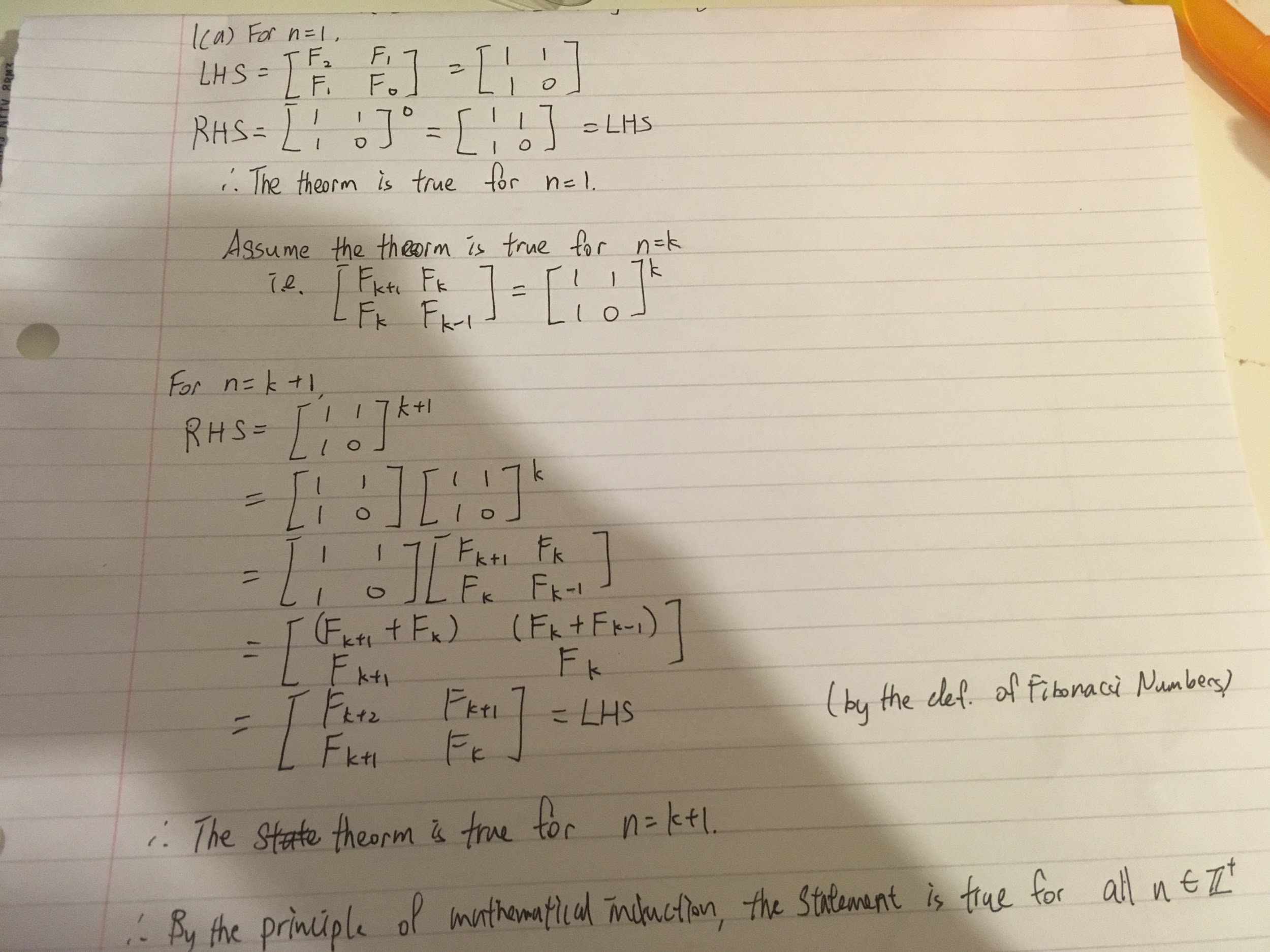
1(a)(i) 

