پروژه پایان‌ترم ساختمان داده
نیمسال تحصیلی 1404- 1403
دکتر امین طوسی

Quad Tree

مقدمه:

کواتری (Quad Tree) یک ساختار داده درختی است که برای تقسیم بازگشتی فضای دو بعدی به چهار بخش یا گره استفاده می‌شود. هر گره از کواتری یا یک مقدار مشخص دارد (برگ) یا به چهار زیر بخش تقسیم می‌شود (چهار فرزند). این ساختار به‌ویژه برای داده‌های دوبعدی مانند تصاویر، نقشه‌ها، و پردازش مکانی مفید است.

کاربردهای کواتری:

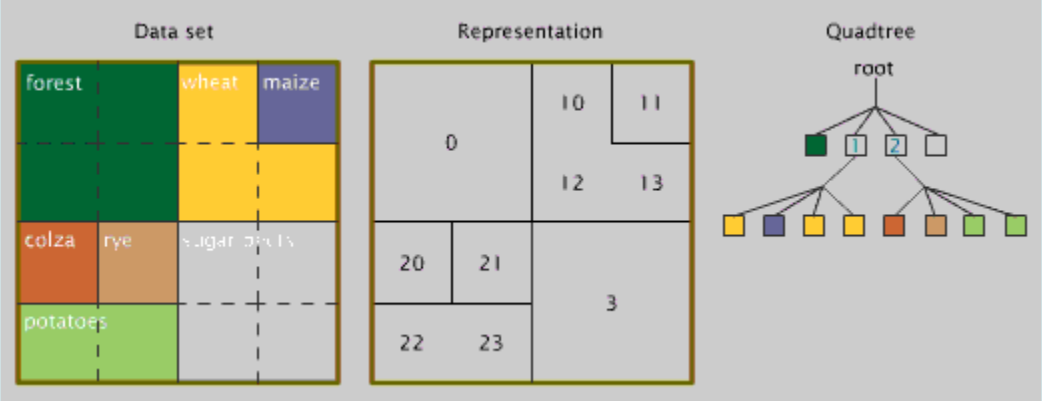
1. **پردازش تصاویر:** برای فشرده‌سازی تصاویر و ذخیره‌سازی اطلاعات مربوط به پیکسل‌ها.
2. **گرافیک کامپیوتری:** تقسیم‌بندی فضا برای تعیین برخورد اشیاء و شبیه‌سازی‌ها.
3. **سیستم‌های اطلاعات جغرافیایی (GIS):** مدیریت داده‌های مکانی، مانند ذخیره‌سازی نقشه‌ها و پرس‌وجوهای مکانی.
4. **نمایه‌سازی مکانی:** بهینه‌سازی جستجوی اشیاء دوبعدی در پایگاه‌های داده.
5. **شبیه‌سازی‌های علمی:** مانند مدل‌سازی جریان سیال یا دینامیک گازها که نیاز به تقسیم فضا دارند.

این ساختار به دلیل بازدهی بالا در کار با داده‌های بزرگ و پراکنده، در بسیاری از برنامه‌های کاربردی دو بعدی مورد استفاده قرار می‌گیرد.

در این پروژه قصد داریم با استفاده از کواتری، بر روی تصاویر فیلتر های مختلف اعمال کنیم و همچنین اطلاعات پیکسل های مورد نظر را بدست آوریم.

