

Fondamenti di Programmazione (A)

I - Problemi e algoritmi

Vincenzo Arceri - Università degli Studi di Parma - vincenzo.arceri@unipr.it

Problema

Problema

- Minimo comune multiplo

Problema

- Minimo comune multiplo
- Calcolo delle radici di un'equazione di secondo grado

Problema

- Minimo comune multiplo
- Calcolo delle radici di un'equazione di secondo grado
- Ricerca di un contatto nella rubrica

Problema

- Minimo comune multiplo
- Calcolo delle radici di un'equazione di secondo grado
- Ricerca di un contatto nella rubrica
- Calcolo tragitto più veloce fra due località

Problema

- Minimo comune multiplo
- Calcolo delle radici di un'equazione di secondo grado
- Ricerca di un contatto nella rubrica
- Calcolo tragitto più veloce fra due località
- ...

Problema

- Minimo comune multiplo
- Calcolo delle radici di un'equazione di secondo grado
- Ricerca di un contatto nella rubrica
- Calcolo tragitto più veloce fra due località
- ...

Problema

Il nostro scopo è “risolvere” un problema dato

Problema

- Minimo comune multiplo
- Calcolo delle radici di un'equazione di secondo grado
- Ricerca di un contatto nella rubrica
- Calcolo tragitto più veloce fra due località
- ...

