از کارهای پیشین که در بخش‌های بعد از آن‌ها استفاده خواهیم کرد، می‌پردازیم.

۳-۱- رنگ‌آمیزی بدون تداخل ناحیه‌های پویا

رنگ‌آمیزی بدون تداخل برای ناحیه‌های پویا در [۱۲] مطالعه شده است. در حالت پویا امکان درج و حذف ناحیه‌ها وجود دارد و هدف نگه‌داری رنگ‌آمیزی به‌صورتی است که بدون تداخل باقی بماند، همچنین تعداد ناحیه‌هایی که بعد از هر درج یا حذف مجدداً رنگ‌آمیزی می‌شوند، کمینه باشد. در این بخش تعدادی از الگوریتم‌های ارائه شده در [۱۲] را ذکر می‌کنیم.

۳-۱-۱- رنگ‌آمیزی بدون تداخل مستطیل‌های مهار شده[۱۴]

مستطیل r مهار شده نامیده می‌شود اگر رأس پایین-چپ آن روی مرکز مختصات قرار داشته باشد. S مجموعه‌ای از n مستطیل مهارشده است و مختصات x و y رأس بالا-راست مستطیل r به ترتیب با r_x و r_y نشان داده می‌شود. رنگ‌آمیزی بدون تداخل S با استفاده از یک درخت قرمز-سیاهِ افزوده شده انجام می‌شود.

برای سادگی فرض می‌شود که مختصات x (و به‌طور مشابه مختصات y) همه‌ی رئوس بالا-راست متمایز است. مجموعه‌ی S در درخت قرمز-سیاه افزوده شده‌ی T ذخیره می‌شود طوری که r_x کلید مستطیل $r \in S$ است. درخت به‌گونه‌ای ذخیره می‌شود که همه‌ی کلیدها در برگ‌ها باشند؛ برای این کار می‌توان در رئوس میانی مقادیر جداکننده‌ای قرار داد.

برای رأس $v \in T$، فرض کنید T_v زیردرخت با ریشه‌ی v و $S(v)$ مجموعه‌ی مستطیل‌های ذخیره شده در برگ‌های T_v است. برای هر رأس v در درخت، مستطیل $r_{max}(v)$ افزوده می‌شود که مستطیلی در $S(v)$ است که بیشینه r_y را دارد.

برای هر رأس میانی v، $\text{left}(v)$ و $\text{right}(v)$ به ترتیب بیانگر فرزند چپ و راست هستند. همچنین تابع $N(r)$ برای هر مستطیل $r \in S$ برابر مجموعه‌ی همه‌ی رئوس $v \in T$ است که v برگی ست که r را ذخیره می‌کند، یا v رأسی میانی ست که $r_{max}(\text{right}(v)) = r$.

تابع $\text{height}(v)$ بیانگر ارتفاع زیر درخت T_v است، پس اگر v برگ باشد، $\text{height}(v) = 0$ و برای رئوس غیر برگ $\text{height}(v) = \max(\text{height}(\text{left}(v)), \text{height}(\text{right}(v))) + 1$. تابع رنگ‌آمیزی $\text{col}(r)$ برابر $\max_{v \in N(r)} \text{height}(v)$ تعریف می‌شود.

ثابت می‌شود که رنگ‌آمیزی فوق بدون تداخل است، لذا رنگ‌آمیزی بدون تداخل با $O(\log n)$ رنگ و $O(\log n)$ رنگ‌آمیزی مجدد بعد از هر درج یا حذف در زمان $O(\log n)$ امکان‌پذیر است.

۳-۱-۲- رنگ‌آمیزی بدون تداخل مربع‌های واحد شامل مبدأ

مجموعه‌ی S شامل n مربع واحد شامل مبدأ مفروض است. یک راه‌حل بدیهی برای رنگ‌آمیزی بدون تداخل S به کمک روش بیان شده در زیربخش ۳-۱-۱، تقسیم هر مربع به چهار مستطیل مهار شده است. در این صورت به هر رأس مربع یک رنگ اختصاص می‌یابد و رنگ مختص هر مربع یک چهارتایی خواهد بود؛ لذا $O(\log^4 n)$ رنگ نیاز است. اما با استفاده از این نکته که ترتیب مختصات x گوشه‌های بالا-راست مربع‌های S با ترتیب گوشه‌های پایین-راست (یا بالا-چپ، و یا پایین-چپ) یکسان است، می‌توان به‌جای چهار درخت متفاوت از یک درخت استفاده کرد. در این مسئله علاوه بر $r_{max}(v)$، $r_{min}(v)$ نیز که به طریق مشابه تعریف می‌شود، به درخت افزوده می‌شود. در [۱۲] به کمک این اطلاعات افزود شده، تابع رنگ‌آمیزی با استفاده از $O(\log n)$ رنگ تعریف شده است.

