به نام خدا

محمد رضا جبلی
۹۴۳۱۰۳۵

شرح انجام پروژه:
پروژه از آبجکت های مختلفی تشکیل شده است که همه ی این ساختمان داده ها به جز ساختمان داده ی هش در یک مورد با هم مشترک هستند و آن درخت بودن است.
برای همین همه ی درخت ها ازیک اینترفیس درخت استفاده میکنند. هر درخت از یک روت تشکیل می شود که به نود مخصوص آن اشاره میکند. در هر درخت کلمات به نحوی قرار داده میشوند. بزرگترین تفاوت و شباهت درختها در این است که هر نود از ساختمان داده ی بی اس تی شامل یک کلمه میشود اما در ساختمان داده های تی اس تی و ترای هر نود شامل یک حرف میشود.
موردی که در الگوریتم اینورتد ایندکس نیز مورد تاکید قرار میگیرد این است که هر نود از این درختها اگر آخرین عضو کلمه ی خود باشند شامل یک لینک لیست از فایل هایی است که این کلمه در آنها وجود دارد.
در ساختمان داده ی هش از یک آرایه استفاده شده است که هر کلمه با یک هش کد به یک خانه از آرایه مربوط میشود و در صورت پر بودن آن خانه در یک لینک لیست قرار میگیرد. هر عضو از این لینک لیست خود یک زیرلیست دارد که به فایلهایی که کلمه در آنها وجود داشته اشاره میکند.

نحوه ی اجرای برنامه:
هر کدام از ساختمان داده های مطرح شده باید وظایف مشخصی را انجام دهند از جمله ی آنها ساختن ساختمان داده ، سرچ کردن در ساختمان داده ، دیلیت کردن یک فایل ، اضافه کردن یک فایل ، بروزرسانی و پیمایش و ...
در ضمن برای اضافه شدن هر کلمه ابتدا باید چک شود که این کلمه استاپ ورد نباشد یعنی جزو کلمات پرتکرار نباشد. برای این منظور یک فایل شامل کلمات پر استفاده تهیه شده است.

ساختن ساختمان داده: هربار یکی یکی کلمات را از داخل تک تک فایلها میخوانیم و طبق الگوریتم و مقایسه با نود های درخت به درخت اضافه میشود. برای متوازن شدن درخت ها لازم است که یک سری جابجایی نیز در درخت انجام شود تا ارتفاعه درخت همواره لگاریتم تعداد کلمات باشد.

