تابع find :

به دلیل استفاده از دیکشنری در پایتون می توان هش هر فرد را در o(1) یافت . اما برا پیدا کردن خود node میتوان در o(nlogn)

تابع ADD :

در این تابع فرزند را به والد اضافه میکنیم که این هم o(1) می باشد

تابع SIZE :

به وسیله تابع LEN میتوان تعداد افراد را در o(1) یافت .

تابع DELETE :

ابتدا باید node مورد نظر را یافت ((o(nlogn) برای حذف کردن فرد به وسیله تابع remove استفاده میکنیم تا این فرد را از فرزندان والد آن حذف کند ((o(n)

تابع areSiblings :

برای چک کردن به o(1) نیاز داریم

تابع areParentAndChild :

برای چک کردن به o(1) نیاز داریم

تابع findCommonAncestor :

تابع بازگشتی حداکثر به اندازه ارتفاع درخت صدا زده میشود در نتیجه o(n)

تابع areRelated :

از تابع findCommonAncestor استفاده می کند در نتیجه o(n)

تابع findFarthestDescendant :

تابع بازگشتی حداکثر به اندازه ارتفاع درخت صدا زده میشود در نتیجه o(n)

تابع getLeaves :

پیمایش روی تمام node ها می باشد در نتیجه o(n)

تابع BFS :

o(n+m) : تعداد یال ها :m تعداد node ها : n

تابع findLongestPath :

به تعداد leaf ها از BFS استفاده می کنیم درنتیجه o(n^2 + nm)

تابع SHA256 :

با استفاده از توابع هش با طول رمز عمومی بزرگ امنیت بالایی فراهم می‌کند . این الگوریتم‌ها تحت تأثیر تحلیل‌های رمزگشایی قوی قرار می‌گیرند و نسبت به Md5 و sha1 امنیت بالاتری دارد چون احتمال برخورد در آن بسیار کمتر از دو روش قبل است. همچنین با توجه به توابع کمکی استفاده شده پیچیدگی زمانی آن o(n^2) میباشد

همچنین امروزه برای بیشتر کردن امنیت و پخش شدن یک سری هش های متداول به انتهای متنی که قرار است هش شود اصطلاحا salt اضافه میشود برای مثال فرض کنید هش یکی از پسورد های متداول را داریم به عنوان مثال 123456 . یک string رندوم به عنوان salt به انتهای آن اضافه میکنیم

Password = 123456
Salt = DataStructures
Password = Password + salt

و بعد پسورد را هش میکنیم که هم به امنیت آن می افزاید و هم از پخش شدن یک سری هش های متداول جلوگیری می کند