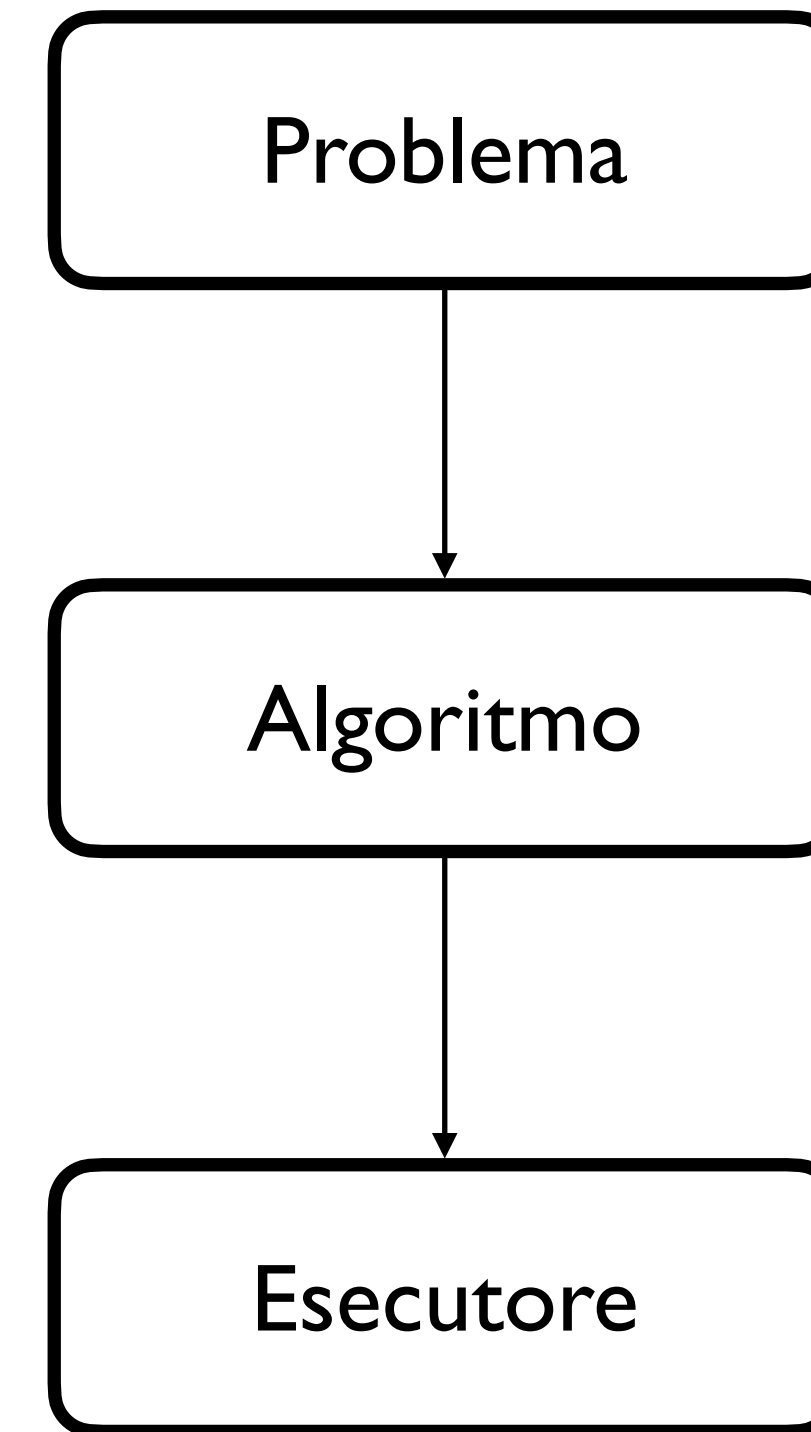
Problema

Il nostro scopo è “risolvere” un problema dato tramite *un metodo risolutivo* che possa essere eseguito, capito e applicato da un *esecutore*