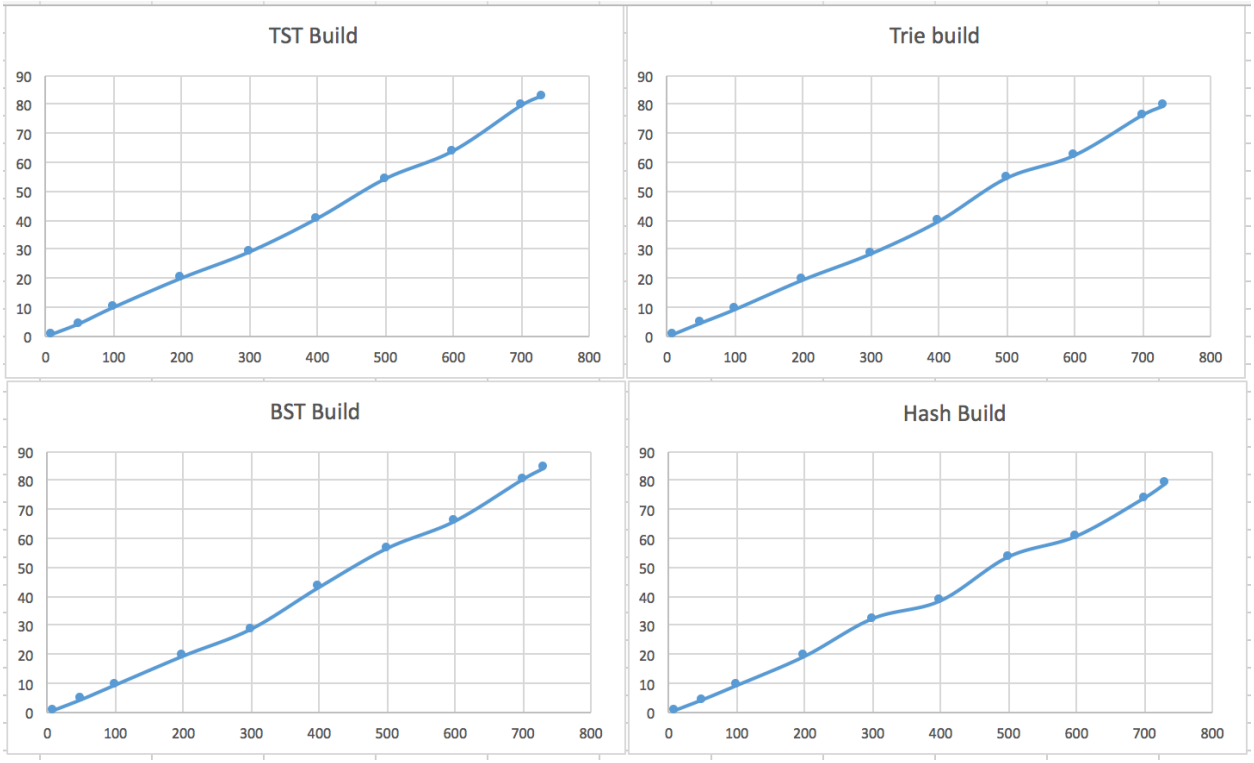
حذف از ساختمان داده: برای حذف از ساختمان داده نیز لازم است که خود نود نیز حذف شود . و در ضمن پس از آن درخت منوازن شود.

تحلیل ساختمان داده های مورد استفاده:

زمان بدست آمده بر اساس تعداد فایلها و ساختمان داده های مختلف

تعداد فایلها	tst	bst	trie	hash
10	0.643ms	0.714ms	0.645ms	0.665ms
50	4.305ms	4.367ms	4.476ms	4.289ms
100	9.874ms	9.403ms	9.147ms	9.191ms
200	19.934ms	19.400ms	19.326ms	19.277ms
300	28.961ms	28.706ms	28.328ms	32.264ms
400				
500	54.158ms	56.558ms	54.471ms	53.600ms
600				
700				
731	82.815ms	84.255ms	79.366ms	78.837ms

که نمودار بیلد آن شکل زیر میشود که محور عمودی آن زمان و محور افقی آن تعداد فایلها است.



مطابق جدول و نمودارها نشاندهنده ی این است که ساختمان داده ی هش بیلد را سریعتر انجام میدهد و اد کردن و دیلیت کردن در آن با تتای ۱ انجام میشود اما مشکلش این است که فضای زیادی را اختیار میکند. بعد از هش بیشترین سرعت را ساختمان داده ی ترای را دارد.

پروژه ی دوم (گراف):
از کاربر یک ماتریس مجاورت میگیریم و طبق آن گراف را میسازیم و همرا یا آن یال ها و راسهای آن را.
هر راس آرایه ای از یالهایی که از آن خارج میشود را دارد و هر یال راسی که به آن میرود را نگهداری میکند.
برای بررسی صحت یک رشته ی ورودی روی راس ها و یالها حرکت میکنیم و در آخر اگر به یک راس فاینال رسیدیم رشته را مقبول اعلام میکنیم.
برای تشخیص دور و حذف دور نیز از الگوریتم دی اف اس استفاده شده است و دو بولین کمکی برای هر راس در نظر میگیریم.
یک بولین که هر بار که یک راس ویزیت شد آن ترو میشود و دیگری هربار که تمام یالهای یک راس بررسی شدند ترو میشود.

نمونه ای از حذف دور از گراف: