A kódban a rekurzív függvényhívás a következőképpen történik:

int fibonacchi(int a){

if(a == 0) return 0;

if(a == 1) return 1;

int f1 = fibonacchi(a-2);

int f2 = fibonacchi(a-1);

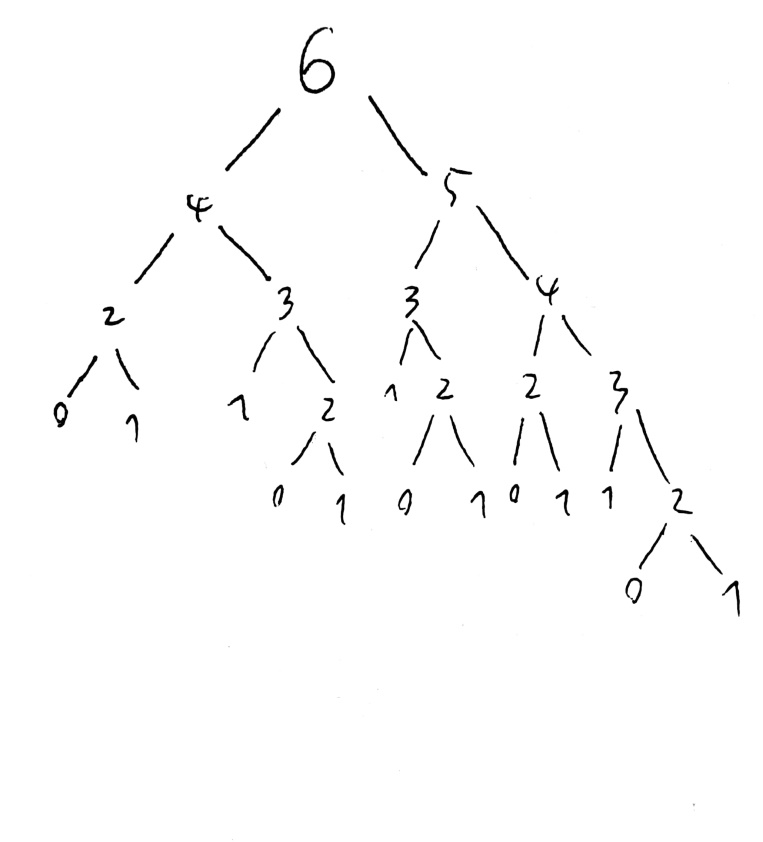
return f1+f2;

}

Ahol „fibonacci(a)” az a. Fibonacci számot jelenti.

Érdemes megfigyelni, hogy mindig először az (a-2). Fibonacci számot számolja ki a program. Ez azt fogja eredményezni, hogy egészen addig fog kettesével lefelé lépkedni a függvény, amíg el nem ér az egyhez vagy a nullához. Ha ez megtörténi, akkor egyszer a program megpróbálja az (a-1). is kiszámolni, és ha ez nem sikerül neki egy lépésben (tehát a-1 != 1), akkor újra az (a-2).-al fog próbálkozni.

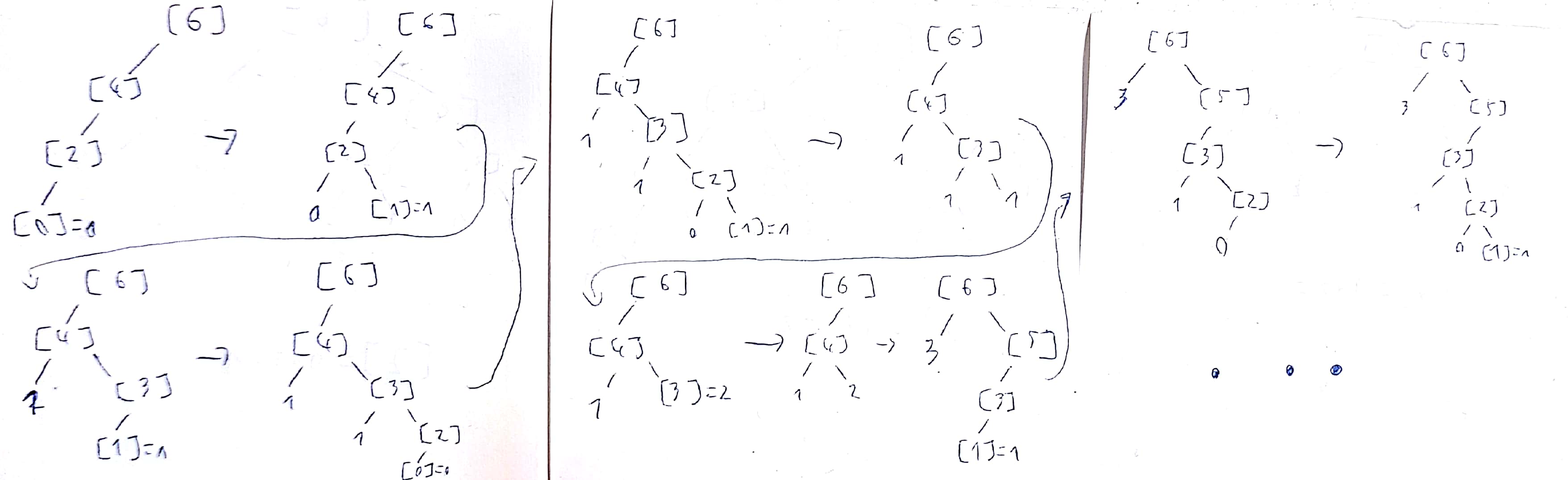
Ha ez nehezen érthető, semmi gond. Egyszerűbben úgy lehet ezt ábrázolni, hogy a program egy fát fog bejárni.



A balra lépés az (a-2). szám kiszámolását jelenti, a jobbra lépés pedig az (a-1).-t. A képen a 6. F. szám kiszámolásához szükséges fa látható.

A rekurzió természetéből adódóan hibás azt feltételezni, hogy mint ahogy azt egy ember csinálná, a program „lentről fölfelé” módszerrel fogja megoldani a feladatot!

Mivel a program először mindig az (a-2)-t próbálja kiszámolni, ezért egészen addig fog a program „balra lépni”, amíg már kénytelen nem lesz jobbra lépni.



Sajnost ennél nagyobbra nem tudtam kinagyítani, de a Driveban megtaláljátok majd külön is.

Ami a fabejárásokat illeti, eleve kis problémát jelent, hogy a fában nem látjuk az egyeseket és a nullákat. (mivel „a” ezeket az értékeket soha nem veszi fel, ezeket csak visszaadja a függvény konstansként.)

Attól függően, hogy mikor íratjuk ki az „a” változót, más fabejárásokat fogunk kapni.

(Érdemes észrevenni, hogy az algoritmus nem optimális, hoszen rengetegszer kiszámolja a program pl. a 3. és a 2. számot.)

