mercoledì 27 novembre 2019 11:14

PRIMA PARTE ALLA LAVAGNA

```
let rec rev l =
        match l with
    [] -> []
  | x::xs -> rev xs @ [x];;
```

```
rev : 'a list -> 'a list = <fun>
```

'a list → tipo l; 'a list → Tipo ris

```
let rec rev_a l a =
        match l with
    [] -> a
  | x::xs -> rev_a xs (x::a);;
```

```
rev_a : 'a list -> 'a list -> 'a list = <fun>
```

'a list → tipo l; 'a list → tipo a; 'a list = <fun> → Tipo ris

```
let rev l =
    let rec rev_a l a = ...
    in rev_a l [];;
```

```
rev : 'a list -> 'a list = <fun>
```

rev : 'a list → 'a list = <fun>

Ricerca del massimo e minimo di una lista.

risultato potrebbe essere una coppia

(min, max)

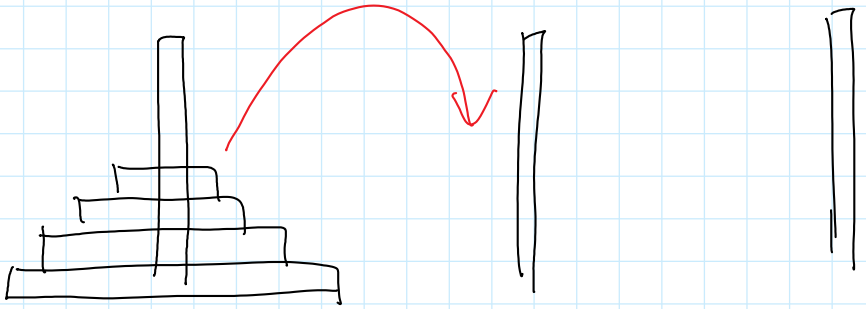
```
let rec minmax l =
    match l with
    [x] -> (x, x)
  | x::y::ys -> let (m1, m2) = minmax (y::ys)
                in
                if x < m1 then (x, m2)
                else if x > m2 then (m1, x)
                else (m1, m2) ;;
```

minmax : 'a list -> 'a * 'a = <fun>

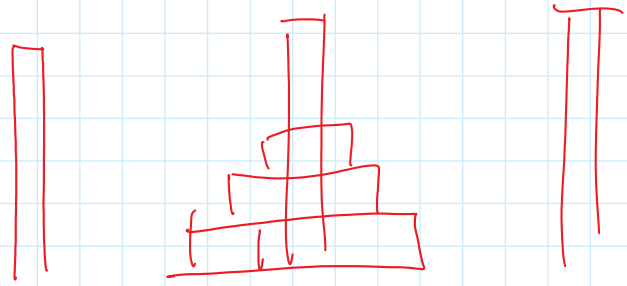
```
let rec minmax_a l (m1, m2) =
    [x] -> if x < m1 then (x, m2)
           else if x > m2 then (m1, x)
                else (m1, m2)
  | x::y::ys -> if x < m1
                then
                minmax_a (y::ys) (x, m2)
                else
                if x > m2
                then
                minmax_a (y::ys) (m1, x)
                else
                minmax_a (y::ys) (m1, m2)
;;
minmax : 'a list -> 'a * 'a -> 'a * 'a = <fun>
```

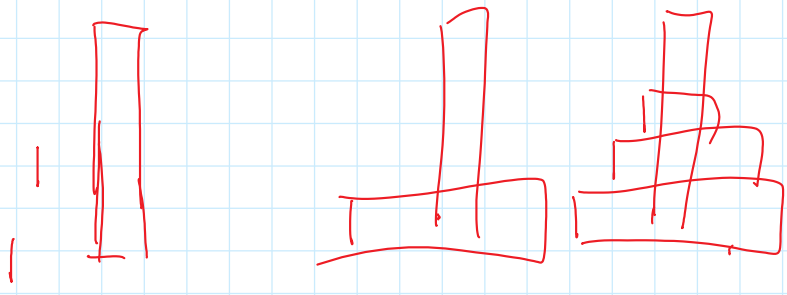
```
let minmax l =
    let rec minmax_a l (m1, m2) = ...
    in minmax l (hd l, hd l);;
```

torre di Hanoi



Un disco più grande non può stare su uno più piccolo





n - 1

n - 1

let rec hanoi n (p1, p2, p3) =

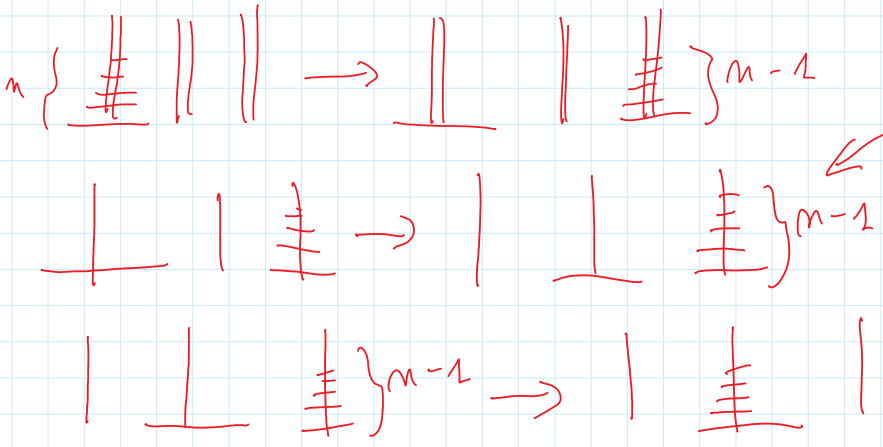
deve spostare n dischi da p1 a p2
utilizzando p3 come ausiliario

match n with

1 -> [(p1, p2)]

| n when n > 1 ->

hanoi (n-1) (p1, p3, p2) @ [(p1, p2)] @
hanoi (n-1) (p3, p2, p1);;



```
# hanoi 3 (1,2,3);;
```

member

```
# let rec member el l =
    match l with
    [] -> false
  | x::xs ->
      if el = x then true
      else member el xs;;
```

≡

```
  | x::xs when x = el -> true
  | x::xs when x <> el -> member el xs;;
```

member : 'a → 'a list → bool = <fun>

'a: tipo el; 'a list: tipo l; bool: tipo risultato

```
let rec take n l =
        match (n, l) with
  | (0, l) -> []
  | (n, []) when n>0 -> []
  | (n, x::xs) when n>0 ->
          x:: take (n-1) xs ;;
take : int -> 'a list -> 'a list = <fun>
```

```
let rec take_a n l a =
    match (n, l) with
  | (0, l) -> a
  | (n, []) when n>0 -> a
  | (n, x::xs) when n>0 ->
        take_a (n-1) xs (x::a)
```

~~(x::a)~~

(a@[x]) ;;

inefficiente.