



**زیربرنامه:**

Extend(EDGE,Surface)

|  |  |  |
| --- | --- | --- |
| **توسعه دهندگان** | مرتضی نامور |  |
| **تهیه کنندگان مستند** | مرتضی نامور | |
| **تاییدکنندگان** | مرتضی نامور | |
| **تاریخ تنظیم سند** | 22/02/1394 | |
| **شناسه سند** | **MC2F003F1** | |
| **زبان برنامه‌نویسی** | **Fortran 90** | |

1. وظایف

این متد به منظور استخراج یک لایه از اضلاع و المان های متصل به آن نوشته شده است که در حوزه های hybrid قرار دارند. این متد دو ورودی دارد. اولی آدرس یک ضلع در حوزه hybrid که از نوع EDGE\* می باشد و در درون متد Ep نام می گیرد. دومین ورودی یک شئ از نوع surface به نام S است که طبیعتا یک شئ خالی است و قرار است در این متد پر بشود.

1. توضیحات و تئوری­ها

برای روشن تر شدن مفهموم عملکرد این متد به شکل زیر توجه کنید :

1. یک لایه از اضلاع و المان ها

در شکل فوق اضلاع قرمز رنگ اضلاع یک لایه می باشند و المان های مربعی دو طرف اضلاع قرمز رنگ نیز المان های یک لایه هستند. ابتدا و انتهای یک لایه ممکن است اضلاع مرزی قرار بگیرند و یا اینکه لایه به اضلاع لایه رابط متصل شوند . برای مثال اضلاع سبز رنگ سمت راست اضلاع مرزی بوده و اضلاع آبی رنگ سمت چپ اضلاع لایه رابط می باشند که المان های طرفین آن با یک دیگر مشابه نیستند.

هدف متد extend این است که اگر یکی از اضلاع لایه (اضلاع قرمز رنگ) به عنوان ورودی به متد داده شد آنگاه متد بتواند تمامی اضلاع قرمز رنگ دیگر و المان های همسایه آن را استخراج کند . بدین منظور متد یک ضلع را می گیرد ، آن را به شئ S که از نوع surface می باشد اضافه می کند ، آنگاه از یک طرف ضلع ورودی (به طور مثال از سمت چپ) سعی می کند ضلع متصل به آن سمت ضلع ورودی را که هم راستای خود ضلع است ، پیدا کند و مجددا این ضلع پیدا شده را به شئ S اضافه کند. از آن جایی که به آن گره از ضلع ورودی ، سه ضلع دیگر متصل شده و فقط یکی از آن ها با ضلع ورودی هم راستاست ، لذا پیدا کردن این ضلع نوعی چالش است. که در هنگام توضیح دستورات متد به آن اشاره می کنیم . نکته حائز اهمیت در این قسمت این است که متد extend یک متد بازگشتی است و خود را فراخوانی می کند. یعنی ابتدا یک ضلع به متد داده می شود. آن را وارد شئ S می کند . ضلع هم راستای ضلع ورودی را پیدا می کند آن گاه این ضلع پیدا شده را به عنوان ضلع ورودی جدید به خود متد extend می دهد. این کار آن قدر ادامه پیدا می کند تا دیگر نتواند ضلعی را به عنوان ضلع ورودی به متد بدهد. توجه شود که چون ما از یک سمت حرکت کرده ایم (به طور مثال سمت چپ) لذا فقط اضلاع سمت چپ اولین ضلع را پیدا می کنیم. برای پیدا کردن اضلاع سمت راست ، زمانی که متد نتواند دیگر از سمت چپ به مسیر خود ادامه دهد بر می گردد به اولین ضلع ورودی و این بار از سمت راست این ضلع ادامه می دهد تا به انتهای مسیر برسد. زمانی که از سمت راست نتواند دیگر ادامه دهد می توان از متد خارج شد و اضلاع و المان های ذخیره شده در S اکنون یک لایه را تشکیل می دهند. به شکل زیر دقت کنید:

1. گام اول در استخراج لایه

فرض کنید ضلع زرد رنگ به عنوان اولین ضلع ورودی به متد داده شود. متد ابتدا این ضلع را وارد شئ S می کند و سپس ضلع سمت چپ را پیدا کرده و به عنوان ورودی مجدداً به متد می دهد :

1. گام دوم در استخراج لایه

در قدم بعدی ضلع سمت چپ ضلع دوم به عنوان ورودی به متد داده می شود :

1. گام سوم در استخراج لایه

مجدداً ضلع دیگر به متد داده می شود:

1. گام چهارم در استخراج لایه

پس از ورود این ضلع آخر به متد و قرار گرفتن ان در شئ S ، متد دیگر نمی تواند از این سمت ادامه پیدا کند زیرا به لایه رابط رسیده و المان های آن طرف لایه المان های مثلثی می باشد. حالا که متد نمی تواند به مسیرش ادامه دهد باید از سمت دیگر مسیر را طی کند. به این نکته توجه کنید تا کنون 4 مرتبه متد extend فراخوانی شده است (به صورت متداخل و بازگشتی ) ولی هنوز هیچکدام از آن ها بسته نشده اند. آخرین فراخوانی متد extend نمی تواند از سمت چپ ادامه دهد اکنون این متد باید به طریقی بتواند همان ضلع اول را مجددا به متد extend وارد کند تا متد از سمت راست ادامه پیدا کند. به همین منظور می توان از تعدادی flag استفاده کرد .زمانی که آخرین فراخوانی متد extend به سمت چپ رسید یکی از flag ها را مقدار می دهد. اکنون خود متد ضلع اول را فراخوانی کرده و به متد extend می دهد. حال متد دیگری فراخوانی شده و ضلع سمت راست ضلع اول را به S اضافه می کند (اولین ضلع آبی پر رنگ). مجددا در دل همان متد ، متد extend فراخوانی شده و دومین ضلع آبی رنگ را وارد می کند. در نهایت یک متد extend دیگر فراخوانی شده و سومین ضلع آبی رنگ را وارد می کند .

در شکل زیر این روال مشخص است:

1. گام های آخر در استخراج لایه

زمانی که دیگر این فراخوانی آخر نتواند از سمت راست ادامه دهد با توجه به شروطی که بر روی flag ها تعریف خواهد شد ، به ناچار این فراخوانی آخر بسته می شود. این اتفاق تکرار می شود تا به اخرین فراخوانی متد extend برسیم که همان ضلع Ep ورودی اولیه ما را فراخوانی کرده است. در اینجا یکی دیگر از flag ها را مقدار می دهیم. باز هم چون تمامی اضلاع برچسب خورده اند و flag های آن ها به گونه ای است که شروط را ارضا نمی کنند ، بسته می شوند تا به آخرین فراخوانی متد extend برسد که ضلع سمت چپ را فراخوانی کرده است. در اینجا با شروط روی flag ها آن هم بسته می شود . این کار مجددا تکرار شده تا به اولین فراخوانی برسیم که ضلع Ep را فراخوانی کرده است. در اینجا برنامه تمام می شود و شئ S تمامی اضلاع و المان ها را در خود جای داده است.

1. بخش­های زیربرنامه
2. تعریف پارامترهای اولیه و پیدا کردن المان های همسایه ضلع ورودی به متد

بدون توضیح

1. اضافه کردن ضلع ورودی به متد و المان های همسایه آن به وکتور S

در این بخش ابتدا سه شرط به صورت همزمان بررسی می شود. ابتدا آنکه المان های همسایه ضلع ورودی مشابه باشند ( مقدار خاصیت state برای ضلع ورودی برابر با 1 باشد) . دوم آنکه ضلع ورودی تاکنون برچسب نخورده باشد (مقدار خاصیت EdgIsLabl برابر 0 باشد) و سوم اینکه تعداد اضلاع المان های همسایه ضلع ورودی برابر با 4 باشد (المان ها مربعی باشند) . پس از اینکه این سه شرط برقرار بود آن گاه می توان ضلع ورودی و دو المان همسایه آن را به وکتور لایه S به کمک متد push\_back اضافه کرد. در اینجا لازم است که ضلع ورودی نیز برچسب هم بخورد.

