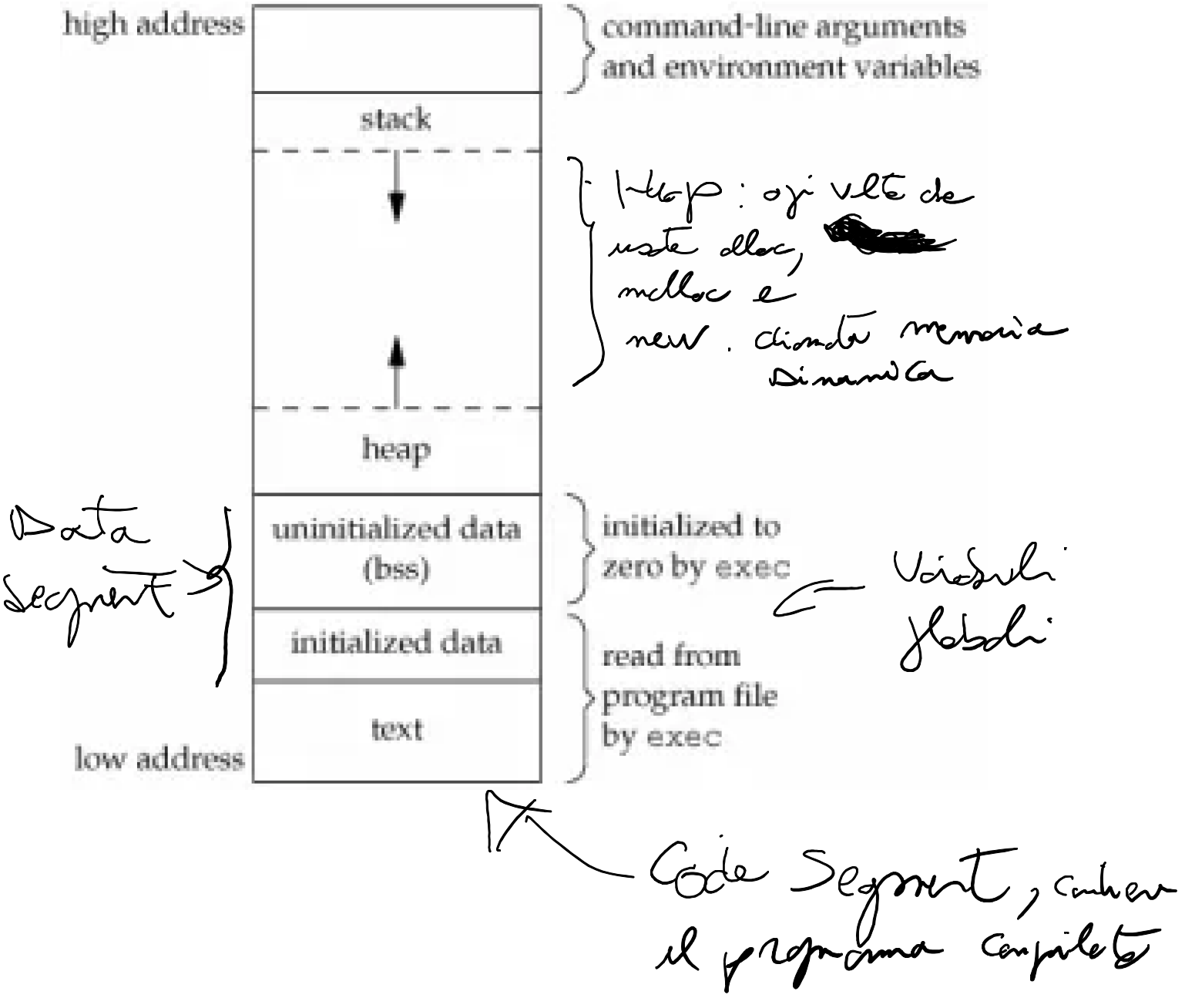
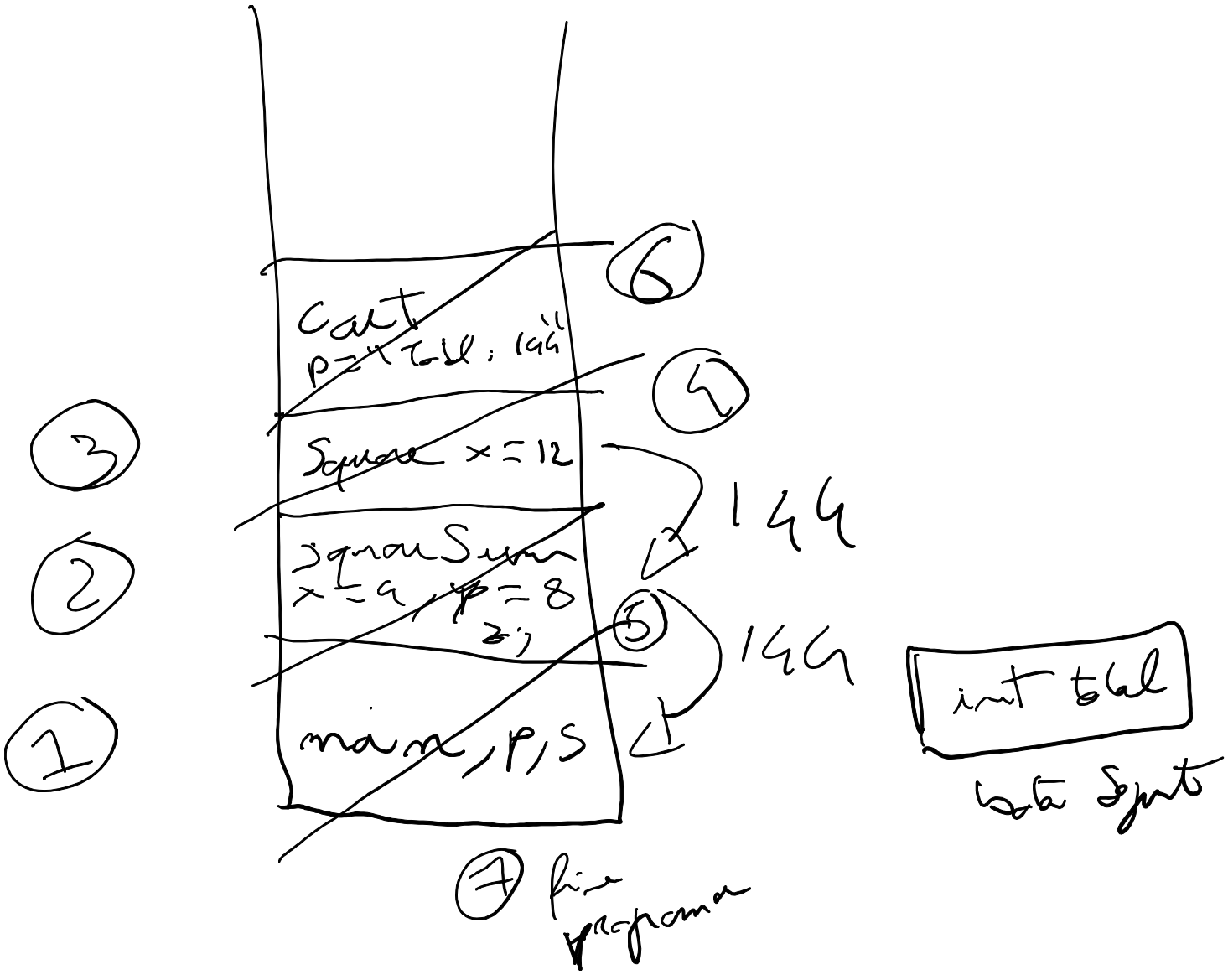
Lezione sulla ricorsione - 29 Novembre 2018
(Dalle slides del professore Cincotti)

