Laboratorio I
a.a. 2025/2026

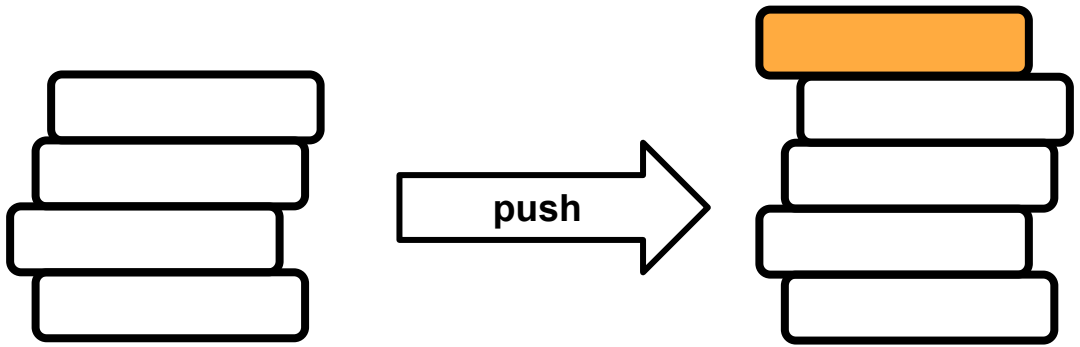
Ricorsione

Funzioni ricorsive

- Funzioni che **utilizzano loro stesse**
  = contengono almeno una chiamata a loro stesse
- Utilizzano lo **stack di record di attivazione** (pila)

Funzioni ricorsive

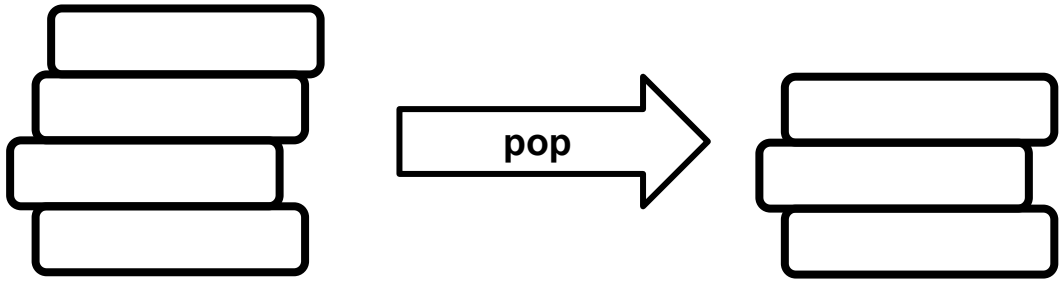
- Funzioni che **utilizzano loro stesse**
  = contengono almeno una chiamata a loro stesse
- Utilizzano lo **stack di record di attivazione** (pila)



stack

Funzioni ricorsive

- Funzioni che **utilizzano loro stesse**
  = contengono almeno una chiamata a loro stesse
- Utilizzano lo **stack di record di attivazione** (pila)

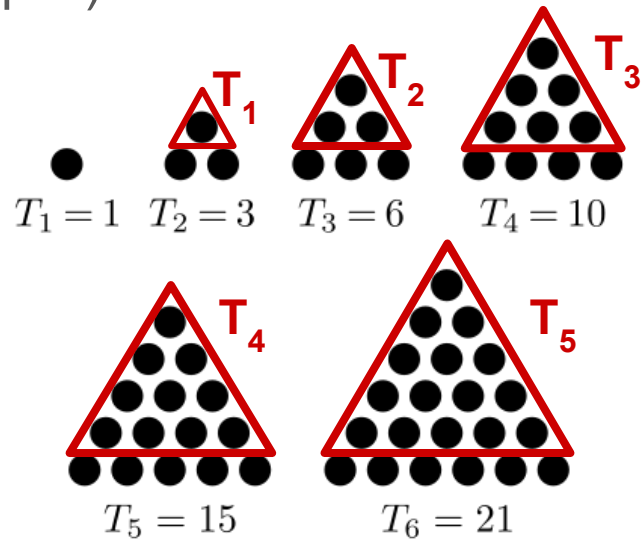


stack

Funzioni ricorsive

- Funzioni che **utilizzano loro stesse**
  = contengono almeno una chiamata a loro stesse
- Utilizzano lo **stack di record di attivazione** (pila)