1. ذخیره کردن دو ضلع عمودی متصل به گره اول از ضلع ورودی

به طور کلی هدف آن است که ضلع مجاور ضلع ورودی لایه را بیابیم. به شکل زیر توجه کنید:

Ep.P[1]

Ep.P[0]

Ep

1. نمایش یک لایه به همراه اضلاع آن

مطابق شکل فوق هدف پیدا کردن ضلع قرمز رنگ متصل به گره آبی رنگ است. این ضلع مجاور ضلع ورودی لایه (ضلع نارنجی) می باشد که به گره اول آن متصل است.

بدین منظور در این بخش دو ضلع آبی رنگ را پیدا می کنیم و در بخش بعد ضلع قرمز رنگ را می یابیم.

برای پیدا کردن دو ضلع آبی رنگ کافیست در بین اضلاع دو المان همسایه ضلع ورودی جستجو کنیم و آن اضلاعی را بیابیم که یکی از گره های آن ها منطبق بر گره اول ضلع ورودی باشد و همچنین این اضلاع خود ضلع ورودی نباشند . با این تفاسیر دو ضلع آبی رنگ براحتی پیدا می شوند. آن ها را در یک آرایه ذخیره می کنیم.

1. پیدا کردن ضلع مجاور ضلع ورودی به متد

در این بخش کافیست در بین اضلاع متصل به گره اول ضلع ورودی جستجو کرده و آن ضلعی را بیابیم که مخالف خود ضلع ورودی و دو ضلع آبی رنگ بدست آمده در بخش قبل باشد. پس از پیدا کردن این ضلع بایستی در ادامه بررسی کنیم که گره اول آن منطبق بر گره اول ضلع ورودی نباشد. اگر منطبق بود جای گره اول و دوم ضلع پیدا شده را عوض می کنیم و این ضلع را مجددا به صورت بازگشتی به عنوان ورودی به متد extend معرفی می کنیم. این برنامه تا ایجا شروع به شناسایی اضلاع با ارضای شروط ذکر شده می کند .

این کار را تا جایی ادامه می دهد که دیگر در طرف گره صفرم آخرین ضلع شناسایی شده ، ضلعی وجود نداشته باشد تا به عنوان ضلع جدید بتواند به لایه آن را اضافه کند ( مثلا از سمت چپ ضلع Ep شروع کرده و به لایه رابط رسیده است.) مانند شکل زیر :

Ep.P[1]

Ep.P[0]

Ep

1. رسیدن متد به یک سمت اضلاع لایه

در این شکل مشخص است که به گره صفرم ضلع Ep آخر دیگر هیچ ضلعی با شرایط قرار گیری در لایه متصل نیست. به همین علت این فراخوانی متد extend به ادامه ی برنامه نوشته شده می رود ( توجه شود تا الان هیچ کدام از فراخوانی ها بیشتر از خط break پیشروی نکرده اند ، زیرا همه ی آن ها یک ضلع دیگر برای متد extend فرستاده اند و فراخوانی جدیدی انجام داده اند و منتظر اتمام فراخوانی داخلی می باشند). آخرین فراخوانی چون نمی تواند ضلعی برای متد extend بفرستد به ادامه برنامه (دستورات بعد از break ) می رود.

1. ایجاد شروط مناسب به منظور پیدا کردن اضلاع سمت گره دوم ضلع ورودی به متد

در اینجا باید با استفاده از flag های m و n شرایطی را فراهم کنیم که اضلاع سمت راست اولین ضلع ورودی نیز وارد شئ S شوند.