- Meccanismo di chiamata alle funzioni
- Stack vs Heap
- Ricorsione
- Esempi: Fattoriale, Fibonacci, Torre di Hanoi.
- Ricorsione vs Iterazione

Gestione della memoria

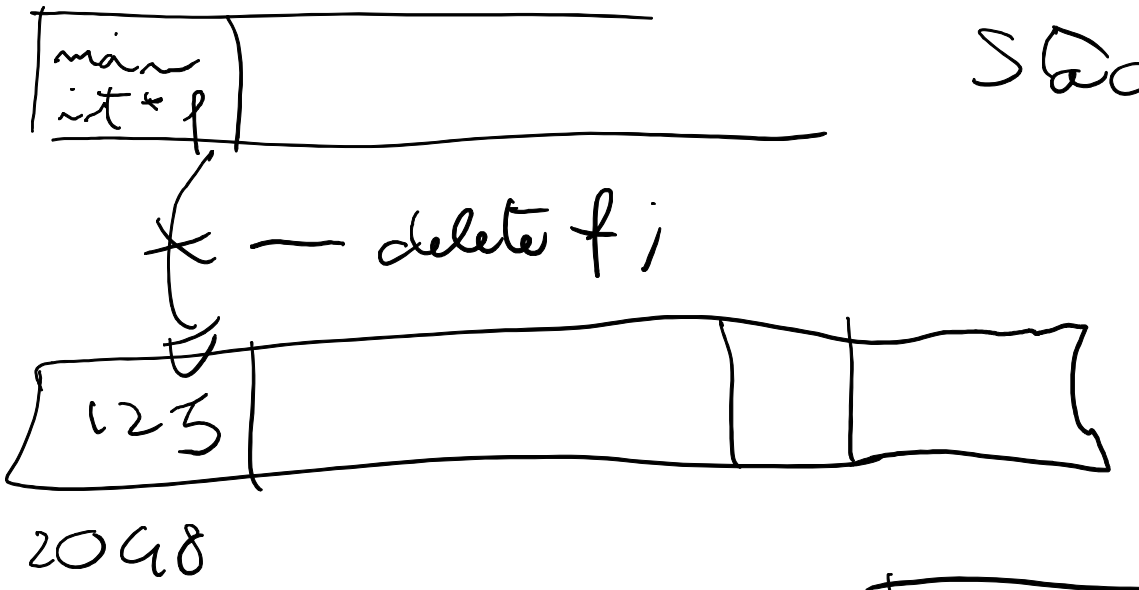


```cpp
int total;
int square(int x)
{
    return x * x;
}

int squareSum(int x, int y)
{
    int z = square(x + y);
    return z;
}

int main()
{
    int p = 4;
    int s = 8;
    total = squareSum(p, s);
    cout << "Total: " << total << endl;
}
```



```
int* f = new int;
*f = 125;
```

```
class Temp
{
public:
    int function1(int foo) { return 1; }
    void function2(int bar) { foobar = bar; }

private:
    int foobar;
};
```



