Problem 3

Just use recursion (No DFS or graphs)

Similar to Shir Case problem in the midterm.

$n = 0 \Rightarrow$

$n = 1$ f is the function.

$n = 2$

Then find the recurrence-relation

Use the recurrence to find the correct for $f(0)$.

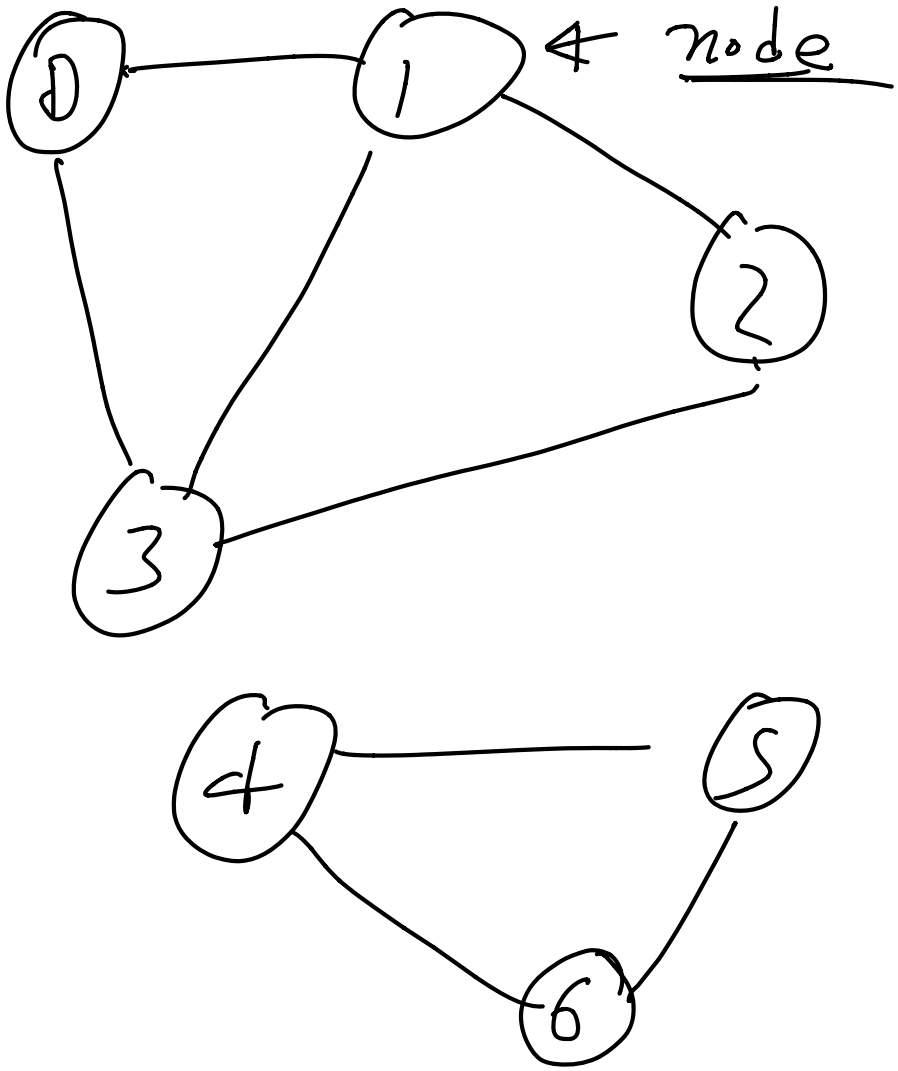
$f(2)$ will be related to $f(0)$ and $f(1)$

by the recurrence relation is.

$f(0) =$

$f(1) =$

Problem 1



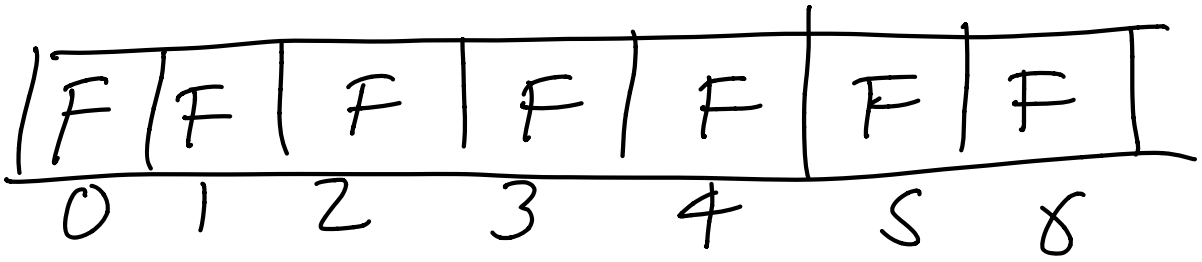
Step 1: Find all the nodes connected to (1) — input node.

Use dfs!

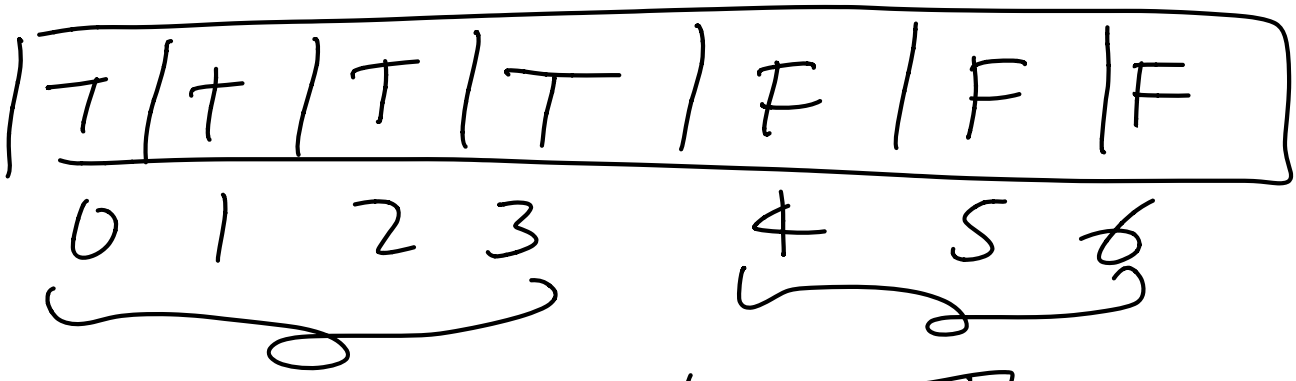
All nodes connected to (1) will have visited value true.

All node not connected to (1) will have visited value false

visited initially



After dfs!



These are connected to 1

These are not connected to 1.

Now search using recursion find the max-value among the vertices which has been visited.

visited

T	F	T	T	F	T	T
0	1	2	3	4	5	6

T	T	T	T	T
0	1	2	3	4

```
(: richest-connected : Graph
   (Vectorof Exact-Rational) Vertex
                             Vertex)
(define (richest-connection g money v)
 (local
   {
    (: n : Integer)
    (define n (Graph-ng g))

    (: visited : (Vectorof Boolean))
    (define visited (make-vector n #f))

    (: find-max-helper : Integer Integer
                         Integer -> Integer)
    (define (find-max-helper curr-max i)
```

max-val

startindx

index of the max seen till now

index of the max current index

```
(cond
  [(= i n) curr-max]
  [else (find-max-helper
```

(write a if statement to return a new curr-max (i+1))]))).

max

```
(begin
  (dfs1 g v visited)
  (find-max-helper v 0))).
```

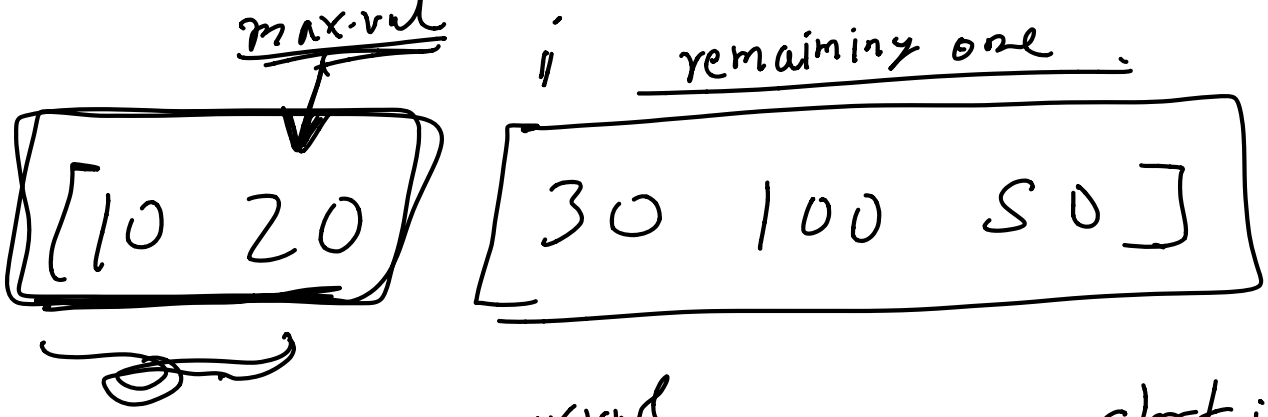
curr-max is index of the max-value seen till now