۳-۱-۳- رنگ‌آمیزی بدون تداخل مستطیل‌های شامل مبدأ

مجموعه‌ی S متشکل از n مستطیل شامل مبدأ مفروض است. در این مسئله ترتیب مختصات گوشه‌های بالا-راست، ممکن است با ترتیب گوشه‌های بالا-چپ متفاوت باشد لذا ساختار درخت برای شرق مستطیل‌ها (برای گوشه‌های بالا-راست و پایین-راست) با ساختار درخت غرب مستطیل‌ها متفاوت است. پس می‌توان دو درخت جداگانه برای شرق و غرب در نظر گرفت و برای هر درخت، مانند قبل رنگ‌آمیزی را به دست آورد. در این صورت یک دوتایی مرتب از رنگ‌ها به هر مستطیل اختصاص می‌یابد؛ پس از $O(\log^2 n)$ رنگ استفاده می‌شود.

۳-۱-۴- رنگ‌آمیزی بدون تداخل مستطیل‌های دارای مختصاتی از جهان ثابت

اگر مختصات مستطیل‌ها از مجموعه‌ی ثابت جهان $U = \{0, 1, \dots, N-1\}$ با اندازه N باشد، می‌توان نتیجه را تعمیم داد. این فرض قابل تصور است زیرا محل‌های ممکن برای ایستگاه‌های پایه محدود هستند.

برای این مسئله، درخت بازه‌ایِ متعادل شده‌ی T_x با اسکلتی ثابت، روی جهان U در نظر گرفته می‌شود. مستطیل $r = [r_{x,1}, r_{x,2}] \times [r_{y,1}, r_{y,2}]$ به بالاترین رأس v در درخت T_x که $x(v) \in [r_{x,1}, r_{x,2}]$ است، اختصاص داده می‌شود. $S(v)$ مجموعه‌ی مستطیل‌های اختصاص داده شده به رأس v است. برای هر رأس $v \in T_x$ یک زیردرخت بازه‌ایِ متعادل شده‌ی $T_y(v)$ با اسکلت ثابت روی جهان ساخته می‌شود، و هر $r \in S(v)$ به بالاترین رأس w در زیردرخت $T_y(v)$ که $y(w) \in [r_{y,1}, r_{y,2}]$ است، اختصاص داده می‌شود. (در واقع یک درخت بازه‌ای دولایه روی مستطیل‌ها ساخته می‌شود که با استفاده از جهان، اسکلت آن ایجاد شده است. دلیل استفاده از این اسکلت ثابت این است که تحت درج و حذف نیازی به متعادل کردن نباشد.) اگر $S(w)$ مجموعه‌ی اشیای اختصاص داده شده به رأس w در هر زیردرخت $T_y(v)$ باشد، تمام مستطیل‌های $S(w)$ در نقطه‌ای مشترک خواهند بود، پس می‌توان با روش ارائه شده در زیربخش ۳-۱-۳، مستطیل‌های مجموعه‌ی $S(w)$ را با $O(\log^2 n)$ رنگ، رنگ‌آمیزی کرد طوری که بعد از هر درج یا حذف به $O(\log n)$ رنگ‌آمیزی مجدد نیاز است.

برای هر دو رأس w و w' در یک سطح از زیردرخت $T_y(v)$، هر دو مستطیل $r \in S(w)$ و $r' \in S(w')$ هیچ اشتراکی ندارند، پس اگر به هر سطح یک مجموعه‌ی رنگ اختصاص داده شود، برای رنگ‌آمیزی تمام رئوس $w \in T_y(v)$ به $O(\log N)$ مجموعه رنگ متمایز نیاز داریم. به‌طور مشابه برای هر دو رأس v و v' در یک سطح از درخت T_x، هر دو مستطیل $r \in S(v)$ و $r' \in S(v')$ هیچ اشتراکی ندارند. لذا تعداد کل مجموعه‌ی رنگ‌های لازم $O(\log^2 N)$ است.

با توجه به مطالب فوق، می‌توان مجموعه‌ی مستطیل‌ها را با استفاده از $O(\log^2 N \log^2 n)$ رنگ و با $O(\log n)$ رنگ‌آمیزی مجدد بعد از هر درج یا حذف، رنگ‌آمیزی کرد.

۴- رنگ‌آمیزی بدون تداخل مستطیل‌های موازی محور و متحرک در صفحه

در این بخش به رنگ‌آمیزی بدون تداخل مستطیل‌های موازی محور که در حال حرکت در صفحه هستند، می‌پردازیم. به کمک روش رنگ‌آمیزی بدون تداخل مستطیل‌های پویا و دارای مختصاتی از جهان ثابت، که در زیربخش ۳-۱-۴ بیان شد، روشی در زیربخش ۴-۱ مطرح می‌کنیم که با $O(\log^4 n)$ رنگ و رنگ‌آمیزی مجدد $O(\log n)$ مستطیل بعد از هر رویداد، یک رنگ‌آمیزی بدون تداخل را نگه‌داری می‌کند. این روش هر رویداد را در زمان $O(\log n)$ پردازش می‌کند. سپس در زیربخش ۴-۲ به تشریح داده ساختار جنبشی مورد نیاز و در زیربخش ۴-۳ به تحلیل الگوریتم می‌پردازیم.

۴-۱- رنگ‌آمیزی

در زیربخش ۳-۱-۴ الگوریتمی برای رنگ‌آمیزی بدون تداخل مستطیل‌های پویا با مختصاتی از جهان ثابت آورده شد. در ادامه به این موضوع می‌پردازیم که چگونه می‌توان n مستطیل متحرک در صفحه را به مستطیل‌های پویا با مختصاتی از $\{1, 2, \dots, 2n\}$ تبدیل کرد، و درنتیجه از الگوریتم یاد شده برای رنگ‌آمیزی بدون تداخل مستطیل‌های متحرک استفاده کرد به‌طوری‌که به ازای هر تغییر ترتیب دو طول و یا عرض، تعداد ثابتی درج و حذف انجام شود.

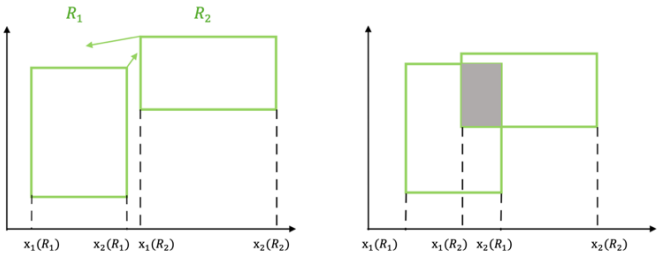
مجموعه‌ی X شامل همه‌ی xهای مستطیل‌ها را در نظر بگیرید. با فرض متمایز بودن همه‌ی xها، این مجموعه $2n$ عضو دارد. کافی است هر $x \in X$ را به رتبه‌ی آن در ترتیب صعودی نگاشت کنیم، بدین ترتیب سمت چپ‌ترین طول به ۱ و سمت راست‌ترین طول به $2n$ نگاشت خواهد شد. به همین ترتیب نیز می‌توان مجموعه‌ی همه‌ی عرض‌ها را به $\{1, 2, \dots, 2n\}$ نگاشت. روشن است که در صورت وجود یک رنگ‌آمیزی بدون تداخل برای مستطیل‌ها، مادامی‌که ترتیب هیچ دو x و یا هیچ دو yای نسبت به یک‌دیگر تغییر نکند، رنگ‌آمیزی بدون تداخل و معتبر خواهد ماند. پس تا زمانی که موقعیت‌های نسبی دستخوش تغییر نشده است، رنگ‌آمیزی و نگاشت نیازی به تغییر ندارند. هنگامی‌که ترتیب دو طول و یا دو عرض تغییر می‌کند، نگاشت تنها برای دو مستطیل مربوط به آن طول یا عرض معتبر نخواهد بود. در این صورت کافی است تنها دو مستطیل درگیر را از مجموعه‌ی مستطیل‌ها حذف کرده و سپس با ترتیب جدید به مجموعه اضافه کنیم.

رویه‌ی ۱ نشان می‌دهد که چگونه تداخل احتمالی پس از رویداد رفع می‌شود. برای نگه‌داری رنگ‌آمیزی بدون تداخل مجموعه‌ی مستطیل‌های پویا، از روش ارائه شده در [۱۲] که در زیربخش ۳-۱-۴ به آن پرداختیم استفاده می‌شود. شکل ۱ مثالی از یک رویداد را نشان می‌دهد.