Stack

Temp *t = new Temp();

Quanto occupa f ?

Quanto occupa *f ?

e quanto *t, t ?

Cosa mettiamo nella Heap ?

! Attenzione alla frammentazione della Heap.

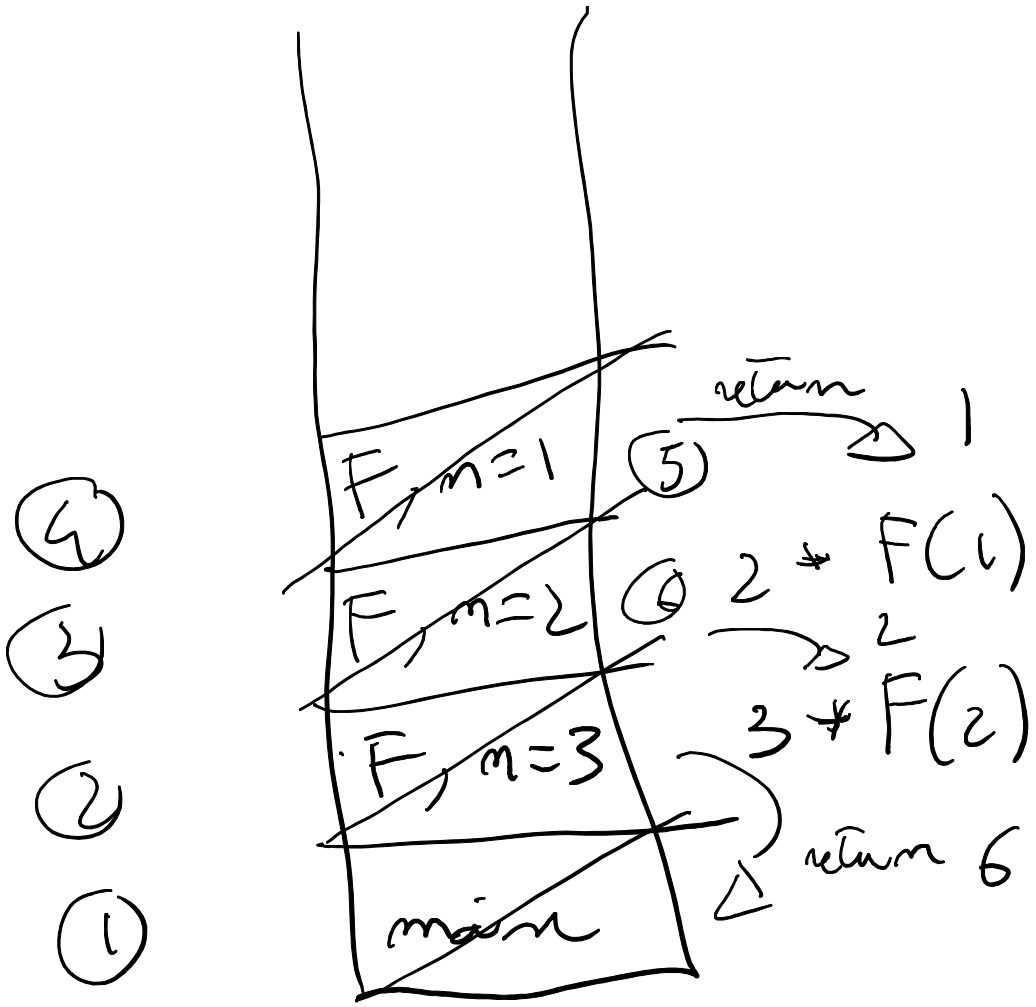
! Allineamento di memoria.

```
int  fattoriale  ( int  n )
      {
            if  ( n == 0 )    return  1;
            return   n * fattoriale ( n – 1 );
      }
```

$$n! = \begin{cases} 1 & \text{se } n = 0, \\ n \cdot (n-1)! & \text{se } n > 0. \end{cases}$$

```
int iterativeFactorial(int n)
{
    int fact = 1;
    for (int j = 1; j <= n; j++)
        fact *= j;
    return fact;
}
```

