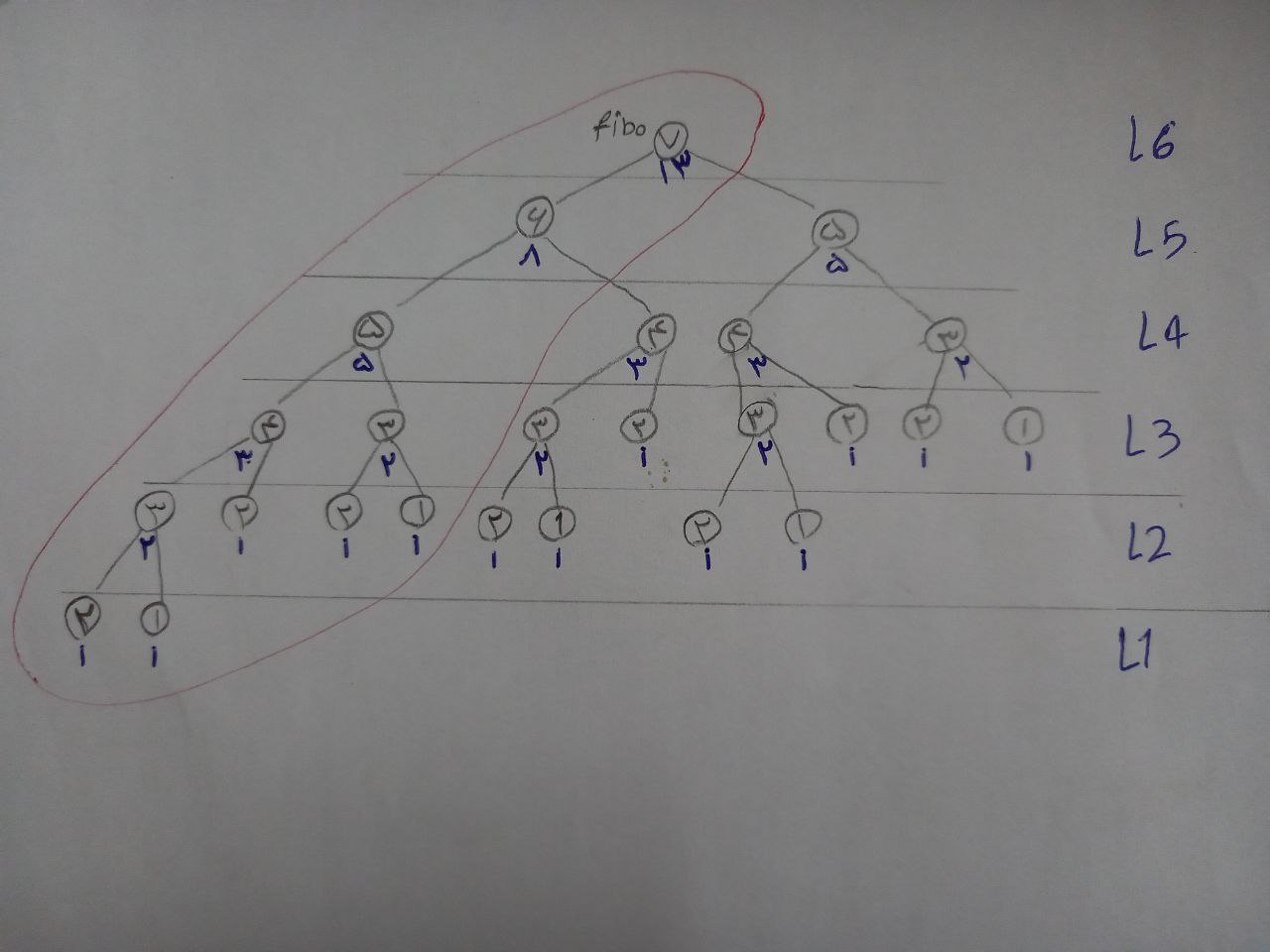
توضیحات برای تمرین اول:

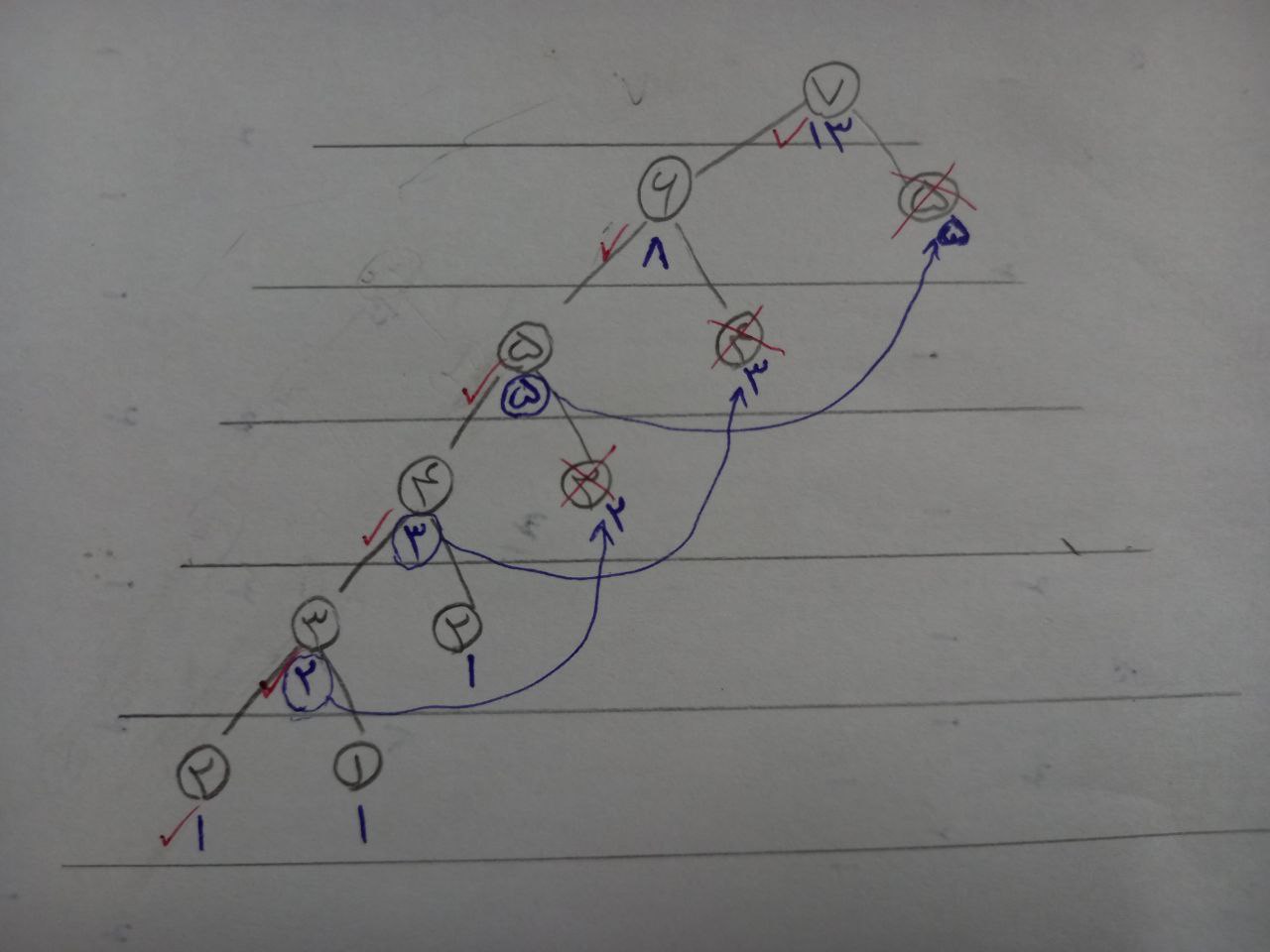
برای cache کردن از دو مرحله استفاده شده. یکی روی RAM به شکل دیکشنری در حال اجرای برنامه و یکی روی HDD به صورت فایل که در شروع برنامه در صورت وجود بارگیری می‌شود و به دیکشنری روی RAM اضافه میشود.

درصورتی که بخواهیم فیبوناچی را به روش بازگشتی بدون بهینه سازی و استفاده از cache انجام دهیم، به عنوان مثال برای fibo(7) برنامه درخت زیر را به شکل کامل محاسبه خواهد کرد تا به نتیجه برسد. در صورتی که برای محاسبه fibo(7) تنها بخشی که با محدوده قرمز مشخص شده است کفایت میکند. این به دلیل بازگشتی بودن تابع مورد نظر است که موارد تکراری را دوباره حساب میکند.



در مرحله اول این مشکل را با cache کردن مراحلی که طی شده در یک دیکشنری حل میکنیم تا از تکرار جلوگیری کنیم. تا همین مرحله سرعت به شکل قابل ملاحظه ای افزایش پیدا خواهد کرد. مخصوصاً برای اعدا بزرگ. با این حساب کل عملیات به شکل یک درخت تک شاخه انجام خواهد شد. که تصویر آن در ادامه آمده است.

برای بهینه‌تر شدن عملیات cache یک فایل برای نگه داری مقادیر فیبوناچی قبلی، روی HDD در نظر گرفتم. به این صورت که ابتدا درون فایل به دنبال آخرین فیبوناچی بدست آمده میگردد و آن را به دیکشنری cache اضافه میکند. برای مثال اگر یکبار fibo(10) را حساب کرده باشیم حالا مقادیر فیبوناچی ۱ تا ۱۰ روی فایل ذخیره شده است. اگر بخواهیم fibo(15) را حساب کنیم درخت تک شاخه ما فقط از فیبوناچی ۱۵ تا ۱۱ را حساب خواهد کرد و مقدار فیبوناچی ۱۰ را از روی فایل بدست خواهد آورد که باعث می‌شود تکرار های درخت تک شاخه هم کاهش پیدا کند.



تعداد محاسبات بازگشتی بدون اعمال این روش از فرمول زیر قابل محاسبه است

number\_of\_recursion = ( 1 / sqrt(5) ) \* 1.618 ^ ( N+1 )

تعداد محاسبات بازگشتی که در این روش انجام می‌شود از فرمول زیر قابل محاسبه است.

number\_of\_recursion = 2 \* ( fibo\_number – last\_cache\_number) + 1

اگر فایل cache خالی باشد و بخواهیم fibo(20) را بدست بیاوریم.

* بدون بهینه سازی:

fibo(20) = ( 1 / sqrt(5) ) \* 1.618 ^ (20 + 1) = 10940

* بعد از بهینه سازی (فقط cache روی RAM)

Number\_of\_recursion = 2 \* (20 – 0) + 1 = 41

اگر فایل cache مقدار فیبوناچی 20 را داشته باشد و ما بخواهیم fibo(35) را بدست بیاوریم.

* بدونه بهینه سازی:

fibo(35) = ( 1 / sqrt(5) ) \* 1.618 ^ (35 + 1) = 14918

بعد از بهینه سازی ( استفاده از cache روی HDD و RAM)

number\_of\_recursion = 2 \* (35 – 20) + 1 = 31

کاملاً مشخص است که عملیات های بازگشتی به تعداد قابل توجهی کاهش پیدا میکند. درمورد اجرای برنامه و مقایسه زمان ها روز اراعه بیشتر توضیح میدم.