**B+ tree visualization**

Documantation

این برنامه با استفاده از زبان C++ به دلیل پشتیبانی خوب از اشاره‌گرها و همچنین آسانی در طراحی ساختمان داده‌ها، کدنویسی شده است.

همچنین در کدنویسی تلاش شده است تا حد ممکن قواعد کد تمیز رعایت شود و کد به صورت صریح و خوانا باشد، همچنین کد کامنت گزاری شده است تا درک کد آسان تر شود.

شما در پوشه این برنامه، 6 فایل سی‌پلاس‌پلاس(با فرمت.cpp مشخص شده‌اند)، یک فایل کتابخنه)با فرمت.h) و. همچنین یک پوشه که شامل چندین تست کیس مختلف برای تست برنامه است مشاهده خواهید کرد.

فایل آغازگر برنامه plusTree.cpp می‌باشد که به جای main نشسته است و برنامه از اینجا آغاز می‌شود. در این فایل افزون بر فراخوانی کتابخوانه شخصی (BplusTree.h) توابع insertionFunction برای افزودن یک کلید به درخت، deleteFunction برای حذف یک کلید از درخت، searchFunction برای جستجو در درخت برای یافتن یک کلید و همچنین printFunction برای چاپ کردن درخت مشاهده می‌کنید.

در ابتدای شروع برنامه، کاربر باید max-degree برای درخت B+ را مشخص کند، max-degree حداکثر درجه این درخت است.

پس از مشخص نمودن max-degree، منوی برنامه به کاربر نمایش داده می‌شود که با استفاده از سوئیچ کیس پیاده سازی شده است. در این منو با انتخاب یکی از اعداد 1 تا 5 می توان عملیات موردنظر را فرخوانی کرد.

**فرآیند افزودن یک کلید به درخت**

با استفاده از کلید 1 شما می توانید یک کلید به درخت اضافه کنید، روند این عملیات بدین شکل است که با استفاده از کتابخانه نوشته شده، پارامتر کلید به فایل insert.cpp پاس داده خواهد شد و در صورتی که اولین کلید وارد شده باشد، یک نود مخصوص برای آن تشکیل می شود.

با استفاده از قوانین ساخت یک درخت، نود های بعدی، اگر از ریشه و سپس نود های فرزند آن کوچکتر باشد، در درخت به سمت چپ حرکت کرده و در جای مناسب اضافه می شود. در صورتی که اضافه شدن کلید به نود، باعث شود که max-degree=node.size() شود، باید نود جدید تشکیل شود و مرتب سازی درخت در آن شاخه صورت بگیرد. این فرآیند در تابع دیگری در هیمن فایل با نام insertInternal صورت گرفته است. اگر این مرتب سازی باعث شود که والد نیز به درجه max-degree برسید، دوباره باید آن شاخه تغییر یافته تا درخت به طور کامل مرتب شود. این فرآیند به دو صورت مرتب سازی نود سمت چپ، نود والد و نود سمت راست انجام می شود و برای اینکه در این فرآیند تداخلی رخ ندهد، از یک نود مجازی با نام virtualTreePtrNode کمک گرفته شده است.

همین رفتار نیز برای هنگامی که نود کلید از نود ریشه و نود های فرزند آن بزرگتر باشد خواهد افتاد.

پس از افزودن اولین کلید، برنامه اولین نود را تولید خواهد کرد، به همین دلیل در هنگام افزودن اولین نود به درخت، برنامه پیغام ": IS ROOT!!" به همراه کلید افزوده شده را نشان خواهد داد.

اگر این فرآیند به طور صحیح انجام شود، برنامه پیغام ":Inserted successfully " به همراه کلید افزوده شده را نمایش خواهد داد.

همچنین اگر نود ریشه در هنگام افزودن کلید به درخت اضافه شود، برنامه عبارت "Created new Root!" را نمایش خواهد داد.

فرآیند افزودن نود به درخت B+ یکی از فرآیند های پیچیده است که پیاده‌سازی های مختلفی از آن شده است اما این برنامه، طبق پیاده‌سازی پرطرفدار که سایت شبیه‌ساز <https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html> هم آن را پیاده سازی کرده است، برنامه نویسی شده است.

**فرآیند حذف یک کلید از درخت**

فرآیند حذف یک عنصر از درخت، دومین فرآیند پیچیده برنامه است، برای حذف یک عنصر از درخت در صفحه اصلی باید عدد 2 را انتخاب کنید، سپس درخت برای شما نمایش داده خواهد شد و از شما پرسیده می شود که قصد دارید کدامیک از کلید های درخت را حذف کنید. اگر کلیدی در درخت موجود نباشد، برنامه پیغام "Tree is Empty Now" را نمایش خواهد داد. اگر کلید انتخابی شما در درخت موجود نباشد، برنامه پیغام "Key Not Found in the Tree" را خواهد داد.

این فرآیند همانند قسمت قبلی انجام می شود با این تفاوت که تابع deleteFunction فراخوانی می‌شود و با استفاده از کتابخانه تابع removeKey موجود در فایل delete.cpp فراخوانی می‌شود.

فرآیند حذف یک کلید از درخت همانند فرآیند افزودن یک کلید به درخت انجام می‌شود؛ منظور از این جمله این است که هنگامی که ما یک کلید به درخت اضافه میکنیم، باید بررسی کنیم که درخت تناسب داشته باشد، در فرآیند حذف یک کلید از درخت نیز همواره پس از حذف باید بررسی کنیم که آیا با حذف این کلید از درخت، نود خود کلید حذف شده یا نود والد آن نیاز به تغییر دارند یا خیر؟

اگر این کلید در نود والد نیز تکرار شده باشد، باید از نود والد نیز حذف شود و اگر از نود والد حذف شود، آیا باید اشاره‌گر مربوط به آن به کلید دیگری اشاره کند و اگر آری، آن کلید کدام کلید است.

همچنین در فرآیند حذف یک کلید، ممکن است یک نود به طور کامل حذف شود و در این صورت باید نود والد آن نیز تغییر کند و گاهی این تغییرات اینقدر وابسته هستند که تا نود ریشه نیز تغییر خواهد کرد.

به طور خلاصه، همانند الگوریتم مذکور در فرآیند افزودن یک کلید به درخت، حذف یک کلید نیز صورت میگیرد و درخت بهینه خواهد شد.

پس از حذف کلید و تغییر در ساختار درخت، درخت نمایش داده خواهد شد.

**فرآیند جستجو در درخت**

فرآیند جستجو در درخت به صورت ساده انجام می‌شود، برای جستجو یک کلید در صفحه ابتدایی، عدد 4 را وارد کنید و کلید مورد نظر را به برنامه بدهید. روند آن همانند قسمت های قبلی با استفاده از تابع searchFunction در ابتدا و سپس با استفاده کتابخانه شخصی نوشته شده تابع مخصوص به آن در فایل search.cpp خواهیم رفت و با استفاده از binary search به دنبال آن کلید در درخت خواهیم پرداخت، در صورتی که درخت خالی باشد عبارت "NO Tuples Inserted yet" را نمایش می دهیم و در صورتی که کلید در درخت موجود باشد عبارت "Key FOUND" و در غیر این صورت عبارت "Key NOT FOUND" را نمایش خواهیم داد.

**فرآیند نمایش درخت**

نمایش مرحله ای درخت با استفاده از عدد از عدد 3 انجام می‌شود، روند این نمایش همانند قسمت های قبلی با استفاده از فرخوانی کتابخوانه و تابع مربوط به آن در فایل display.cpp مراجعه شده و تابع display که پارامتر اشاره گر نود را میگیرد انجام می شود.

نمایش مرحله ای درخت بدین صورت انجام می‌شود که از نود ریشه در مرحله یک شروع می کنیم و سپس نود های فرزند آن که در مرحله دوم و الی آخر وجود دارند را نمایش می‌دهیم.

این نمایش به صورت برگی برای نود والد صورت میگیرد و همانند پیمایش خطی درخت است با این تفاوت که علاوه بر پیمایش، عناصر نیز نمایش داده می‌شوند.

**توابع دیگر**

در فایل های برنامه مانند functions.cpp توابعی همانند تابع بررسی کننده نود روت (isRoot)، یافتن نود والد (findParent) و... یافت می‌شود که در توابع دیگر مورد استفاده قرار می‌گیرند.

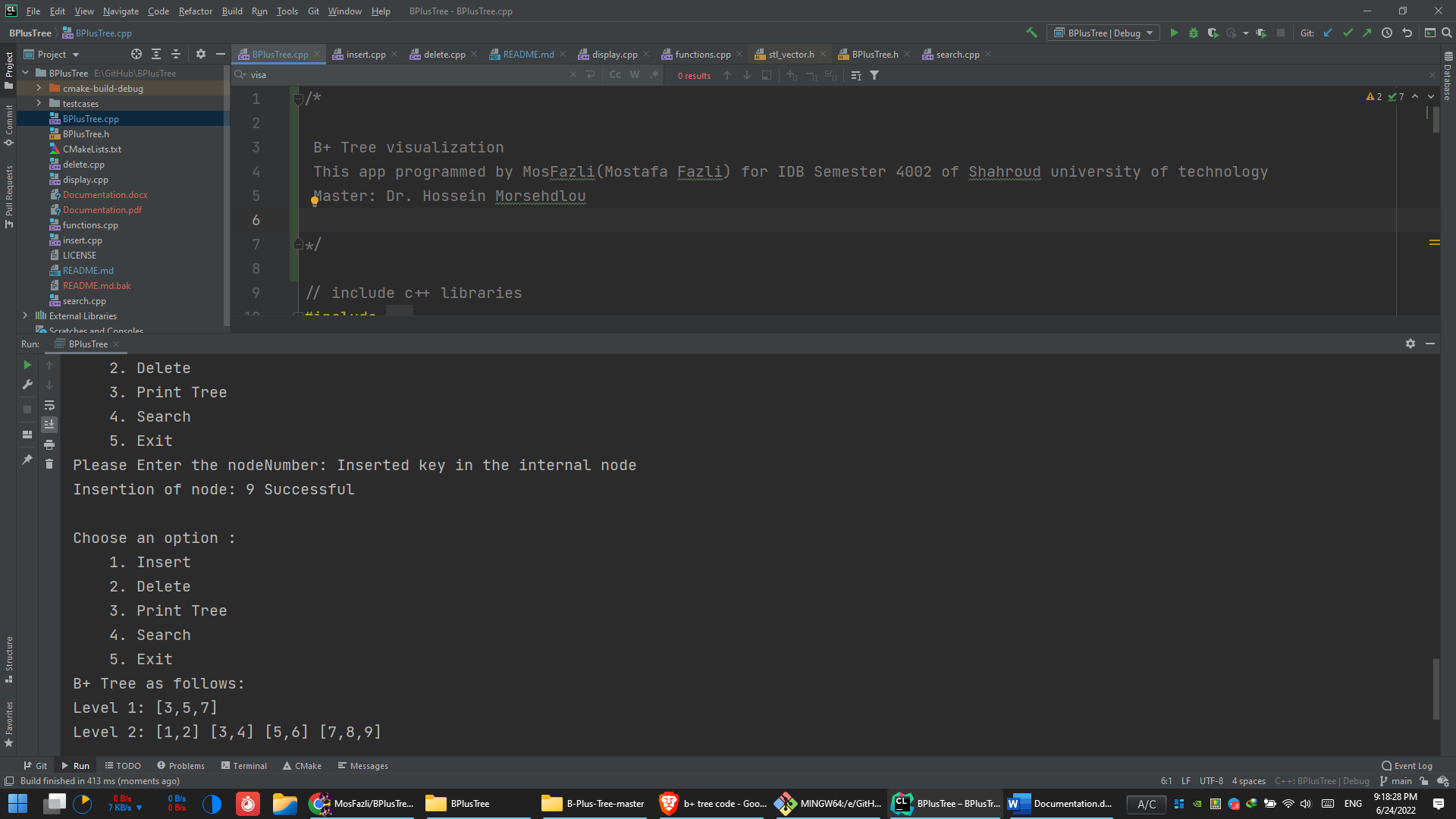
همگی این توابع به دلیل اشتراک استفاده در یک فایل جدا آمده‌اند و با استفاده از کتابخانه، دسترس‌پذیر شده‌اند.

فایل هایی متنی در پوشه testcases با نام های testcase1.txt … ایجاد شده اند که شما می توانید با استفاده از آن‌ها، طریقه مقدار دهی به برنامه را بررسی کرده و صحت عملکرد برنامه را مورد آزمایش قرار دهید.