توضیحات کواتری:

1. ساختار درخت:



کواتری از یک ریشه شروع می‌شود که نمایانگر کل فضای دوبعدی است. هر گره درخت یا:

- **برگ (Leaf) است که داده‌ای را ذخیره می‌کند و دیگر تقسیم نمی‌شود.**
- **یا به چهار زیربخش (فرزند) تقسیم می‌شود که هر کدام بخشی از فضا را نمایندگی می‌کنند.**

2. فرایند ساخت کواتری:

1. **شروع از فضای اصلی:** با فضای کل شروع می‌کنیم.
2. **تقسیم فضا:** اگر داده‌های موجود در یک بخش از فضا بیش از حد مشخص (Threshold) باشد(در این پروژه چون ما با تصویر کار میکنیم، تا وقتی به خود پیکسل نرسیم همیشه در هر زیرفضا داده داریم)، آن فضا به چهار بخش کوچک‌تر تقسیم می‌شود:
   - **بالا-چپ (Top-Left)**
   - **بالا-راست (Top-Right)**
   - **پایین-چپ (Bottom-Left)**
   - **پایین-راست (Bottom-Right)**
3. **باز نگاشت (بازگشتی):**
   - **برای هر بخش، همین فرایند تکرار می‌شود تا زمانی که:**
     i. تعداد داده‌ها در هر بخش کمتر از مقدار آستانه باشد.
     ii. یا به عمق مشخصی از درخت برسیم.

3. جستجو در کواتری:

برای یافتن یک داده یا نقطه در کواتری:

1. از گره ریشه شروع می‌کنیم.
2. بسته به مکان نقطه، به یکی از فرزندان (ربع‌ها) مراجعه می‌کنیم.
3. این فرایند ادامه می‌یابد تا به گره برگ برسیم که داده موردنظر در آن قرار دارد.

مثال ساده:

فرض کنید یک تصویر 16×16 داریم. اگر کواتری بخواهد این تصویر را فشرده کند:

- ابتدا کل تصویر به چهار قسمت 8×8 تقسیم می‌شود.
- هر قسمت که داده‌های یکنواختی داشته باشد (مثلاً تمام پیکسل‌ها سفید)، دیگر تقسیم نمی‌شود.
- بخش‌هایی که داده‌های متنوع دارند (مثلاً پیکسل‌های سیاه و سفید)، دوباره به چهار بخش کوچک‌تر تقسیم می‌شوند.

کلاس QuadTree

```java
public class QuadTree {
    // Constructor,Builds the QuadTree recursively
    public QuadTree(int[][] image) {}

    // Return the Depth of the tree
    public int TreeDepth() {}

    // Returns the depth of the pixel in the QuadTree
    public int pixelDepth(int px, int py) {}

    // Returns the subspaces that overlap with a rectangle
    private void searchSubspacesWithRange(int x1, int y1, int x2, int y2) {}

    // Compresses the image into a smaller size
    public int[][] compress(int newSize) {}

    // Masks subspaces that overlap with a rectangle
    public int[][] mask(int x1, int y1, int x2, int y2) {}


}
```

:QuadTree

یک آرایه دو بعدی که شامل پیکسل های عکس است را دریافت می‌کند و درخت مربوط به آن را تشکیل می‌دهد. تقسیم زیرفضاها تا زمانی ادامه پیدا می‌کند که یا زیرفضا فقط شامل یک پیکسل باشد و یا تمام پیکسل های داخل یک زیرفضا، همگی یک رنگ باشند.

:TreeDepth

عمق درخت را برمی‌گرداند.

:pixelDepth

مختصات یک پیکسل را دریافت می‌کند و عمق آخرین زیرفضایی که پیکسل در آن وجود دارد را برمیگرداند.

:searchSubspacesWithRange

مختصات دو راس یک مستطیل را دریافت، و زیرفضاهایی که برگ هستند و تمام آنها، یا بخشی از آنها در مستطیل حضور دارد را به عنوان یک عکس جدید نشان می‌دهد.(برای اینکه خروجی مربعی/مستطیلی باشد میتوانید پیکسل های سفید به عکس نهایی اضافه کنید تا خروجی قابل نمایش باشد.)