Esempio: trovare l'n-simo **numero triangolare** (https://it.wikipedia.org/wiki/Numero_triangolare)
Ovvero, numeri "rappresentabili in forma di triangolo equilatero" come mostrato in figura:




```
function triangle(n){
  if(n==1)
    return 1;
  return n+triangle(n-1);
}
```

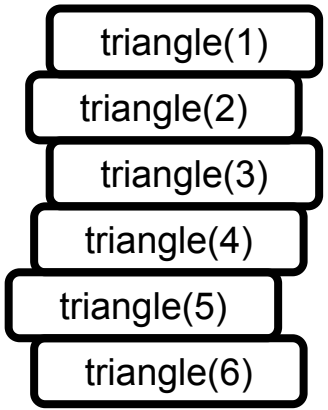
Funzioni ricorsive

- Funzioni che **utilizzano loro stesse**
  = contengono almeno una chiamata a loro stesse
- Utilizzano lo **stack di record di attivazione** (pila)

Esempio: trovare l'n-simo **numero triangolare**

```
function triangle(n){
  if(n==1)
    return 1;
  return n+triangle(n-1);
}
```

n = 6



Stack di record di attivazione

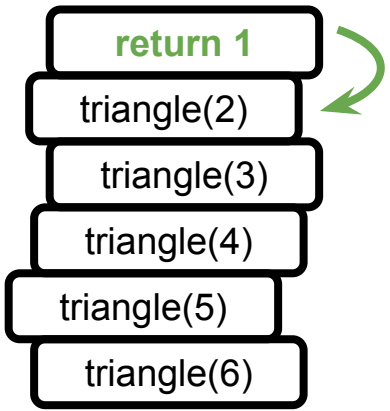
Funzioni ricorsive

- Funzioni che **utilizzano loro stesse**
  = contengono almeno una chiamata a loro stesse
- Utilizzano lo **stack di record di attivazione** (pila)

Esempio: trovare l'n-simo **numero triangolare**

```
function triangle(n){
  if(n==1)
    return 1;
  return n+triangle(n-1);
}
```

n = 6



Stack di record di attivazione

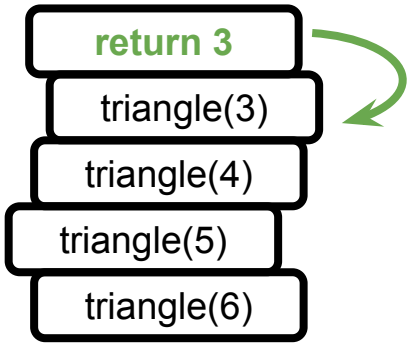
Funzioni ricorsive

- Funzioni che **utilizzano loro stesse**
  = contengono almeno una chiamata a loro stesse
- Utilizzano lo **stack di record di attivazione** (pila)

Esempio: trovare l'n-simo **numero triangolare**

```
function triangle(n){
  if(n==1)
    return 1;
  return n+triangle(n-1);
}
```

n = 6



Stack di record di attivazione

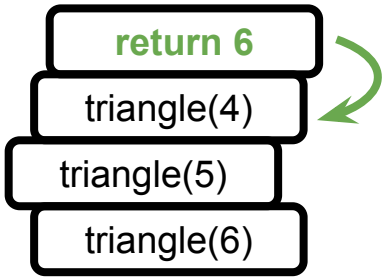
Funzioni ricorsive

- Funzioni che **utilizzano loro stesse**
  = contengono almeno una chiamata a loro stesse
- Utilizzano lo **stack di record di attivazione** (pila)

n = 6

Esempio: trovare l'n-simo **numero triangolare**

```
function triangle(n){
  if(n==1)
    return 1;
  return n+triangle(n-1);
}
```