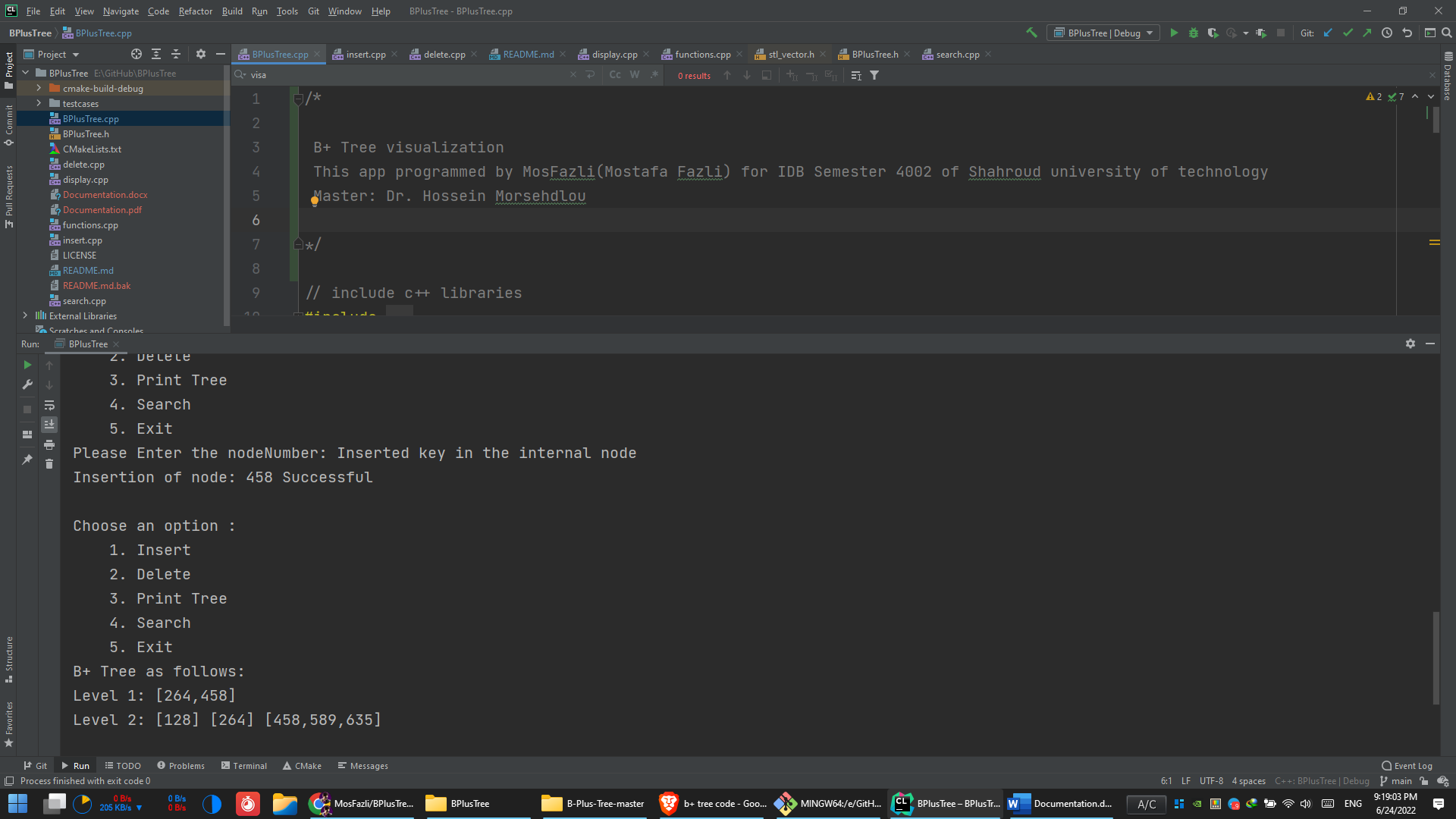
همچنین این برنامه در گیتهاب به آدرس <https://github.com/MosFazli/BPlusTree> آمده است و می‌توانید از آن به صورت رایگان استفاده کنید.

* این برنامه با استفاده از برنامه Clion از سری JetBrains کدنویسی شده است.

Test case 1 result:



Test case 2 result:



Test case 3 result:

