Exercise 1

P(n): "any post. of n cents can. b. formed unsing just 3 and 5cents", $n \geq 8$.

Basis Step

$P(8)$: $1 \cdot \boxed{5} + 3 \cdot \boxed{1}$
$P(9)$: $3 \cdot \boxed{3}$
(10) : $2 \cdot \boxed{5}$

Inductive Step

We assume $P(j)$ true, $8 \leq j \leq k$.
and we want to show that
$P(k+1)$ is true.

We know that $P(k-2)$ holds.
We can just add a $\boxed{3}$ to the existing combination.

Conclusion

$P(n)$ holds for $P(j)$, $12 \leq j < k$

Exercise 2

$P(n)$: " $f_1^2 + f_2^2 + \dots + f_n^2 = f_n f_{n+1}$ ", $n \in \mathbb{N}$.

Basis Step

$f_0 = 0, \ f_1 = 1, \ f_2 = 1, \ f_3 = 2$

$f_1^2 + f_2^2 = 2 = 1 \cdot 2$. $\boxed{\text{Ok}}$

Inductive Step

We assume $P(j)$ is true $1 \leq j \leq k$.

$P(k+1)$?

$$f_1^2 + f_2^2 + \dots + f_n^2 + f_{n+1}^2 \overset{?}{=} f_{n+1} f_{n+2}$$

$$\Leftrightarrow f_n f_{n+1} + f_{n+1}^2 = f_{n+1} f_{n+2}$$

$$\Leftrightarrow \cancel{f_{n+1}}(f_n + f_{n+1}) = \cancel{f_{n+1}}(f_{n+2})$$

$$\Leftrightarrow f_n + f_{n+1} = f_{n+2}$$

True by definition of the F. seq.

Conclusion

$P(1)$ is true and we showed that $P(k) \Rightarrow P(k+1)$, therefore $P(n)$ is true $\forall n \in \mathbb{N}$.

on a besoin de strong induction?

Exercise 3

does not work because :

$|T_2| \leq |T|$ (equal if $k = 0$)

Exercise 4

does not work (basis step
+ 3)

Exercise 5

(1)

```
procedure compute(n, x):
    if n = 1:
        return x
    else:
        n = n - 1
        return x + compute(n, x)
```

(2) Basis step

if $n = 0$: $\text{compute}(0, x)$

$= \underline{0}$.

Inductive Step

$P(k)$: compute $(k, x) = kx$. holds.

$P(k+1)$: compute$(k+1, x) = (k+1) \cdot x$?

$$= \text{compute}(k, x) + x$$
$$= \underline{x(k+1)}.$$

Conclusion

$P(1)$ holds and $P(k) \to P(k+1)$,

therefore $\forall n \in \mathbb{N}$, $P(n)$ holds.

Exercice 6

① • $(0101)^R$

$= 1010$

• $(1\ 1011)^R$

$= (11011)$

• $(1\ 0000\ 1001\ 0111)^R$

$= (1110\ 1001\ 0000\ 1)$

②
```
procedure reversal(a_1, a_2 ... , a_n):
    if (not a_2)
        return a_1
    return a_n + reversal(a_1, ..., a_{n-1})
```

Recursive definition of $R(w)$, the reversal of the string w:

- Basis Step: $R(\lambda) = \lambda$
- Recursive Step:
  - $R(w_1 s) = s R(w_1)$

③ $(w_1 w_2)^R = w_2^R w_1^R$

$\left[P(w_1 w_2) : (w_1 w_2)^R = w_2^R w_1^R. \right.$

Basis Step

$P(w_1 \lambda) : (w_1 \lambda)^R = (\lambda^R w_1^R)$

$P(\lambda)$ is true.

$= w_1^R$

Inductive Step

Suppose $(w_1 w_2)^R = w_2^R w_1^R$

Is $(w_1 w_2 w_3)^R = w_3^R w_2^R w_1^R$?

$w_1 w_2 w_3 = w_1 w_2 w_3$

Say $w_2 w_3 = w_4$

$$(w_1 w_4)^R = w_4^R w_1^R$$

$$= w_3^R w_2^R w_1^R \quad \boxed{\text{OK}}$$

$$w_4^R = w_3^R w_2^R$$

Exercise 7

(1) $a_0 = 1 \quad a_2 = 2$

$a_1 = 2 \quad a_3 = 4$

$a_n = a_{n-1} \cdot a_{n-2}$

```
procedure get_el(n):
    a = (1,2)
    for i=0; i < n; i = i+1:
        a.add(a[i+1] · a[i])
    return a[n]
```

procedure get-el(a_0, a_1, n, i)

if $i = n$:

return a_n

$i = i + 1$

return get_el($a_1, \overbrace{a_0 \cdot a_1}^{a_2}, n, i$)

$$f_n = f_{n-1} \cdot f_{n-2}$$

$$f_0 = 1$$

$$f_1 = 2$$

Exercise 8

$$L_n = L_{n-1} + L_{n-2}$$

$$\begin{cases} L_0 = 2 \\ L_1 = 1 \end{cases} \qquad f_n = f_{n-1} + f_{n-2}$$

$$L_n = f_{n-1} + f_{n+1}$$

Basis Step

$L_0 = 2$ $\qquad f_0 = 0$

$L_1 = 1$ $\qquad f_1 = 1$

$f_2 = 1$

$$L_1 = f_0 + f_2$$
$$= 1$$

<u>Induction Step</u>

$P(k-1)$ is true $\quad L_{k-1} = f_{k-2} + f_k$

$P(k)$ is true. $\quad L_k = f_{k-1} + f_{k+1}$

$P(k+1)$?