Simulazione F(3)

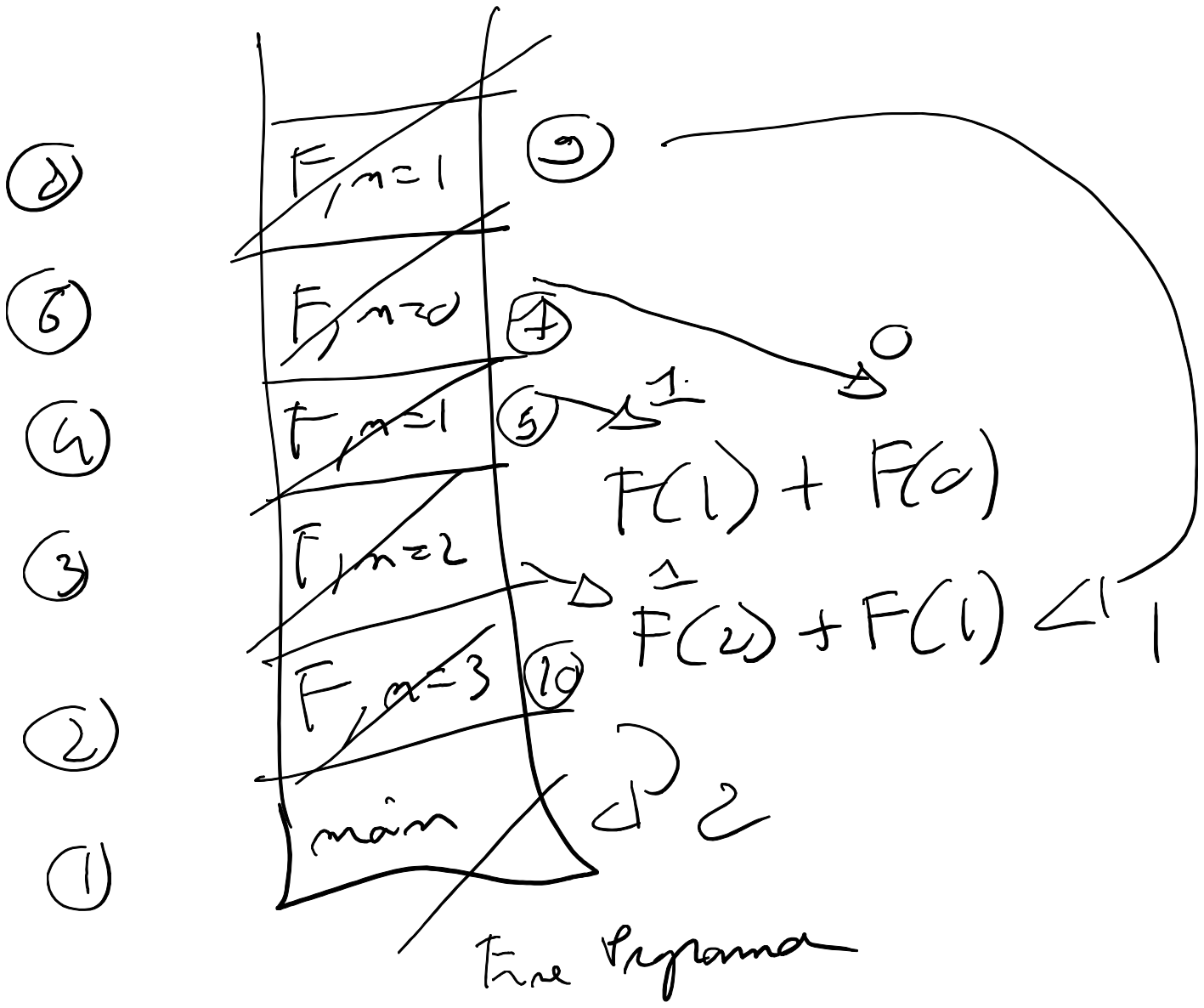


F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	4181	6765

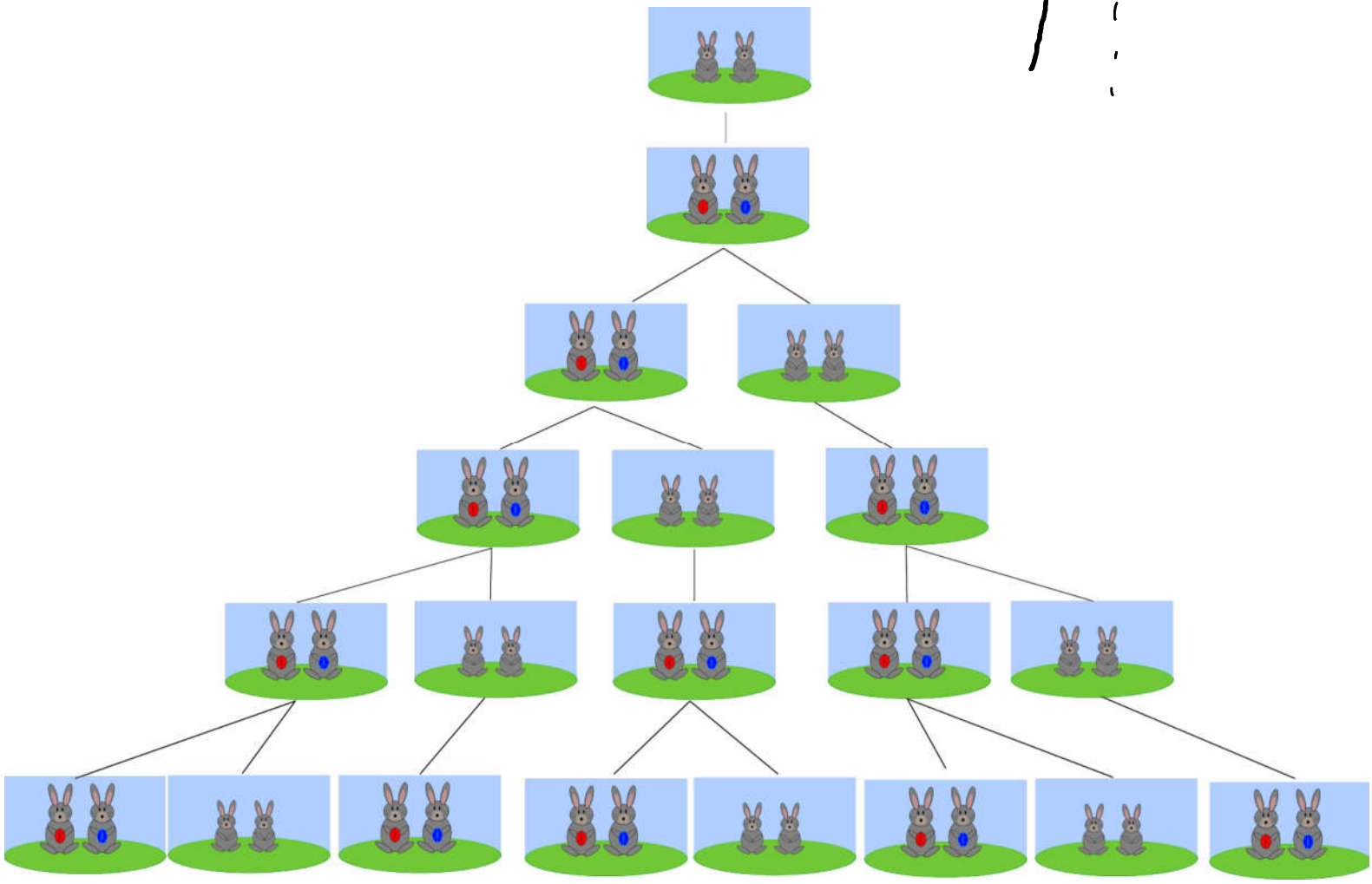
```java
int fibonacci(int n)
{
    if (n <= 1) return n;
    return (fibonacci(n - 1 +
            fibonacci(n - 2)));
}
```

```java
int fibonacciIterative(int n)
{
    if (n <= 1)
        return n;
    int temp = 0;
    int prev = 1;
    int result = 0;
    for (int i = 2; i <= n; i++)
    {
        result = prev + temp;
        temp= prev;
        prev = result;
    }
    return result;
}
```

F(3)