money ↓

[10 20 30 100 50] ✓

[T T F T F]

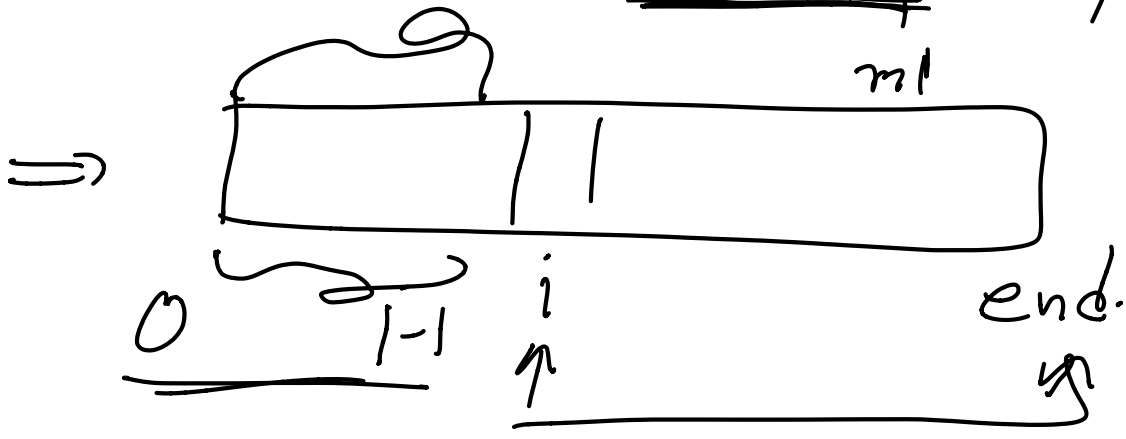
visited



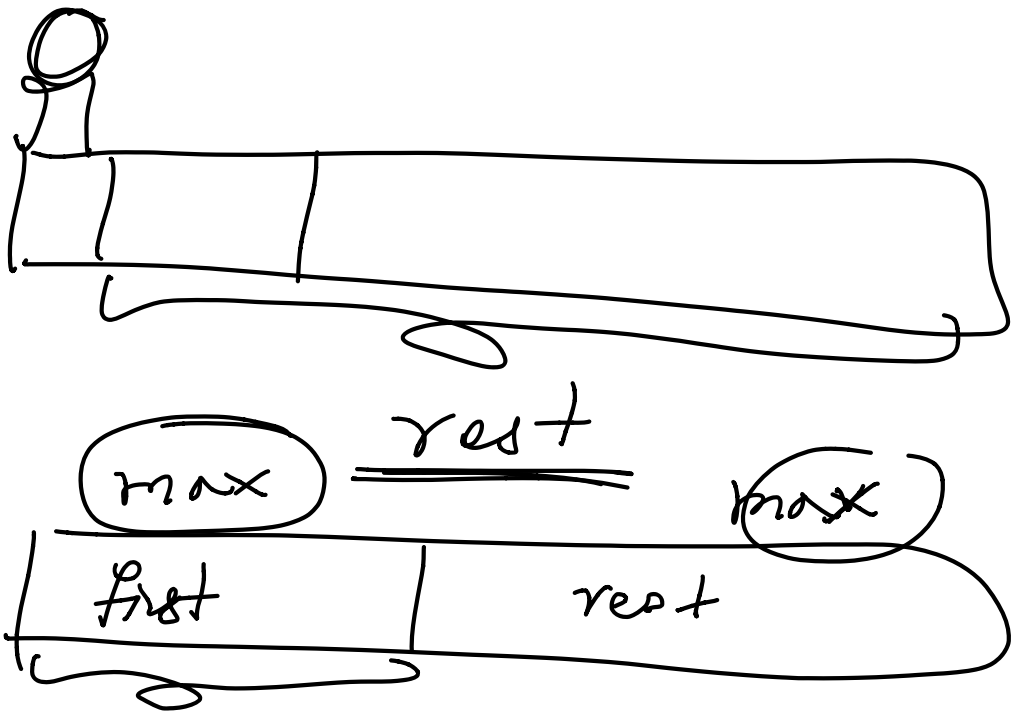
curr-max val

start index

(find-max-helper curr-max i)



max(m1, curr-max) ⇒



curr-max = 1

i = 0

10	20	30	100	50

T	T	F	T	F

curr-max = 10

V = 1

i = 0 ⟹ 10$

curr-max ⟹ 20$