Stack di record di attivazione

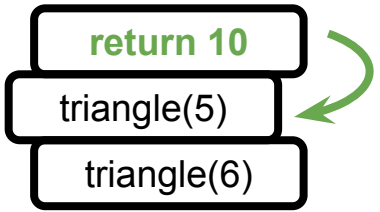
Funzioni ricorsive

- Funzioni che **utilizzano loro stesse**
  = contengono almeno una chiamata a loro stesse
- Utilizzano lo **stack di record di attivazione** (pila)

n = 6

Esempio: trovare l'n-simo **numero triangolare**

```
function triangle(n){
  if(n==1)
    return 1;
  return n+triangle(n-1);
}
```



Stack di record di attivazione

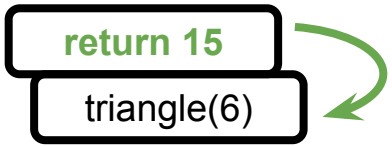
Funzioni ricorsive

- Funzioni che **utilizzano loro stesse**
  = contengono almeno una chiamata a loro stesse
- Utilizzano lo **stack di record di attivazione** (pila)

n = 6

Esempio: trovare l'n-simo **numero triangolare**

```
function triangle(n){
  if(n==1)
    return 1;
  return n+triangle(n-1);
}
```



Stack di record di attivazione

Funzioni ricorsive

- Funzioni che **utilizzano loro stesse**
  = contengono almeno una chiamata a loro stesse
- Utilizzano lo **stack di record di attivazione** (pila)

n = 6

Esempio: trovare l'n-simo **numero triangolare**

```
function triangle(n){
  if(n==1)
    return 1;
  return n+triangle(n-1);
}
```

return 21

Stack di record di attivazione

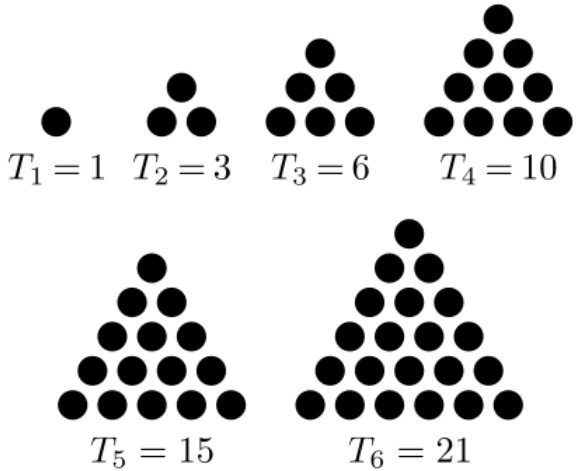
Funzioni ricorsive

- Implementazione di **definizioni induttive** per insiemi o sequenze o funzioni
  - Passo base - calcolo diretto
  - Passo induttivo - regola per calcolo ricorsivo

- Molto adatte per lavorare con strutture dati che rappresentano '**collezioni**' di oggetti
  - Definizione induttiva della lista: una lista di lunghezza l consiste in un elemento più una lista di lunghezza l-1

Funzioni ricorsive

- **Ricorsione in coda**: ultima operazione è la chiamata ricorsiva
  - Molto più efficiente
  - Si riutilizza lo stack di record di attivazione




```
function triangle(n){
  if(n==1)
    return 1;
  return n+triangle(n-1);
}

function triangleCoda(n, s=0){
  if(n<1)
    return s;
  return triangleCoda(n-1,s+n);
}
```

Dopo la chiamata ricorsiva, il controllo va passato al chiamante per effettuare operazioni

Esercizio 1: Cifre

Scrivere le seguenti funzioni **ricorsive**:

- `stampaCifreRec`: dato in input un numero n in base dieci, stampa separatamente ogni sua cifra, in ordine dalla piú alla meno significativa
- `arrayCifreRec`: variante della precedente, che invece di stampare ritorna un array contenente le cifre di n, in ordine dalla piú alla meno significativa

  Esempio: Input: 164392

  Output: [1, 6, 4, 3, 9, 2]

Esercizio 2: Fibonacci

La successione di Fibonacci F_0, F_1, F_2, … è una sequenza infinita di numeri definita (ricorsivamente) come segue:

- $F_0 = 0$ ed $F_1 = 1$
- Per ogni altro $n>1$, $F_n = F_{n-1} + F_{n-2}$

I primi 7 numeri della successione sono ad esempio: 0,1,1,2,3,5,8

1. Scrivere una funzione ricorsiva **`fibonacciNsimo`** che, dato n, calcola F_n
2. Scrivere una funzione ricorsiva **`fibonacciPrimiN`** che, dato n, ritorna un array contenente i primi n+1 numeri della successione

Esercizio 2: Fibonacci

1. Scrivere una funzione ricorsiva **`fibonacciNsimo`** che, dato n, calcola F_n

   Esempio: Input: 4

   Output: 3 = F_4

2. Scrivere una funzione ricorsiva **`fibonacciPrimiN`** che, dato n, ritorna un array contenente i primi n+1 numeri della successione

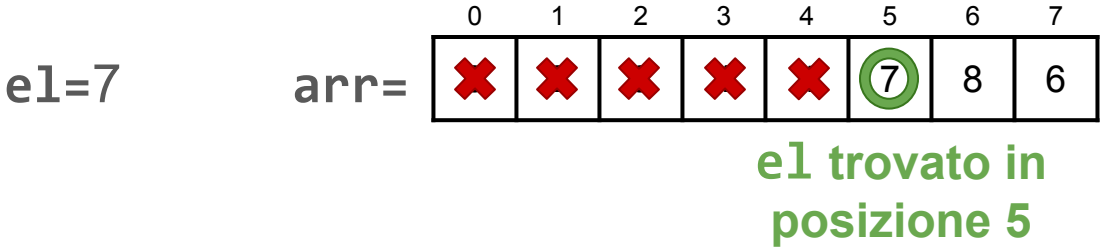
   Esempio: Input: 5

   Output: [0,1,1,2,3,5]

Esercizio 3: Ricerca in un array

Ci sono diversi algoritmi per cercare un dato elemento **`el`** in un array **`arr`**

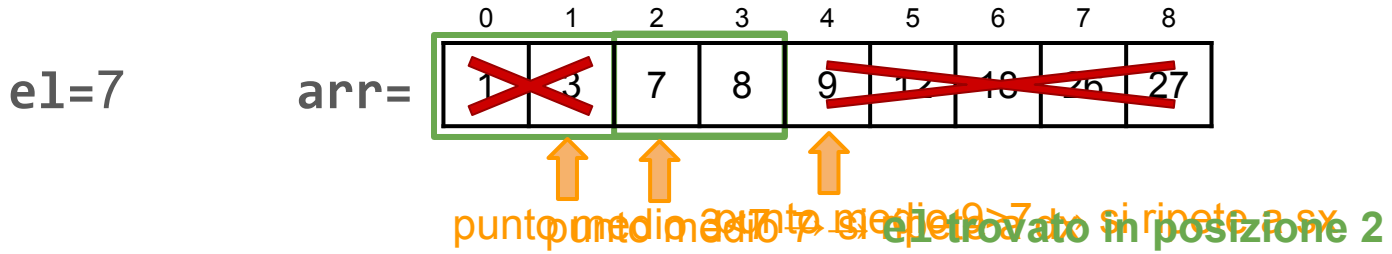
- **Ricerca lineare**: Si scorre **`arr`**, fermandoci quando si trova un'occorrenza di **`el`**



Esercizio 3: Ricerca in un array

Ci sono diversi algoritmi per cercare un dato elemento **`el`** in un array **`arr`**

- **Ricerca lineare**: Si scorre **`arr`**, fermandoci quando si trova un'occorrenza di **`el`**
- Se l'array è **ordinato**, si può fare una **ricerca binaria**: si divide **`arr`** in due parti uguali, e si ripete la ricerca nella unica (per via dell'ordine) metá dove **`el`** può apparire



Esercizio 3: Ricerca in un array

Ci sono diversi algoritmi per cercare un dato elemento `el` in un array `arr`

- **Ricerca lineare**: Si scorre `arr`, fermandoci quando si trova un'occorrenza di `el`
- Se l'array è **ordinato**, si può fare una **ricerca binaria**: si divide `arr` in due parti uguali, e si ripete la ricerca nella unica (per via dell'ordine) metá dove `el` può apparire

Implementare entrambe le ricerche mediante funzioni ricorsive (se l'elemento non appartiene all'array, restituire -1).

Q & A