رویه‌ی ۱: رسیدگی به رویداد (R_i و R_j)

ورودی: دو مستطیل R_i و R_j که در لحظه‌ی فعلی باعث ایجاد رویداد شده اند

1. مستطیل‌های R_i و R_j را از مجموعه‌ی مستطیل‌های پویا حذف کن
2. مستطیل‌های R_i و R_j را با رتبه‌های جدید مختصات که بعد از برخورد تغییر کرده است، به مجموعه‌ی مستطیل‌های پویا اضافه کن



شکل ۱- مثالی از برخورد دو مستطیل متحرک در صفحه. چپ: دو مستطیل $\boldsymbol{R_1}$ و $\boldsymbol{R_2}$ در حال نزدیک شدن به یک‌دیگر هستند. اگر رتبه‌ی هر $\boldsymbol{x}$ در مجموعه‌ی $\boldsymbol{X}$ را با $\boldsymbol{rank_x(x)}$ نشان دهیم، $\boldsymbol{rank_x(x_1(R_1)) = ۱}$، $\boldsymbol{rank_x(x_2(R_1)) = ۲}$ و $\boldsymbol{rank_x(x_1(R_2)) = ۳}$ و $\boldsymbol{rank_x(x_2(R_2)) = ۴}$ است. راست: اگر دو مستطیل $\boldsymbol{R_1}$ و $\boldsymbol{R_2}$ هم‌رنگ باشند، بعد از برخوردِ این دو مستطیل در ناحیه‌ی مشترک ایجاد شده که با رنگ خاکستری مشخص شده است، شرط بدون تداخل بودن رنگ‌آمیزی دیگر برقرار نیست. در لحظه‌ی رویداد، رویه‌ی ۱ این دو مستطیل را از مجموعه‌ی مستطیل‌های پویا حذف کرده و سپس آن‌ها را با رتبه‌ها مختصات جدیدشان به مجموعه‌ی پویا اضافه می‌کند. در این مثال $\boldsymbol{rank_x(x_1(R_2))}$ به ۳ و $\boldsymbol{rank_x(x_2(R_1))}$ به ۲ تغییر می‌کند.

۴-۲- داده ساختار جنبشی

همان‌طور که اشاره شد، رویداد در این روش، تغییر ترتیب دو طول و یا عرض متوالی است. بدین ترتیب برای هر دو x و یا هر دو y متوالی یک گواه در نظر گرفته می‌شود. داده ساختار جنبشی لازم شامل دو آرایه، یک صف اولویت و یک درخت بازه‌ای دولایه به شرح زیر است.

- آرایه‌ی A_x به طول $2n$ که xها را به ترتیب صعودی نگه‌داری می‌کند. به همین ترتیب آرایه‌ی A_y، yها را به ترتیب صعودی نگه‌داری می‌کند. بعد از هر برخورد، به‌سادگی با جابه‌جا کردن دو عنصر متوالی، می‌توان آرایه را به‌روزرسانی کرد.
- صف اولویت Q که برای هر گواه، یک عنصر با اولویت زمان ابطال نگه می‌دارد. بدین ترتیب، رویداد پیشِ رو در ابتدای صف اولویت قرار می‌گیرد و همچنین پس از پردازش هر رویداد، با تعداد ثابتی درج و حذف، می‌توان Q را به‌روزرسانی کرد.
- داده ساختار مورد استفاده در بخش ۳-۱-۴ که یک درخت بازه‌ای دولایه با اسکلت ثابت است. هر مستطیل به یکی از رئوس این درخت اختصاص داده می‌شود و برای رنگ‌آمیزی مستطیل‌های مختص هر رأس، لازم است یک درخت قرمز-سیاه افزوده شده، که در زیربخش ۳-۱-۱ تشریح داده شد، در نظر گرفته شود. این درخت در هنگام حذف یا افزودن یک مستطیل، به‌طور خودکار تداخل را تشخیص داده و با تغییر رنگ تعدادی از مستطیل‌ها آن را برطرف می‌کند. ازآنجاکه هر مستطیل تنها به یک رأس اختصاص می‌یابد، به‌روشنی می‌توان دید که پیچیدگی این ساختار $O(n)$ است.

۳-۴- تحلیل الگوریتم

قضیه‌ی ۱: برای مستطیل‌های متحرک در صفحه، رنگ‌آمیزی بدون تداخلی می‌توان نگه‌داری کرد که از $O(\log^4 n)$ رنگ استفاده کرده و برای هر رویداد $O(\log n)$ رنگ‌آمیزی مجدد انجام می‌دهد. همچنین هر رویداد در زمان $O(\log n)$ پردازش می‌شود.