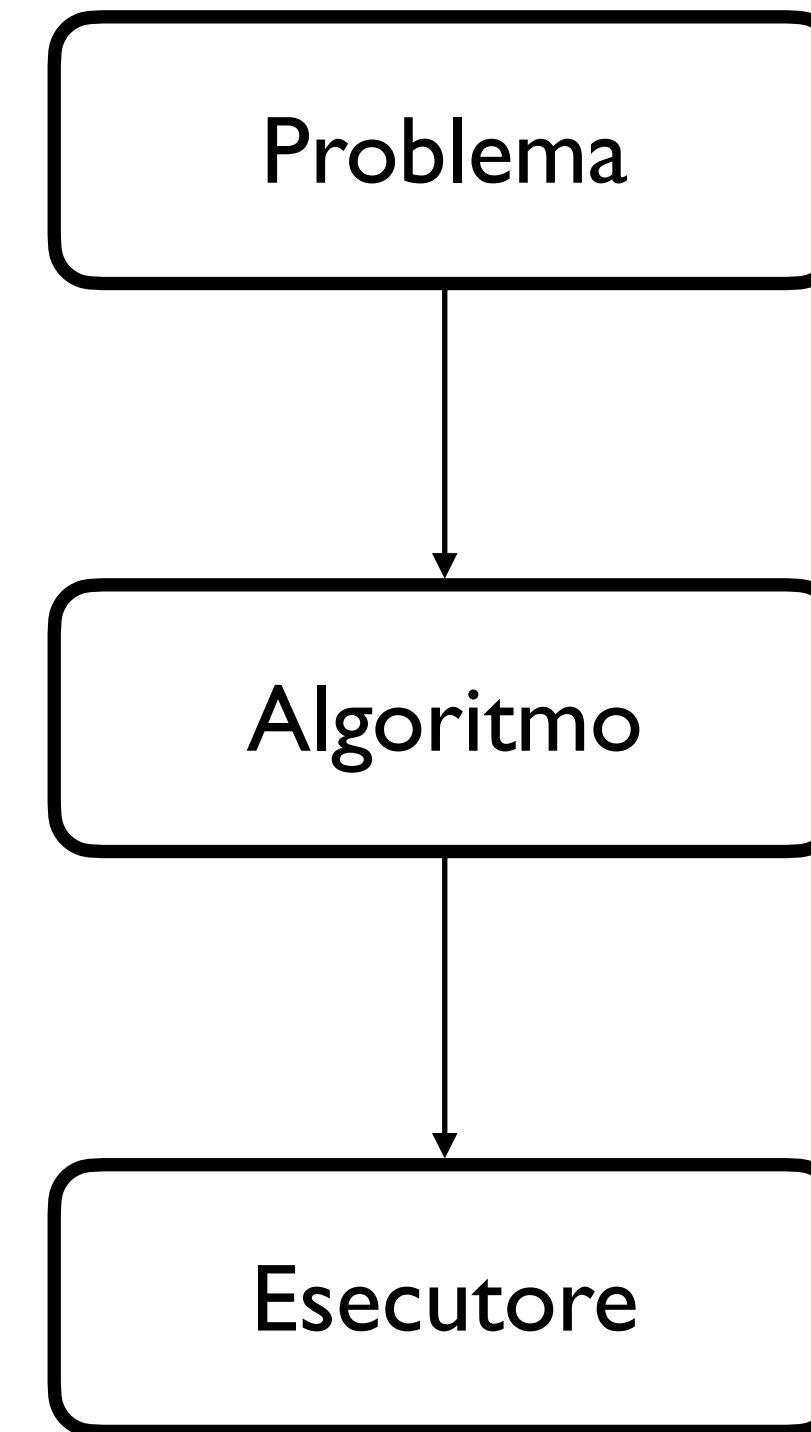
Problema

- Minimo comune multiplo
- Calcolo delle radici di un'equazione di secondo grado
- Ricerca di un contatto nella rubrica
- Calcolo tragitto più veloce fra due località
- ...

Il nostro scopo è “risolvere” un problema dato tramite *un metodo risolutivo* che possa essere eseguito, capito e applicato da un *esecutore*

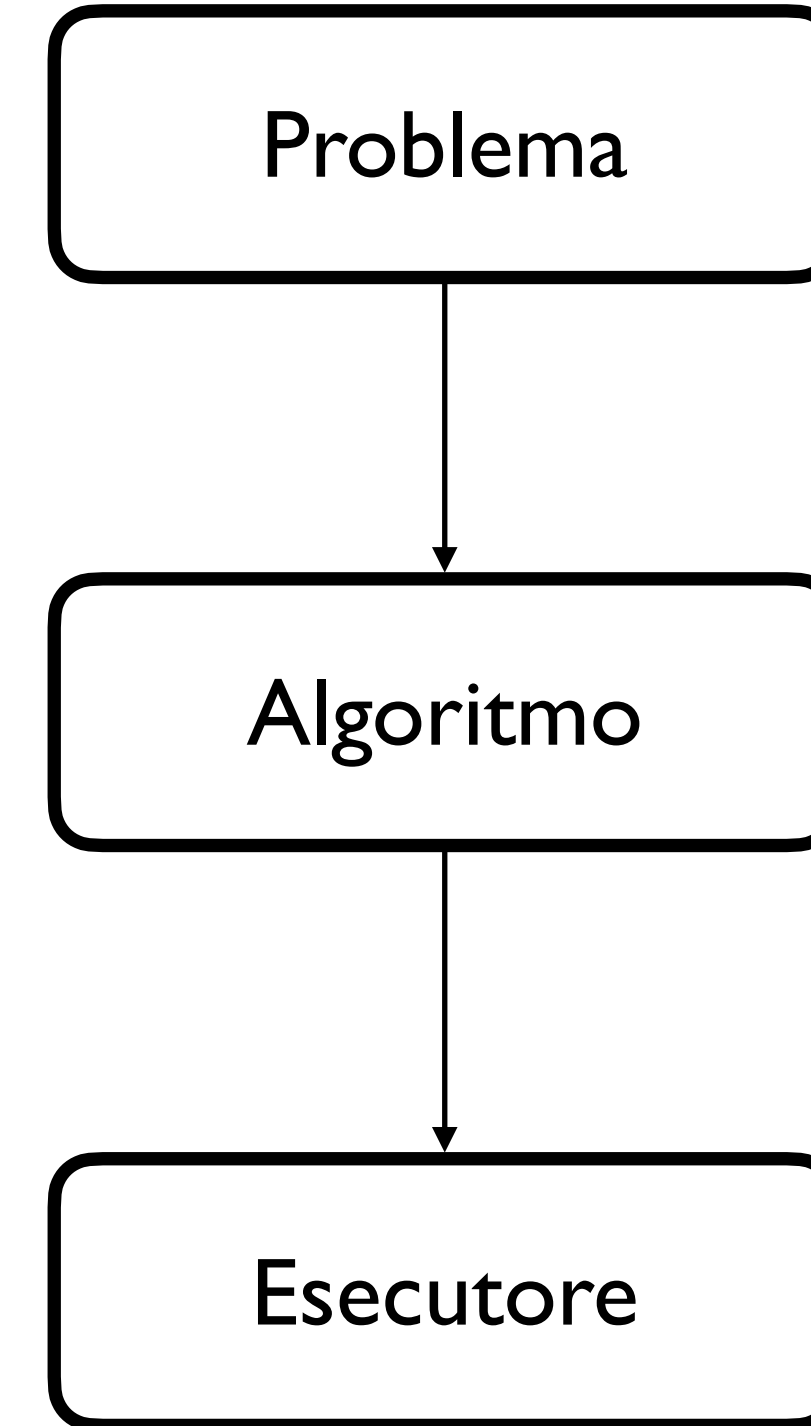


Algoritmo



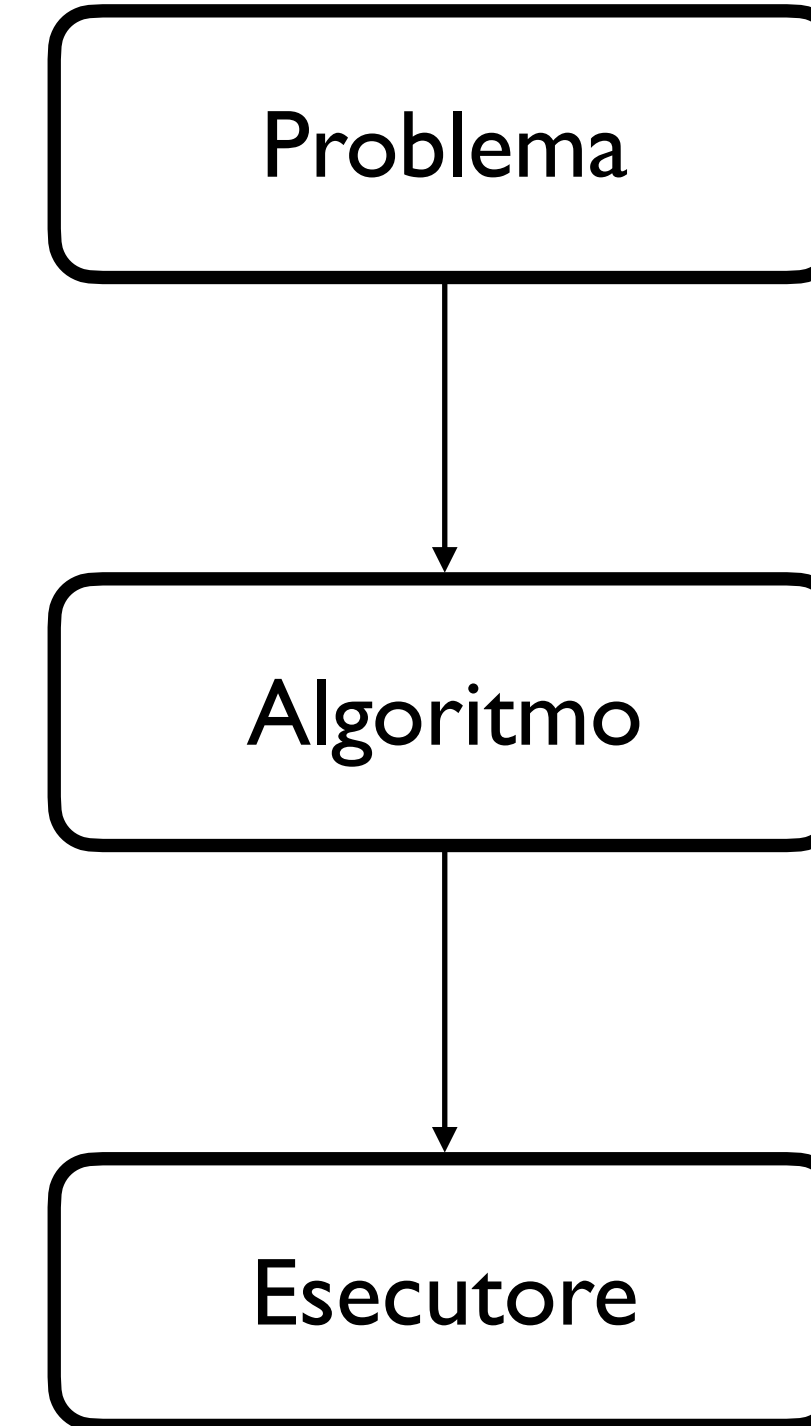
Algoritmo

- Sequenza finita di **istruzioni** che specificano come certe **operazioni elementari** debbano susseguirsi nel tempo per risolvere una data **classe di problemi**