You will not update the curr-max

Curr-max = 1 → curr-max = 3

I = 3

(vector-ref money i)

> (vector-ref marray curr-max)
→ if duke tk curr-max → i
else use tk old
curr-max

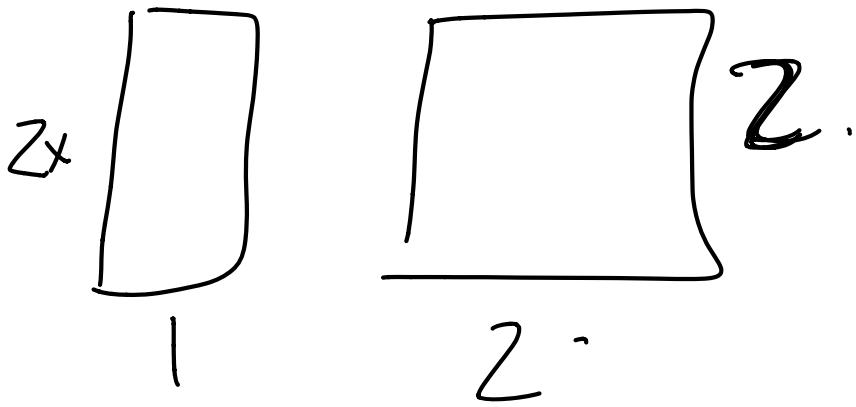
cond (vector-ref visited i)

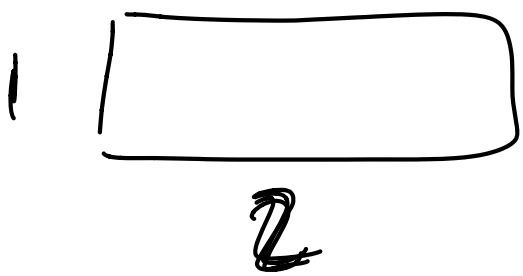
;; 0 → "A"
;; 1 → "B"
;; 2 → "C"