اثبات. همان‌طور که در زیربخش ۴-۱ شرح دادیم، مسئله‌ی مستطیل‌های متحرک در صفحه قابل تبدیل به مسئله‌ی مستطیل‌های پویا با مختصاتی از جهان ثابت $\{1,2,\dots,2n\}$ است. اگر پس از هر رویداد، رویه‌ی ۱ را فراخوانی کنیم، با حذف دو مستطیل و درج دو مستطیل با مختصات جدید در مجموعه‌ی مستطیل‌های پویا، رنگ‌آمیزی بدون تداخل باقی خواهد ماند. الگوریتم مربوط به مستطیل‌های پویا با مختصاتی از جهان ثابت $\{0,1,\dots,N-1\}$ در زیربخش ۳-۱-۴ از $O(\log^2 N \log^2 n)$ رنگ استفاده می‌کند و هر درج یا حذف را با $O(\log n)$ رنگ‌آمیزی مجدد و در زمان $O(\log n)$ پردازش می‌کند، پس الگوریتم ما نیاز به $O(\log^4 n)$ رنگ دارد و هر رویداد با $O(\log n)$ رنگ‌آمیزی مجدد در زمان $O(\log n)$ پردازش می‌شود.

داده ساختار جنبشی مورد استفاده، هر رویداد را در زمان $O(\log n)$ پردازش می‌کند، پس پاسخگو است. تعداد گواه‌ها و پیچیدگی داده ساختار $O(n)$ است، لذا فشردگی نیز برقرار است. هر x (و به همین ترتیب هر y) فقط با دو x راست و چپ خود در یک گواه می‌تواند باشد، پس هر مستطیل حداکثر درگیر هشت گواه است و درنتیجه داده ساختار جنبشی محلی است.

در این الگوریتم تنها با موقعیت نسبی عرض‌ها و طول‌ها کار داریم. همچنین در صورت تغییر این وضعیت نسبی، رویداد رخ می‌دهد که خود الگوریتم رسیدگی کرده و به‌روزرسانی لازم را انجام می‌دهد. پس می‌توان نتیجه گرفت که در صورت انبساط و یا انقباض مستطیل‌ها، این الگوریتم همچنان کار می‌کند .

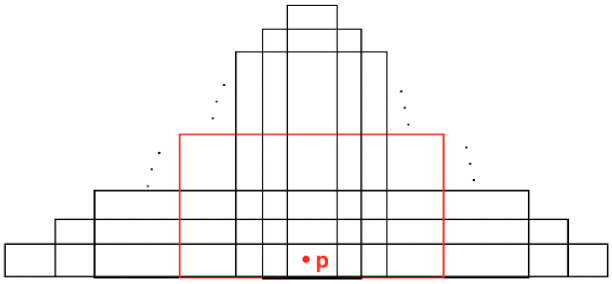
۵- رنگ‌آمیزی بدون تداخل مستطیل‌های متحرک بر روی محور طول‌ها

مسئله‌ی مورد بررسی در این بخش، رنگ‌آمیزی بدون تداخل مستطیل‌های متحرک روی محور است. در این بخش نیز مشابه رویکرد بخش ۴، مسئله را به مستطیل‌های پویای روی محور با مختصاتی از جهان ثابت $\{1,2,\dots,2n\}$ تبدیل می‌کنیم. روش ارائه شده با استفاده از $O(\log^2 n)$ رنگ و $O(\log n)$ رنگ‌آمیزی مجدد پس از هر برخورد، رنگ‌آمیزی بدون تداخل را نگه‌داری می‌کند، همچنین هر برخورد در زمان $O(\log n)$ پردازش می‌شود.

ابتدا در زیربخش ۵-۱ روش رنگ‌آمیزی را بیان می‌کنیم، سپس در زیربخش ۵-۲ به بررسی داده ساختار جنبشی می‌پردازیم، و در زیر بخش ۵-۳ الگوریتم را تحلیل می‌کنیم.

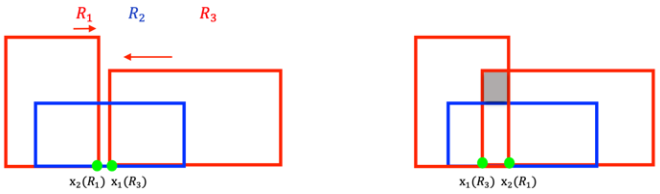
۵-۱- رنگ‌آمیزی

شکل ۱ نشان می‌دهد که $\Omega(\log n)$ رنگ لازم است. در این شکل نقطه‌ی p در تمام مستطیل‌ها مشترک است. پس باید مستطیلی با رنگ یکتا وجود داشته باشد. این مستطیل با رنگ قرمز نشان داده شده است. یکی از دو مجموعه‌ی مستطیل‌های با ارتفاع بیش‌تر از مستطیل قرمز و مستطیل‌های با ارتفاع کم‌تر، شامل حداقل نیمی از مستطیل‌های باقی مانده است. پس اگر تعداد رنگ‌های لازم $T(n)$ باشد، $T(n) \geq T(\lfloor n/2 \rfloor) + 1$ و درنتیجه $T(n) = \Omega(\log n)$.