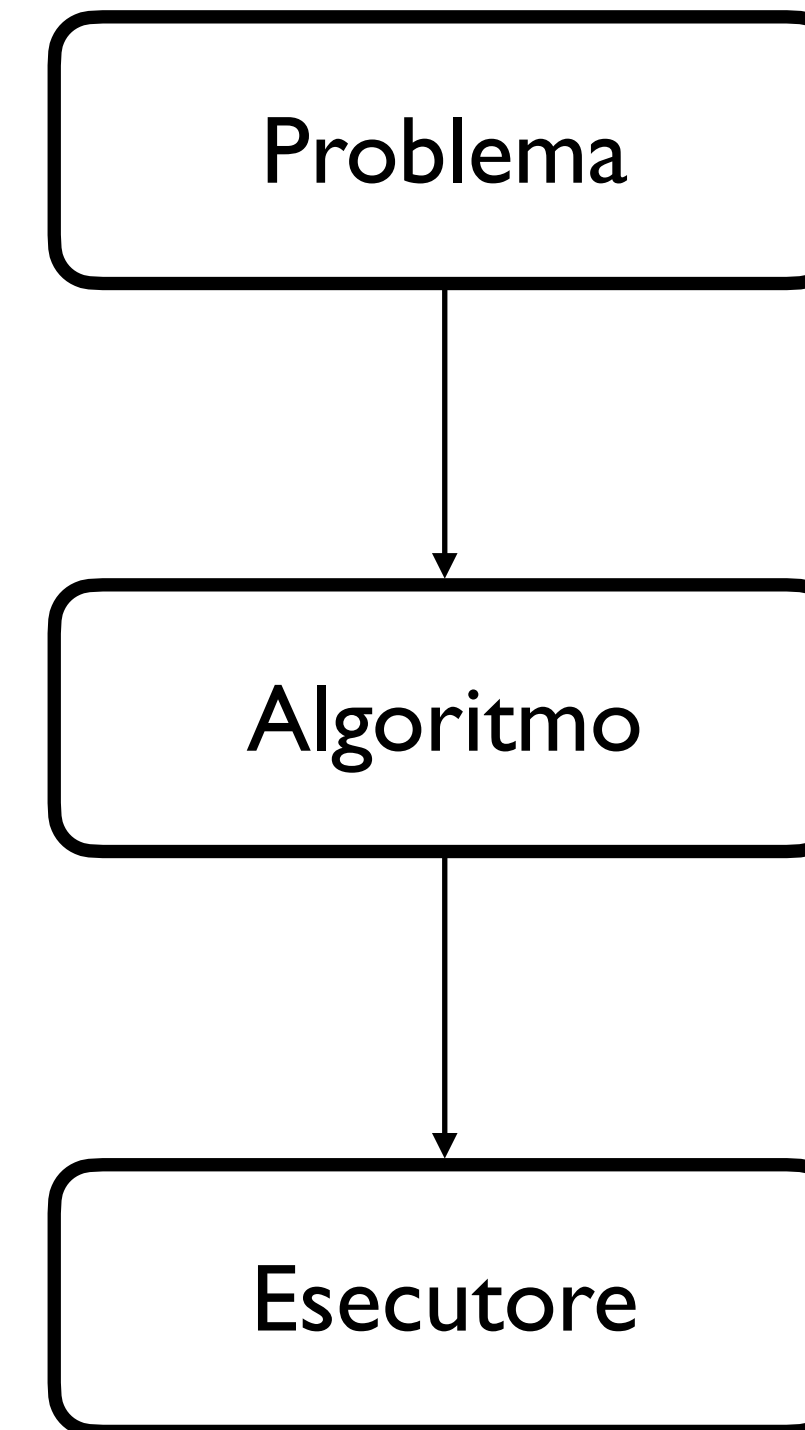
Algoritmo

- Sequenza finita di **istruzioni** che specificano come certe **operazioni elementari** debbano susseguirsi nel tempo per risolvere una data **classe di problemi**
- **Istruzioni:** richieste di azioni rivolte all'esecutore, che deve essere in grado di capirle ed eseguire