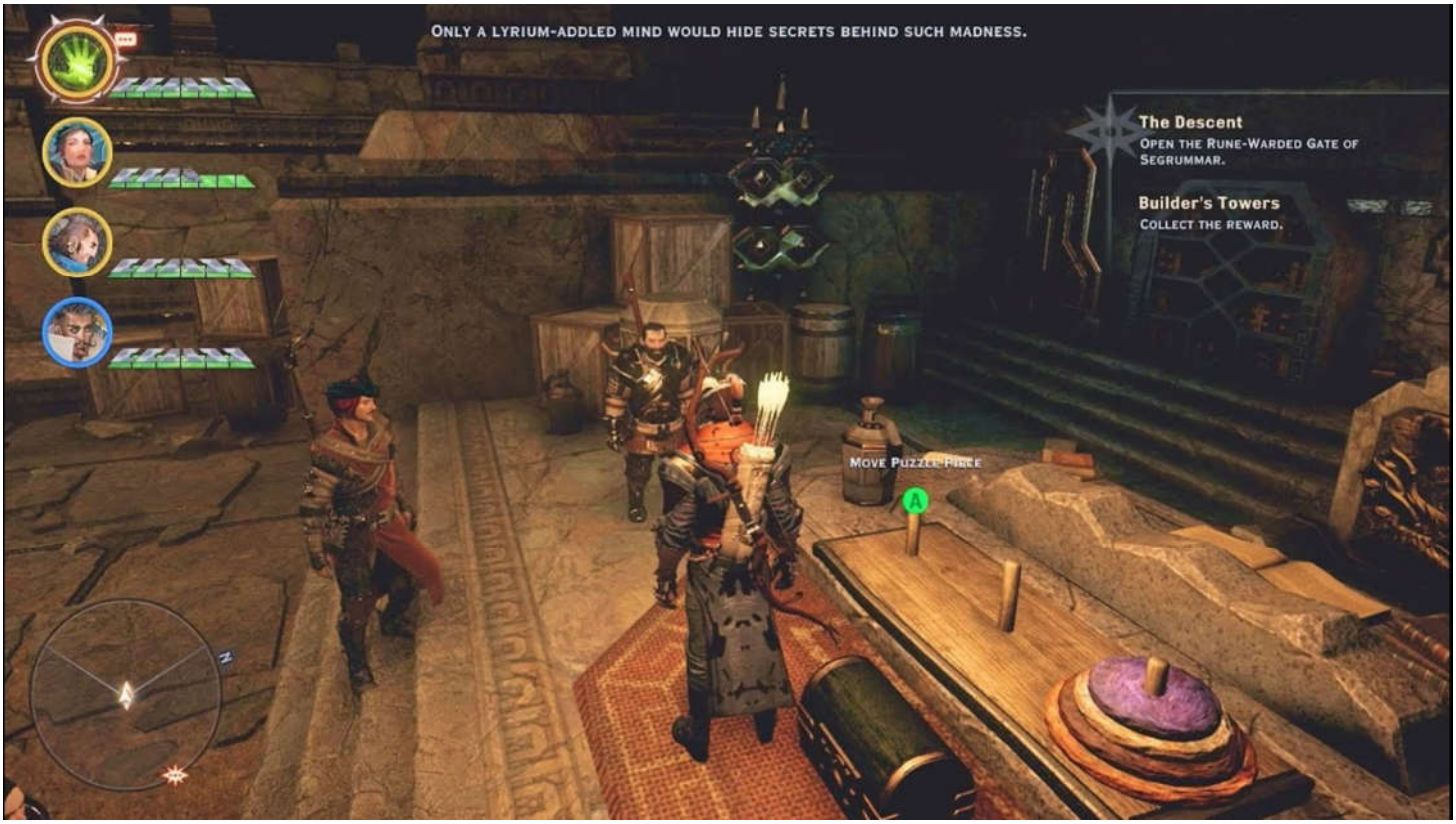
N	Recursive	Recursive opt.	Iterative
5	5 ticks	22 ticks	9 ticks
10	36 ticks	49 ticks	10 ticks
20	2315 ticks	61 ticks	10 ticks
30	180254 ticks	65 ticks	10 ticks
100	too long/stack overflow	158 ticks	11 ticks
1000	too long/stack overflow	1470 ticks	27 ticks
10000	too long/stack overflow	13873 ticks	190 ticks
100000	too long/stack overflow	too long/stack overflow	3952 ticks



I conigli diventano adulti al termine del primo mese e si riproducono al termine del secondo. I conigli colorati sono fecondi.

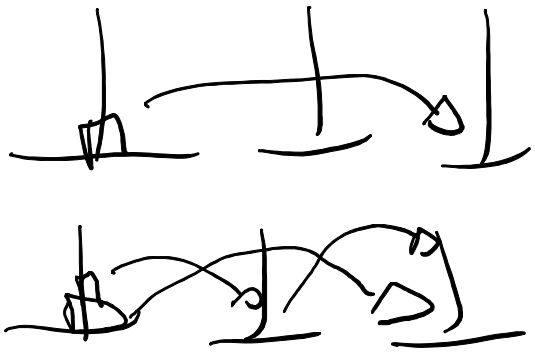
Al termine del n-esimo mese, il numero di coppie di conigli sarà pari al numero di nuove coppie (n-2) più il numero di conigli vivi al mese precedente (n-1).