شکل ۲- مثالی از وضعیت مستطیل‌ها که $\Omega(\log n)$ رنگ نیاز دارد.

مشابه بخش ۴ می‌توان به ترتیب هر طول و عرض را به رتبه‌ی آن در ترتیب صعودی مجموعه‌ی طول‌ها و عرض‌ها نگاشت. مادامی‌که دو مستطیل برخورد نکنند، نگاشت و همچنین رنگ‌آمیزی معتبر خواهد ماند. بعد از هر برخورد، نگاشت تنها برای دو مستطیل درگیر برخورد تغییر خواهد کرد. پس اگر یک الگوریتم برای رنگ‌آمیزی مستطیل‌های پویای روی محور داشته باشیم، می‌توانیم آن را برای مستطیل‌های متحرک روی محور نیز داشته باشیم، به‌صورتی که بعد از هر برخورد، دو مستطیل درگیر برخورد حذف و سپس با ترتیب جدید افزوده می‌شوند. شکل ۳ مثالی از یک رویداد را نشان می‌دهد.



شکل ۳- مثالی از برخورد دو مستطیل متحرک بر روی محور. چپ: دو مستطیل R_1 و R_3 در حال نزدیک شدن به یک‌دیگر هستند. اگر رتبه‌ی هر x در مجموعه‌ی X را با $rank_x(x)$ نشان دهیم، $rank_x(x_2(R_1)) = 3$ و $rank_x(x_1(R_3)) = 4$ است. راست: اگر دو مستطیل R_1 و R_3 هم‌رنگ باشند، بعد از برخورد این دو مستطیل در ناحیه‌ی خاکستری رنگ، شرط بدون تداخل بودن رنگ‌آمیزی برقرار نیست. در لحظه‌ی رویداد، رویه‌ی ۱ این دو مستطیل را از مجموعه‌ی مستطیل‌های پویا حذف کرده و سپس آن‌ها را با رتبه‌های مختصات جدیدشان به مجموعه‌ی پویا اضافه می‌کند. در این مثال $rank_x(x_2(R_1))$ به ۴ و $rank_x(x_1(R_3))$ به ۳ تغییر می‌کند.

جهت نگه‌داری رنگ‌آمیزی برای مستطیل‌های پویای روی محور نیز می‌توان مانند مستطیل‌های موازی محور که در بخش ۳-۱-۴ ذکر شد عمل کرد: با این تفاوت که درخت بازه‌ای با اسکلت ثابت را فقط برای xها می‌سازیم که درنتیجه درخت یک لایه خواهد بود. مستطیل‌های اختصاص شده به یک رأس درخت بازه‌ای، مستطیل‌های شامل یک نقطه‌ی مشترک خواهند بود. همان‌طور که در زیربخش ۳-۱-۳ ذکر شد، برای رنگ‌آمیزی شرق مستطیل‌های موازی محور و شامل یک نقطه‌ی مشترک، $O(\log n)$ کافی است. با یک دوران نود درجه‌ای به‌روشنی می‌توان دید که رنگ‌آمیزی شرق مستطیل‌های موازی محور و شامل یک نقطه‌ی مشترک، معادل رنگ‌آمیزی مستطیل‌های روی محور و شامل یک نقطه‌ی مشترک است، لذا مجموعه مستطیل‌های اختصاص داده شده به هر رأس درخت بازه‌ای را می‌توان با $O(\log n)$ رنگ، رنگ‌آمیزی کرد. برای هر سطح درخت نیز کافی است یک مجموعه رنگ مجزا در نظر گرفته شود. ازآنجاکه عمق درخت بازه‌ای با اسکلت ثابت $O(\log n)$ است، $O(\log n)$ مجموعه‌ی رنگ کافی است. پس در کل از $O(\log^2 n)$ رنگ استفاده می‌شود.

۵-۲- داده ساختار جنبشی

مادامی‌که برخوردی صورت نگیرد، وضعیت نسبی مستطیل‌ها تغییری نکرده و درنتیجه رنگ‌آمیزی بدون تداخل معتبر باقی می‌ماند. هر دو x متوالی را یک گواه این مسئله در نظر می‌گیریم، لذا رویداد در این مسئله، برخورد دو مستطیل است. داده ساختار جنبشی مورد استفاده برای این مسئله، مشابه داده ساختار استفاده شده در بخش ۴ است:

- آرایه‌ی A به طول $2n$ برای نگه‌داری xها به ترتیب صعودی.
- صف اولویت Q که با اولویت زمان برخورد، گواه‌ها را نگه‌داری می‌کند.
- درخت بازه‌ای با اسکلت ثابت T که هر رأس آن یک درخت قرمز-سیاه افزوده شده است. هر مستطیل، به یکی از رئوس این درخت بازه‌ای اختصاص داده می‌شود و برای رنگ‌آمیزی مجموعه مستطیل‌های اختصاص داده شده به هر رأس از یک درخت قرمز-سیاه افزوده شده استفاده می‌شود.

۵-۳- تحلیل الگوریتم

قضیه‌ی ۲: برای مستطیل‌های متحرک روی محور طول‌ها، رنگ‌آمیزی بدون تداخلی می‌توان نگه‌داری کرد که از $O(\log^2 n)$ رنگ استفاده می‌کند و برای هر رویداد $O(\log n)$ رنگ‌آمیزی مجدد انجام می‌دهد. همچنین هر رویداد در زمان $O(\log n)$ پردازش می‌شود.

اثبات. در بخش ۵-۱ بیان کردیم که مسئله‌ی مستطیل‌های متحرک بر روی محور، قابل تبدیل به مسئله‌ی مستطیل‌های پویای روی محور با مختصاتی از جهان ثابت $\{1, 2, \dots, 2n\}$ است. پس از هر رویداد، کافی است که رویه‌ی ۱ را فراخوانی کنیم. برای نگه‌داری رنگ‌آمیزی مجموعه‌ی مستطیل‌های پویای روی محور، همان‌طور که در زیربخش ۵-۱ شرح دادیم، می‌توان از روشی مشابه مستطیل‌های پویای در صفحه استفاده کرد، با این تفاوت که درخت بازه‌ای با اسکلت ثابت تنها یک لایه خواهد داشت. همان‌طور که توضیح دادیم، این روش از $O(\log^2 n)$رنگ استفاده می‌کند و هر رویداد با $O(\log n)$ رنگ‌آمیزی مجدد در زمان $O(\log n)$ پردازش می‌شود.

ازآنجاکه داده ساختار جنبشی مشابه داده ساختار زیربخش ۴-۲ و در واقع حالت ساده‌تری از آن است، پس استدلال‌های ذکر شده در زیربخش ۴-۳، برای خاصیت‌های پاسخگویی، فشردگی، و محلی بودن، برای این داده ساختار جنبشی نیز برقرار است.

با استدلالی مشابه بخش ۴-۳ نتیجه می‌شود که این الگوریتم تحت انبساط و انقباض نیز کار می‌کند.

۶- نتیجه‌گیری

در این مقاله، مسئله‌ی رنگ‌آمیزی بدون تداخل را برای مستطیل‌های موازی محور و متحرک در صفحه و همچنین مستطیل‌های متحرک روی محور بررسی کردیم. الگوریتمی برای مستطیل‌های متحرک در صفحه و الگوریتمی برای مستطیل‌های متحرک روی محور طول‌ها ارائه دادیم.

هدف ما در ادامه یافتن الگوریتم‌هایی برای رنگ‌آمیزی بدون تداخل سایر ناحیه‌ها و فضاهای محدوده است. در حال حاضر در حال بررسی رنگ‌آمیزی بدون تداخل برای مربع‌های واحد و متحرک هستیم. هدف یافتن روشی برای رنگ‌آمیزی با $O(\log n)$ و یا $O(\log^2 n)$ رنگ است. مسئله‌ی دیگر مورد بررسی، رنگ‌آمیزی نقاط با توجه به سه‌پهلو[۱۵]ها است، که به دنبال الگوریتمی با استفاده از $O(\log n)$ رنگ هستیم.

۷- مراجع

[1] S. Smorodinsky. *Combinatorial Problems in Computational Geometry.* PhD thesis, School of Computer Science, 2003.

[2] G. Even, Z. Lotker, D. Ron, and S. Smorodinsky. "Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks." *SIAM Journal on Computing*, 33(1):94– 136, 2003.

[3] T. Leijsen. *Kinetic conflict-free coloring.* Master's thesis, Eindhoven University of Technology, the Netherlands, 2014.

