Che cos'è un programma?

Algoritmo Babilonese per la Radice Quadrata

Un metodo vecchio di 4000 anni, ancora usato oggi

La formula

Per calcolare $\sqrt{S}$, si parte da una stima x_0 e si itera:

$$x_{n+1} = \frac{x_n + \frac{S}{x_n}}{2}$$

Convergenza

Bastano **5–6 iterazioni** per raggiungere la precisione massima di un calcolatore (15 cifre decimali).

Perché funziona?

Si crea una serie di rettangoli di area S, e ad ogni step la differenza tra i lati diminuisce.

```
S : 2.0
x : S

for i in 1..10
    x : (x + S / x) / 2.0
```

```
step 1: x = 1.5
step 2: x = 1.4166666666666665
step 3: x = 1.4142156862745097
step 4: x = 1.4142135623746899
step 5: x = 1.414213562373095
step 6: x = 1.414213562373095
step 7: x = 1.414213562373095
step 8: x = 1.414213562373095
step 9: x = 1.414213562373095
sqrt(2) = 1.414213562373095
```

Ho una mirabile dimostrazione algebrica ma non rientra nel margine della slide

...però non è un vero programma, non usa strutture dati!

Strutture Dati: Stack e Queue

Due modi fondamentali di organizzare i dati

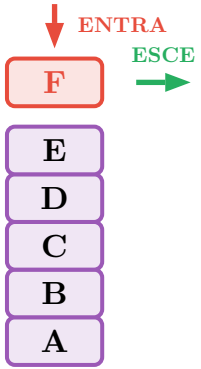
Queue (Coda)

FIFO — First In, First Out

Come una **fila al supermercato**: il primo che arriva è il primo ad essere servito. Si aggiunge **in fondo** e si toglie **dalla testa**.

Stack (Pila)

LIFO — Last In, First Out

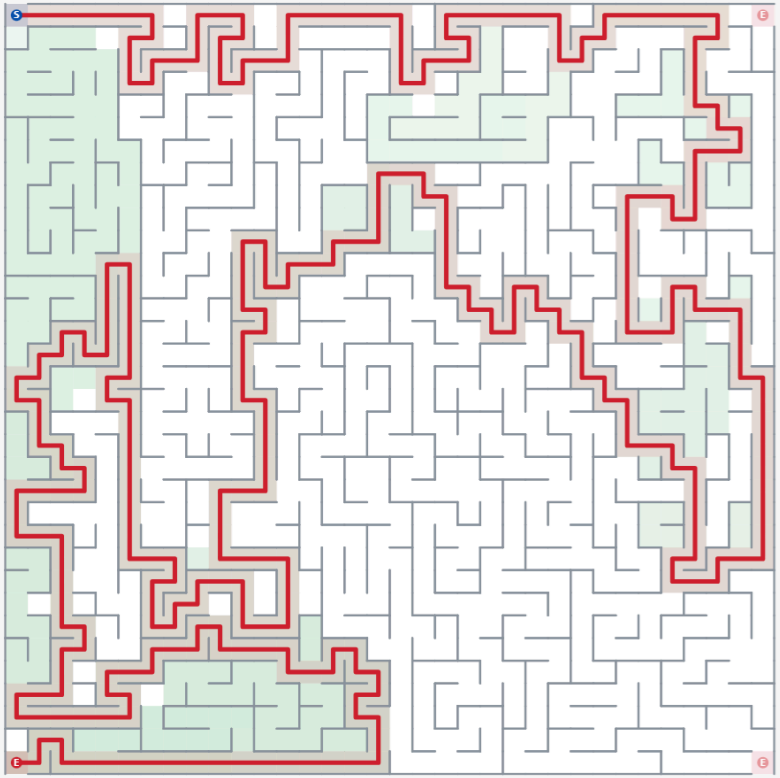


Come una **pila di piatti**: l'ultimo piatto appoggiato è il primo che togli. Si aggiunge e si toglie sempre **dalla cima**.

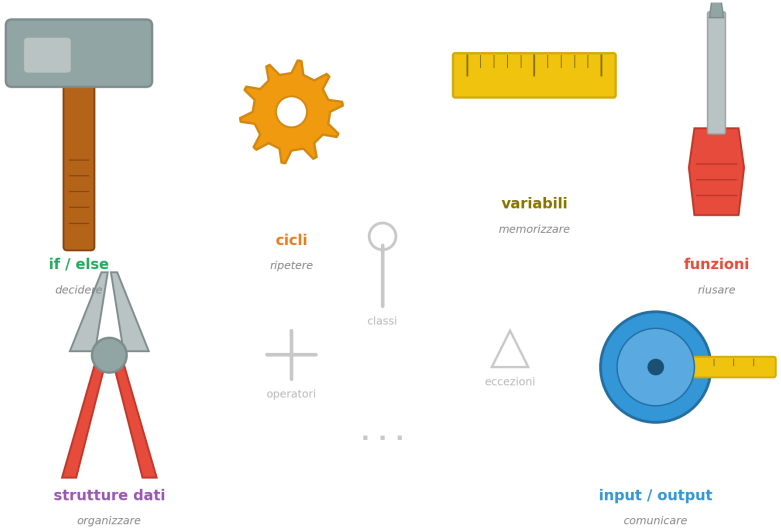
```python
Edge = tuple

def edge(A, B):
    return Edge(sorted([A, B]))

def confini(square):
    """The 4 neighbors of an (x, y) square."""
    (x, y) = square
    return {(x + 1, y), (x - 1, y),
            (x, y + 1), (x, y - 1)}

def cerca(maze, frontier):
    start = (0, 0)
    uscite = {
        (maze.width - 1, maze.height - 1),
        (maze.width - 1, 0),
        (0, maze.height - 1)
    }
    frontier.put(start)
    paths = {start: [start]}
    while frontier:
        s = frontier.pop()
        if s in uscite:
            return paths[s]
        for snew in confini(s):
            if snew not in paths \
               and edge(s, snew) in maze.edges:
                frontier.put(snew)
                paths[snew] = paths[s] + [snew]

soluzione = cerca(maze, Stack())
```

maze · dfs grid 34x34 path 304 steps explored 503 cells

size speed new maze solve step



La Cassetta degli Attrezzi del Programmatore



Condizioni (`if`)

```
if eta >= 18:
  print("ok")
```

Cicli (`for`)

```
for i in range(10):
  print(i)
```

Variabili

```
x = 42
nome = "Alice"
```

Funzioni

```
def area(b, h):
  return b*h/2
```

Strutture Dati

```
voti = [8, 7, 9]
d = {"a": 1}
```

Input / Output

```
n = input("?")
print(n)
```

Che cifra è?