1(a)(ii)

f(x, n) = 1 for n = 0

f(x, n) = x for n = 1

if (n is even)

y = f(x, n / 2)

f(x, n) = y \* y

else

y = f(x, (n - 1) / 2)

f(x, n) = y \* y \* x

1(b)(i)

T(n) = 2T(n/2 + 17) + n Guess O(n lg n)

Prove T(n) <= cn lg n - dn

T(n) ≤ 2(c(n/2 + 17) lg (n/2 + 17) - dn) + n

= 2c(n/2 + 17) lg (n/2 + 17) - 2dn + n

= c(n + 34) lg (n/2 + 17) - 2dn + n

= c(n + 34) lg (n/2 + 17) + c(n + 34) lg 2 - c(n + 34) lg 2 - 2dn + n

= c(n + 34) lg ((n/2 + 17) \* 2) - c(n + 34) lg 2 - 2dn + n

= c(n + 34) lg (n + 34) - c(n + 34) lg 2 - 2dn + n

≤ 2cn lg 2n - 2cn -2dn + n (for n ≥ 34)

= 2cn lg2 + 2cn lg n - 2cn - 2dn + n

= 2cn + 2cn lg n - 2cn - 2dn + n

= 2cn lg n - 2dn + n

≤ cn lg n (for 2d ≥ 1)

----------------------

Alternative version (ignoring the +17)

T(n) = 2T(n/2) + n

Need to prove T(n) <= cnlgn - bn

Base case:

T(2) = 2T(1) + 2 = 4

<= 2clg2 = 2c for c>=2

Inductive case:

Assume T(n/2) <= cn/2 lg n/2 - bn/2

T(n) <= 2c n/2 lg n/2 - bn/2 + n

= cn lg n - cnlg2 - bn/2 + n

<= cn lg n for b>=2

1(b)(ii) T(n) = 4T(n - 2) + 1 Guess O(4n)

Prove T(n) ≤ c4n

T(n) ≤ 4(c4n-2) + 1

= c4n-1 + 1

≤ c4n

1(c)(i) logb a = log2 2 = 1 d = 0

logb a > d

Θ(n)

1(c)(ii) logb a = log4/5 1 = 0 d = 0

logb a = d

Θ(lg n)

1(c)(iii) logb a = log16 4 = 0.5 d = 0.5

logb a = d

Θ(√n lg n)

1(c)(iv) logb a = log2 4 = 2 d = 3

logb a < d

Θ(n3)

1(d)(i) (=, according to lecture notes)

1(d)(ii)

(note: D&C slide 14 states that "the cost of divide and combine is given by summing the operations for each recursion level" and does not include the theta part)

2(a)(i) Consider case

|  |  |  |  |
| --- | --- | --- | --- |
| Activity | A | B | C |
| Starting time | 0 | 2 | 3 |
| Finishing time | 3 | 4 | 6 |
| Duration | 3 | 2 | 3 |

The greedy algorithm that select activity with least duration will ends up with {B}

while {A, C} should be the optimal solution.

2(a)(ii) Consider case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Activity | A | B | C | D |
| Starting time | 0 | 1 | 2 | 4 |
| FInishing time | 5 | 2 | 3 | 6 |

The greedy algorithm that select activity with earliest starting time will end up with {A}

while {B, C, D} should be the optimal solution.

2(b)(i)