زمانی که آخرین فراخوانی به خطوط فوق می رسد بررسی می کند و می بیند که هم m=0 است و هم n=0 می باشد. پس شرط درست بوده و وارد این مجموعه دستورات می شود. اکنون اولین کاری که انجام می شود این است که یکی از flag ها مثلا m یک واحد به مقدارش اضافه شود لذا از این به بعد دیگر هیچ فراخوانی از متد extend نمی تواند وارد این مجموعه دستورات شود چون شرط داخل if نادرست خواهد بود. پس از ورود متد به این قسمت و افزایش مقدار m ، اکنون دو کار مهم باید انجام شود. قصد داریم تا اولین ضلع را مجددا برای متد extend بفرستیم به گونه ای که متد از سمت راست ان ضلع پیشروی کند. پس لازم است جای گره اول و دوم آن ضلع (اولین ضلع ورودی ) را عوض کنیم. همچنین برچسب آن را برداریم یعنی پارامتر is\_labeled آن را 0 کنیم تا بتواند از اولین شرط مهم مربوط به متد عبور کند. پس از این دو اصلاح ، ضلع اول را مجددا به درون متد extend می فرستیم . توجه کنید این فراخوانی توسط آخرین فراخوانی متد extend انجام شده برای آخرین ضلع سمت چپ فراخوانی می شود. اکنون فراخوانی های متد extend از سمت راست ضلع اول پیشروی کرده تا به انتهای مسیر برسند. زمانی که آخرین متد فراخوانی شد ، دیگر نمی تواند ضلعی را پیدا کند که تمامی شروط را برآورده کرده و وارد شئ S شود لذا مجددا از خط دستور break عبور می کند و به شروط مربوط به flag ها می رسد. در اینجا نیز چون m=1 شده است (توسط آخرین فراخوانی که آخرین ضلع سمت چپ را فراخوانی کرده بود) نمی تواند وارد مجموعه دستورات فوق بشود.

1. ایجاد شروط مناسب به منظور بسته شدن کامل تمامی متدهای بازگشتی باز شده تا این مرحله

آخرین فراخوانی متد extend وارد این شرط ها نیز نمی تواند بشود . برای مجموعه شرطی دوم چون هم n=0 است و هم ضلع ورودی ، اولین ضلع نمی باشد و برای مجموعه شرطی سوم نیز چون ضلع ورودی همان ضلع اول نمی باشد لذا نمی تواند وارد شود و به ناچار بسته می شود. فراخوانی قبل از آن نیز به همین دلایل بسته می شود. این روند ادامه پیدا می کند تا برسیم به آخرین فراخوانی اولین ضلع ورودی (از جایی که برنامه تصمیم گرفت از سمت راست حرکت کند) . در اینجا مجموعه شرط سوم ارضا خواهد شد چون هم n=0 می باشد و هم ضلع درون متد همان ضلع اول است. پس در اینجا یکی به مقدار n اضافه شده و n نیز برابر 1 می شود. پس از این کار این فراخوانی نیز بسته می شود. فراخوانی قبل از این مربوط ببه آخرین ضلع سمت چپ بود که وارد مجموعه شرط اول شده بود. اما این بار دیگر نمی تواند وارد آن مجموعه شود چون هم مقدار m و هم n برابر 1 شده است. این فراخوانی وارد دو مجموعه شرط بعدی هم نمی شود و به ناچار بسته می شود. فراخوانی های قبل از آن نیز به همین علت بسته می شوند تا در نهایت برسیم به اولین فراخوانی این متد که مربوط به همان ضلع اول بود . در اینجا توجه شود که مقدار n برابر 1 است و ضلع ورودی متد همان ضلع اول است . لذا برنامه وارد مجموعه شرط دوم شده مقدار m و n را صفر می کند و بسته می شود. اکنون شئ S ، لایه مورد نظر ما را تشکیل داده است.