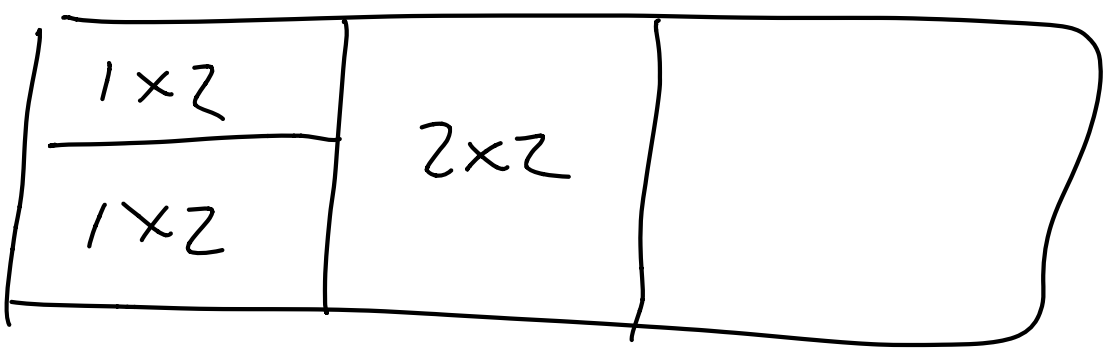
3. Problems

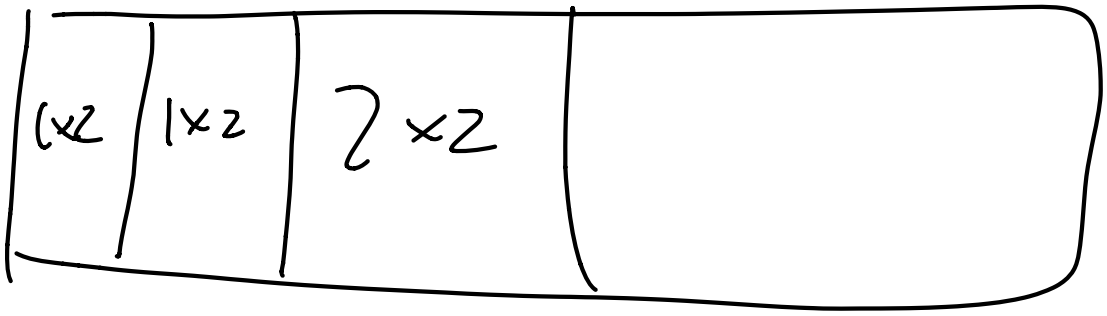
Let my function be f(n)
= number of tilings of 2×n
floor

3. (5 points)









Strategy

Consider all possible tilings of a floor of size $2 \times n$

S

We want the number of elements in S, $\#S$.

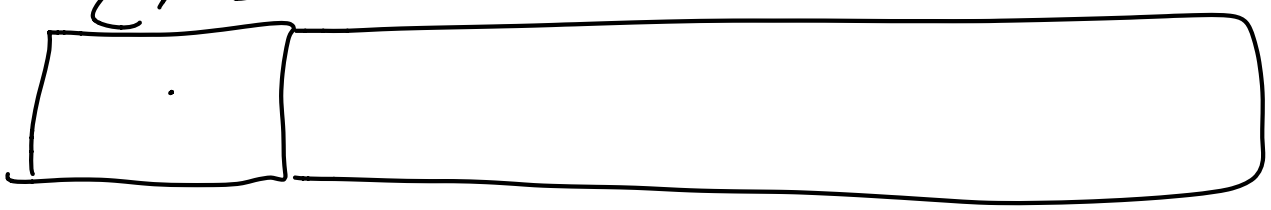
Divide S in to disjoints-sets.