**def** max\_apples(grid, m, n):

*# Desroys input, make fresh grid if this isn't allowed.*

**for** i **in** range(m):

**for** j **in** range(n):

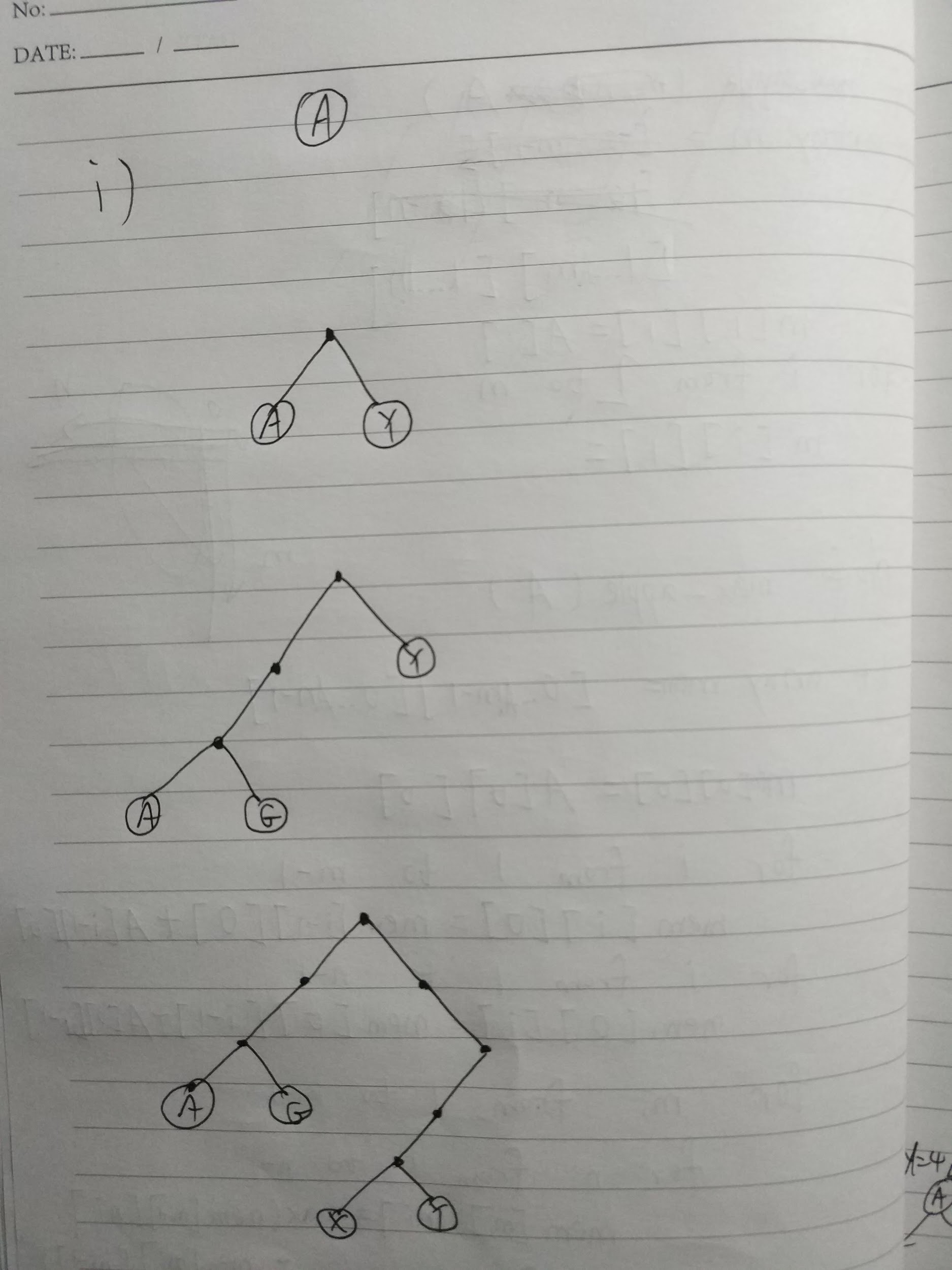
cell\_above = grid[i-1][j] **if** i > 0 **else** 0