$$f_n = f_{n-1} + f_{n-2}$$
$$f_{n+2} = f_{n+1} + f_n$$
$$f_{n-2} = f_n - f_{n-1}$$

$$\boxed{L_{k+1} = f_k + f_{k+2}}$$

$\Leftrightarrow \quad L_k + L_{k-1} = L_{k+1}$

$\Leftrightarrow \quad f_{k-1} + f_{k+1} + L_{k-1} = L_{k+1}$

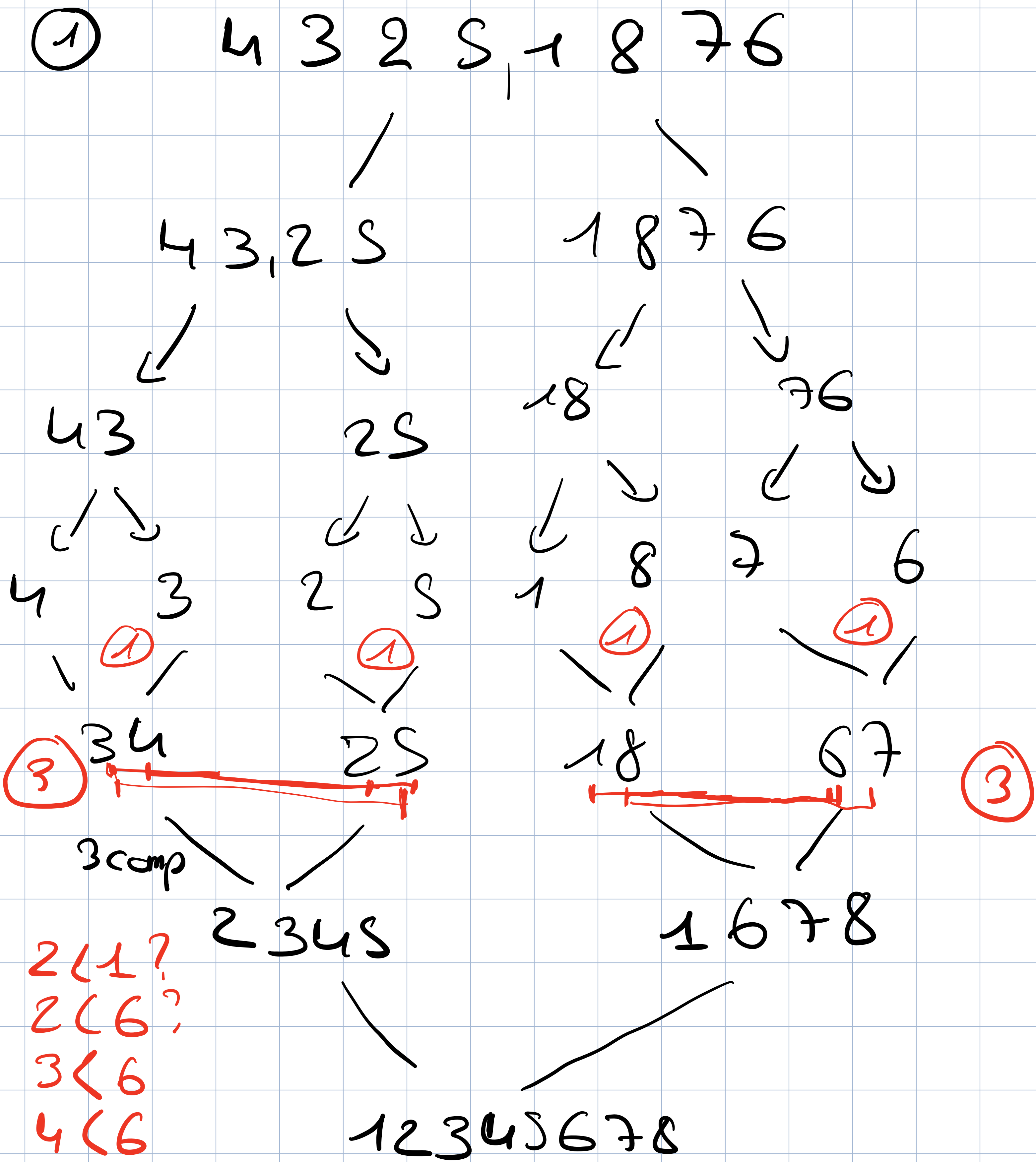
$\Leftrightarrow \quad f_{k-1} + f_{k+1} + f_{k-2} + \cancel{f_k} = \cancel{f_k} + f_{k+2}$

$\Leftrightarrow \quad f_{k-1} + \cancel{f_{k+1}} + \cancel{f_k} - f_{k-1} = \cancel{f_{k+1}} + \cancel{f_k}$

e) $\delta_{k-1} - \delta_{k-1} = 0$

$\boxed{0_k}$

Exercise 9



2<1?
2<6?
3<6
4<6

$5 < 6$

$$5 + 6 + 4 = \underline{15}$$

Ex 10

5 3 4 1 7 6

3415 76

13 45 6 7

13 45 6 7

```
procedure quick_sort(a1, a2 ... an):

    small_list = []
    greater_list = []
    for (element in [a2..., an]):
        if (element ≤ a2):
            small_list.add(element)
        else:
            grecter_list.add(element)

    small_list.add(a1)

    new_small_list = quick_sort(small_list)
    new_grater_list = quick_sort(greder_list)

    return concat(new_small_list,
                  new, greater_list)
```