$$S = S_1 \cup S_2 \cup S_3 \cup S_n.$$

$$\#S = \#S_1 + \#S_2 + \#S_3 + \dots$$
$$= \quad = \quad \#S_n$$

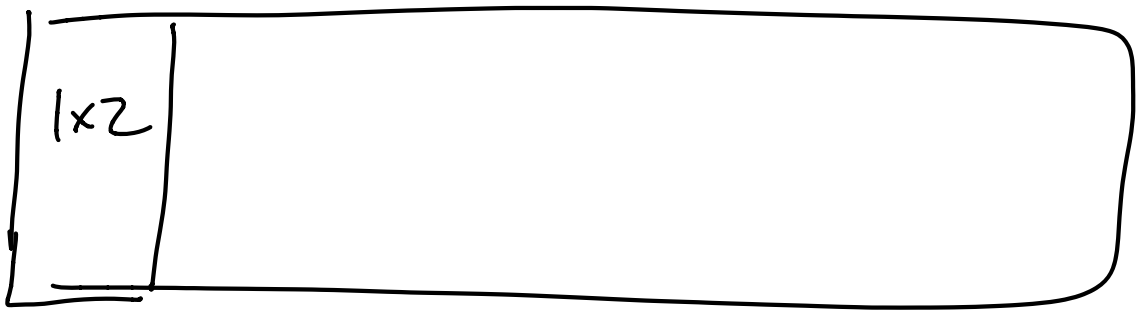
Consider how a tiling can begin

$S_1 =$



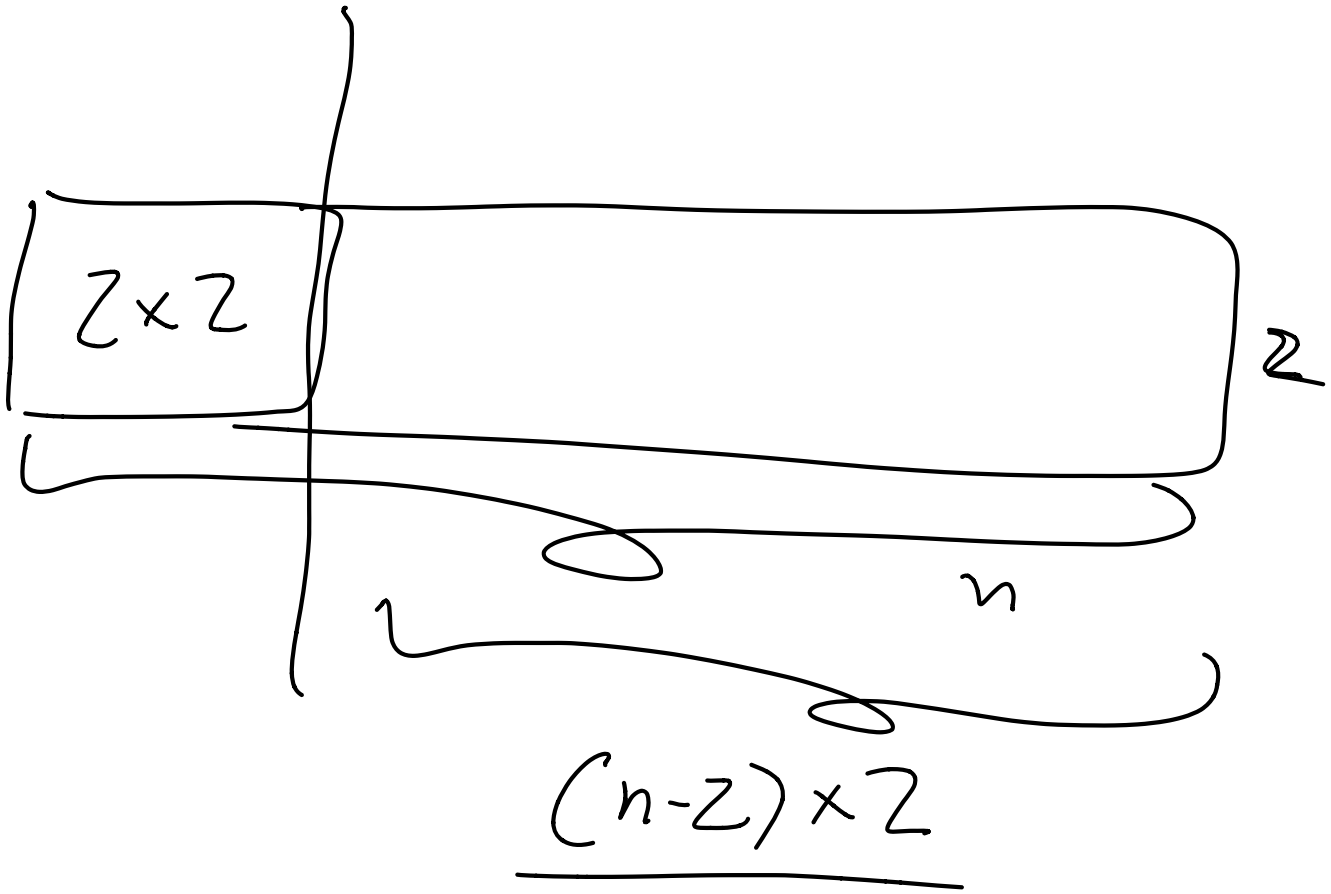
S_2

S_3



$$S = S_1 \cup S_2 \cup S_3$$

H S$_1$: Number of filling of $2 \times n$ floor which stores 2×2 tile



Remove thejay is filing of $2 \times (n-2)$

There is a one-one correspondence between tilings of $2 \times n$ floor which start with 2×2 and tilings of $2 \times (n-2)$

$$\boxed{\#S_1 = f(n-2)}$$

$$\#S = \#S_1 + \#S_2 + \#S_3$$

$$= f(n-2) + \#S_2 + \#S_3$$

Now compute $\#S_2$ and $\#S_3$.

$$f(n) = f(n-2) + \#S_2 + \#S_3$$