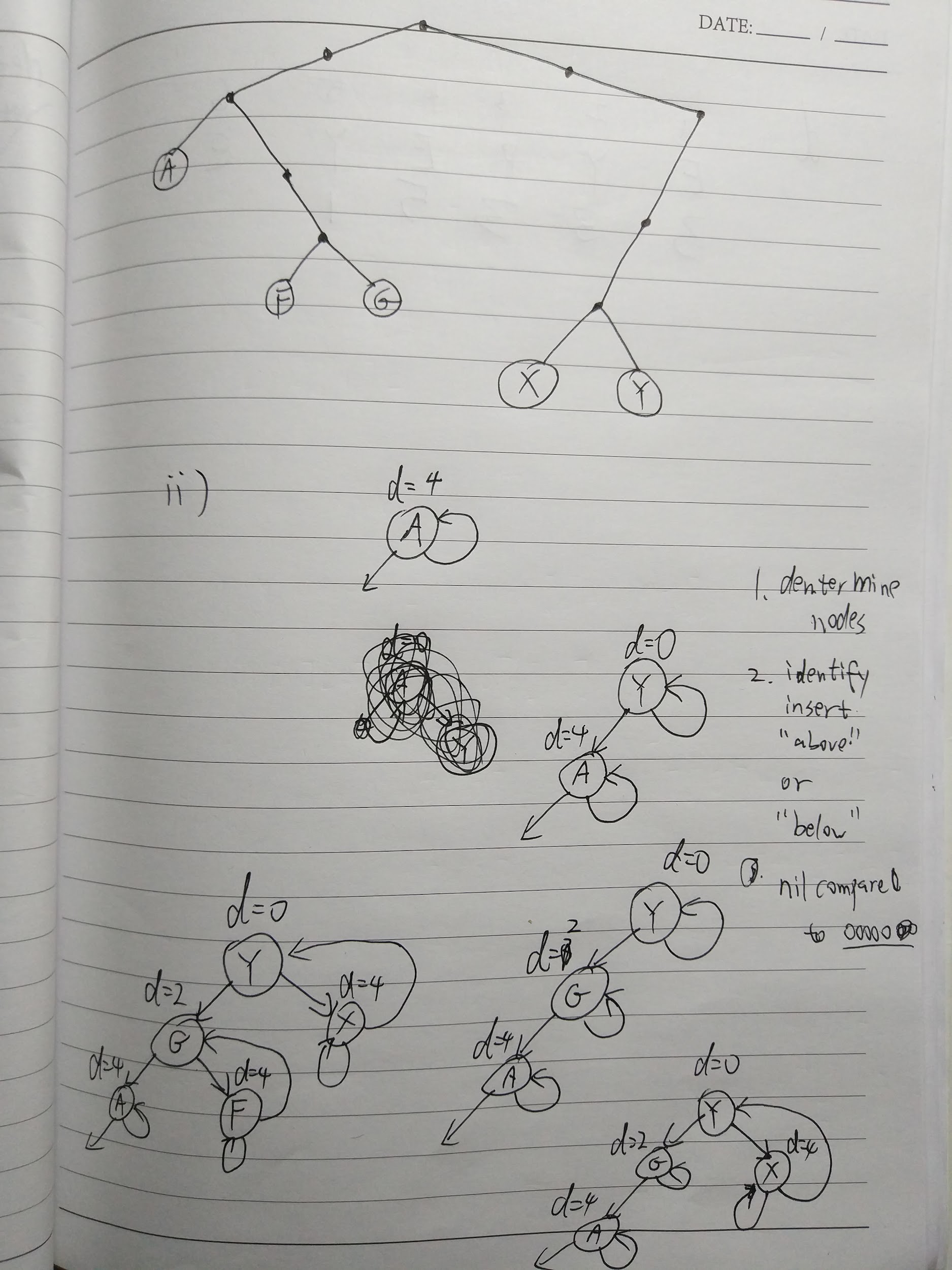
cell\_left = grid[i][j-1] **if** j > 0 **else** 0

grid[i][j] += max(cell\_above, cell\_left)

**return** grid[m-1][n-1] **if** grid **else** 0

2(b)(ii) O(mn)





t

2(d)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 | 5 | 6 |
| P[i] | E | Y | P | E | Y | ε |
| gsr[i] | 3 | 3 | 3 | 5 | 1 |  |

-I love that sudden activity a day before the exam!

-When else?

-On the exam day of course!