[4] J. Basch, L. J. Guibas, and J. Hershberger. "Data structures for mobile data." *Journal of Algorithms*, 31(1):1–28, 1999.

[5] J. Basch and L. J. Guibas. *Kinetic data structures*. PhD thesis, Department of Computer Science, Stanford University, United States, 1999.

[6] L. J. Guibas. "Kinetic data structure: a state of art report." In *Workshop on Algorithmic Foundations of Robotics,* pages 191–209, 1998.

[7] J. Pach and G. Tóth. "Conflict-free colorings." In *Discrete and computational geometry*, pages 665–671. Springer, 2003.

[8] S. Har-Peled and S. Smorodinsky. "Conflict-free coloring of points and simple regions in the plane." *Discrete & Computational Geometry*, 34(1):47–70, 2005.

[9] K. M. Elbassioni and N. H. Mustafa. "Conflict-free colorings of rectangles ranges." In *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, pages 254–263, 2006.

[10] D. Ajwani, K. M. Elbassioni, S. Govindarajan, and S. Ray. "Conflict-free coloring for rectangle ranges using $O(n^{0.382})$ colors." *Discrete & Computational Geometry*, 48(1):39–52, 2012.

[11] M. de Berg, T. Leijsen, A. Markovic, A. van Renssen, M. Roeloffzen, and G. J. Woeginger. "Fully-dynamic and kinetic conflict-free coloring of intervals with respect to points." In *28th International Symposium on Algorithms and Computation, ISAAC 2017, December 9-12, 2017, Phuket, Thailand*, pages 26:1–26:13, 2017.

[12] M. de Berg and A. Markovic. "Dynamic conflict-free colorings in the plane." *Computational Geometry*, 78:61–73, 2019.

محمدعلی آبام مدرک کارشناسی و کارشناسی ارشد مهندسی کامپیوتر خود را از دانشگاه صنعتی شریف به ترتیب در سال‌های ۱۳۷۸ و ۱۳۸۰ اخذ نموده است. او همچنین مدرک دکتری خود را از دانشگاه آیندهون هلند در زمینه علوم کامپیوتر در سال ۱۳۸۶دریافت نموده است. در حال حاضر ایشان استادیار دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف می‌باشند. زمینه تحقیقاتی موردعلاقه ایشان هندسه محاسباتی، الگوریتم‌های

تصادفی و الگوریتم‌های داده‌های حجیم می‌باشد. آدرس پست الکترونیکی ایشان عبارت است از:

abam@sharif.edu

لیلا بیابانی مدرک کارشناسی علوم کامپیوتر خود را از دانشگاه تهران در سال ۱۳۹۶ و کارشناسی ارشد مهندسی کامپیوتر خود را از دانشگاه صنعتی شریف در سال ۱۳۹۸ دریافت کرده است. از علاقه‌مندی‌های پژوهشی او می‌توان به هندسه‌ی محاسباتی و الگوریتم‌های داده‌های حجیم اشاره کرد. آدرس پست الکترونیکی ایشان عبارت است از:

biabani@ce.sharif.edu

---

[1] Conflict-Free Coloring
[2] Regions
[3] Range Space
[4] Event
[5] Kinetic Data Structures
[6] Attribute
[7] Certificates
[8] Failure
[9] Responsiveness
[10] Compactness
[11] Near Linear
[12] Locality
[13] Efficiency
[14] Anchored
[15] 3-sided

The CSI Journal on Computing Science and Information Technology

The CSI Journal on
Computing Science and
Information Technology
Vol. 18, No.1, 2020

Conflict-Free Coloring for Moving Rectangles

Mohammad Ali Abam, Leyla Biabani

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Abstract

Let S be a set of n regions of some fixed type (such as discs, axis-parallel rectangles, etc). A conflict-free coloring (CF-coloring) of S is an assignment of colors to each region such that for every point p covered by a non-empty subset of S, there is a region with a unique color among the regions containing p. We try to maintain a CF-coloring as the objects move. Regions are moving, so after an intersection, the coloring might not be conflict-free anymore, and we need to recolor some regions. The goal is to recognize the intersections and to minimize both the total number of colors used and the number of recolorings needed. In this paper, we study kinetic CF-coloring for moving axis-parallel rectangles in the plane, and also rectangles moving on the x-axis. We present an algorithms for axis-parallel rectangles that requires $O(\log^4 n)$ colors and $O(\log n)$recolorings per event. We also present an algorithm for rectangles moving on the x-axis, using $O(\log^2 n)$ colors and $O(\log n)$recolorings per event.

Keywords: Conflict-Free Coloring, Moving Objects, Kinetic Data Structure.