La torre di Hanoi



-Dobbiamo spostare tutti i dischi dal primo piolo all'ultimo.
-Possiamo muovore un solo disco alla volta.
-Un disco può giacere solo sopra un disco più grande

Muoviamo n-1 dischi sul piolo temporaneo, poi un disco solo in quello finale, poi gli n-1 dischi sopra di esso..
La relazione di ricorrenza è H(n)=2* H(n-1)+1 .
Di quanti passi abbiamo bisogno?



progressione geometrica

$$\begin{cases} H(1) = 2*0+1 = 1 \\ H(2) = 2*H(1)+1 = 3 \\ \vdots \\ H(n-1) = 2*H(n-2)+1 = \dots \\ H(n) = 2*H(n-1)+1 = 2^n-1 \end{cases}$$

Per induzione:

$P(1)$: $H(1) = 2^1 - 1 = 1$

$P(k)$: $H(k) = 2^k - 1$ per ipotesi induttiva

Verifichiamo $P(k+1)$: $\boxed{H(k+1) = 2^{k+1} - 1}$ Usiamo Ip. Ind.

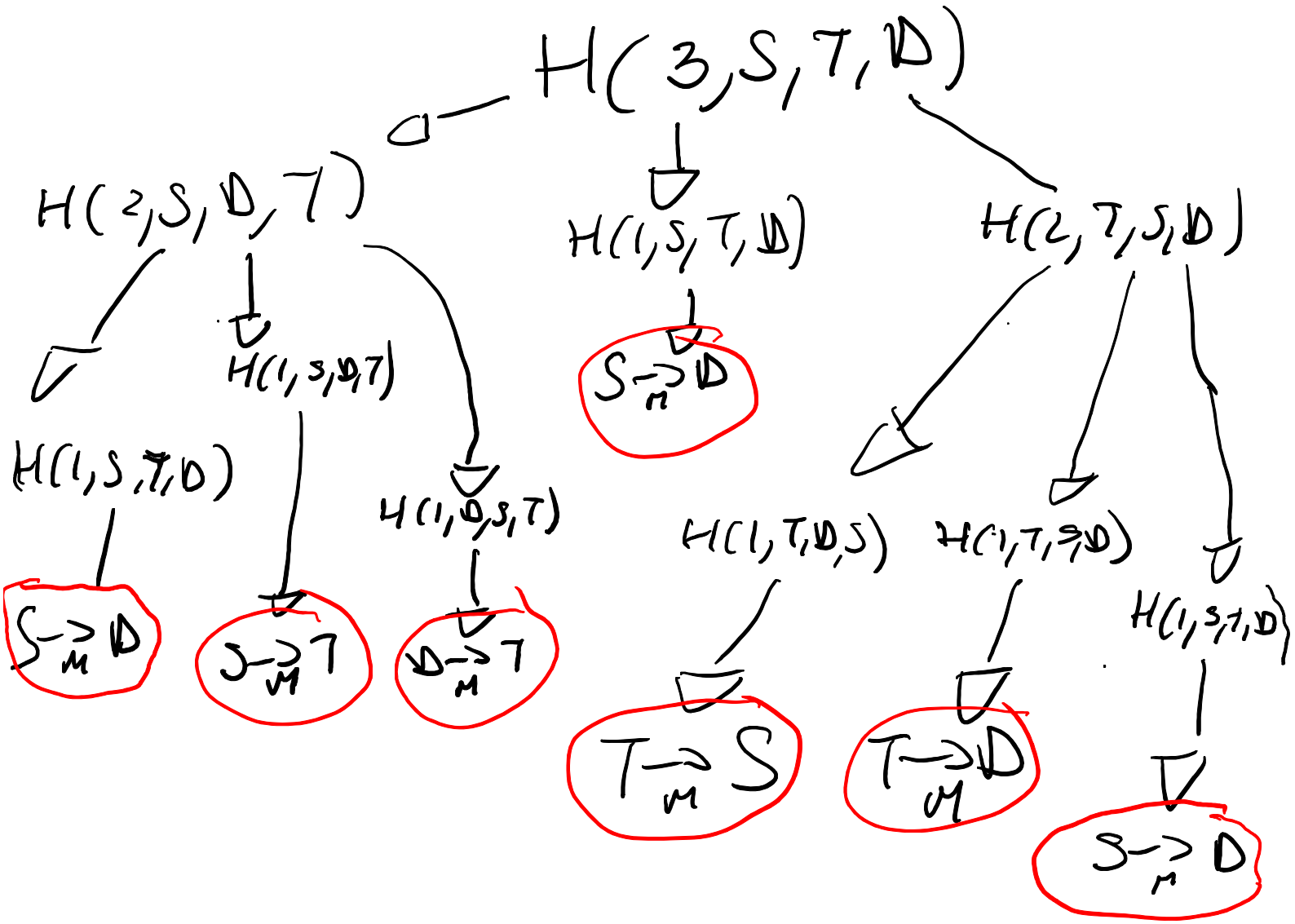
$H(k+1) = 2 \cdot (2^k - 1) - 1$

$H(k+1) = 2 \cdot 2^k - 2 - 1$

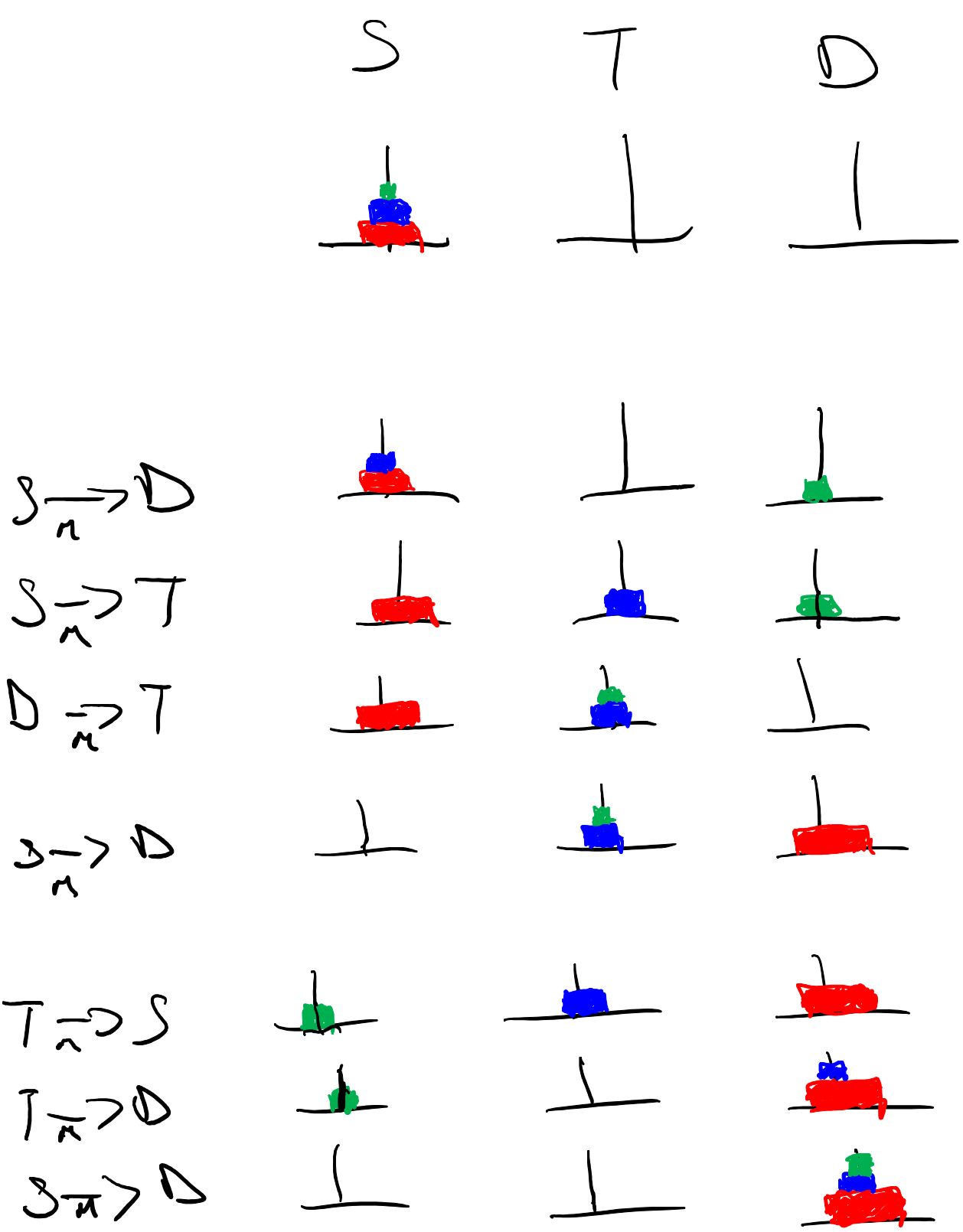
$H(k+1) = 2^{k+1} - 1$ C.V.D

```c
void hanoi(int n, int sorgente, int temporaneo, int destinazione)
{
    if (n == 1)
      return;
    hanoi(n - 1, sorgente, destinazione, temporaneo);
    hanoi(1, sorgente, temporaneo, destinazione);
    hanoi(n - 1, temporaneo, sorgente, destinazione);
}
```

$n = 3$



Leggiamo le mosse da sinistra a destra



n dispari, per n pari cambia il piolo della prima mossa.

Soluzione ricorsiva (senza ausilio di stack)?
 -Prima mossa, dipende dal numero di dischi (pari o dispari).
 -Mosse successive, muovere la cima su un piolo (diverso da quello in cui si trova).
-Muovere (legalmente) un altro disco (c'è una sola possibilità).
-Considera un set di dischi in una sotto soluzione ottimale.

Tuttavia, la posizione di un disco può essere determinata direttamente dalla rappresentazione binaria del numero di mossa. Questa soluzione è nota come soluzione binaria. Esistono anche altre soluzioni.