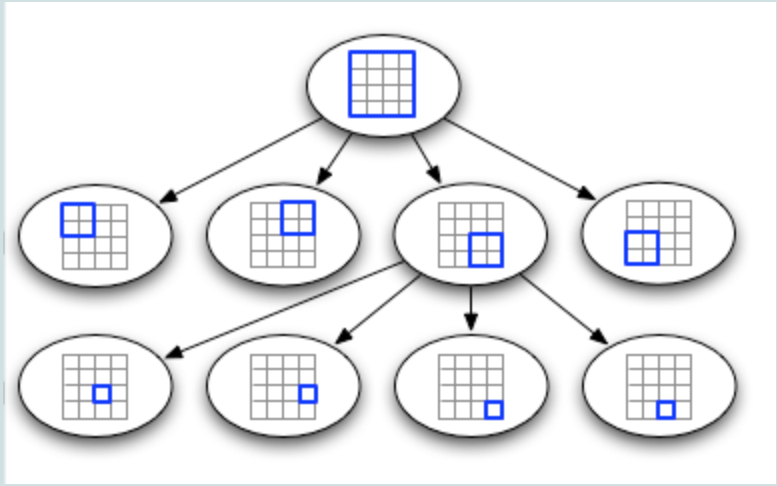
:mask

دقیقا مخالف تابع قبل است. مختصات دو راس یک مستطیل را دریافت، و رنگ زیرفضاهایی که برگ هستند و تمام آنها، یا بخشی از آنها در مستطیل حضور دارد را برابر سفید قرار می‌دهد. سپس عکس جدید که شامل بخش سفید شده و عکس اصلی است را نمایش می‌دهد.

:Compress

(محدود سازی عمق تولید درخت)

یک سایز دریافت میکند و سایز عکس را تغییر میدهد. این تغییر سایز باعث تار شدن تصویر میشود. به عنوان مثال اگر عکس 128*128 پیکسل داشته باشیم و ورودی عدد 8 باشد، سایز عکس به 8*8 تغییر می‌کند. این کار با میانگین گرفتن روی رنگ پیکسل هایی که در یک زیر فضا هستند انجام می‌شود. به عنوان مثال، در سایز های ذکر شده شما باید در زیر فضاهایی که تصویر را به قسمت های 16*16 تقسیم می‌کنند میانگین‌گیری کنید و عدد جدید را به عنوان یک پیکسل جدید ثبت کنید.



نمرات اضافه:

- اعمال تابع Compress بر روی ویدئو و پخش همزمان ویدئو
- پشتیبانی از تصاویر رنگی (برای تمامی توابع)

نکات تکمیلی:

- استفاده از هر زبان برنامه نویسی برای انجام پروژه مجاز است.
- در صورت نیاز می‌توانید متدهای کمکی پیاده‌سازی کنید.
- پیاده سازی ساختار درخت حتما باید با استفاده از LinkedList باشد که توسط خود شما پیاده سازی شده.
- عکس های هر مرحله باید قابلیت نمایش داده شدن، داشته باشند.
- سه تابع `mask` , `compress`, `searchSubspaceWithRange` باید روی درخت اصلی تغییرات را اعمال کنند و درخت جدید حاصل از تغییرات را نیز برگردانند.
- ورودی و سایز های خواسته شده در توابع، همگی توانی از 2 هستند.
- با توجه به اینکه پیکسل های هر عکس به صورت آرایه یک بعدی ذخیره می‌شوند، می‌توانید ورودی توابع را از آرایه دو بعدی به یک بعدی تغییر دهید.
- انجام پروژه به صورت تکی یا گروه های دو نفره است.
- در صورت مشاهده شباهت غیر متعارف میان پروژه افراد، نمره 100- برای هر دو گروه در نظر گرفته میشود.
- تسلط تمامی اعضا گروه به بخش‌های مختلف پروژه در هنگام تحویل الزامی است.
- در صورت هرگونه ابهام می‌توانید با Kiarash__ab یا Maresha82 در ارتباط باشید.(تلگرام)
- فایل‌های نهایی پروژه خود را در قالب زیر در سامانه vu بارگذاری کنید (بارگذاری توسط یکی از اعضای گروه کافیست):

FirstNamesLastNames_StudentNumbers_PR2.zip

موفق باشید