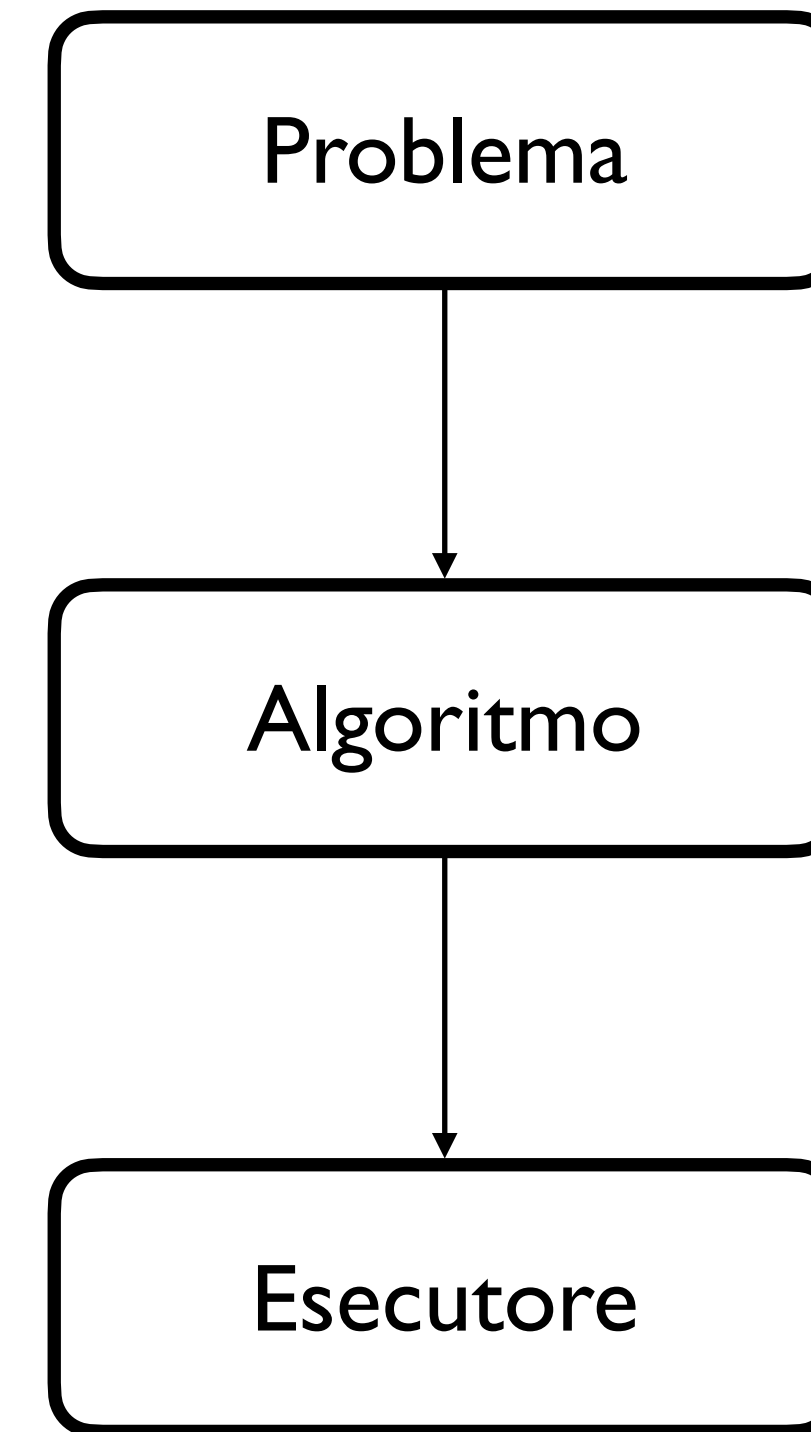
Algoritmo

- Sequenza finita di **istruzioni** che specificano come certe **operazioni elementari** debbano susseguirsi nel tempo per risolvere una data **classe di problemi**
- **Istruzioni:** richieste di azioni rivolte all'esecutore, che deve essere in grado di capirle ed eseguire
- **Operazioni elementari:** insieme di operazioni che si assume siano note all'esecutore (cioè, che è in grado di eseguire)



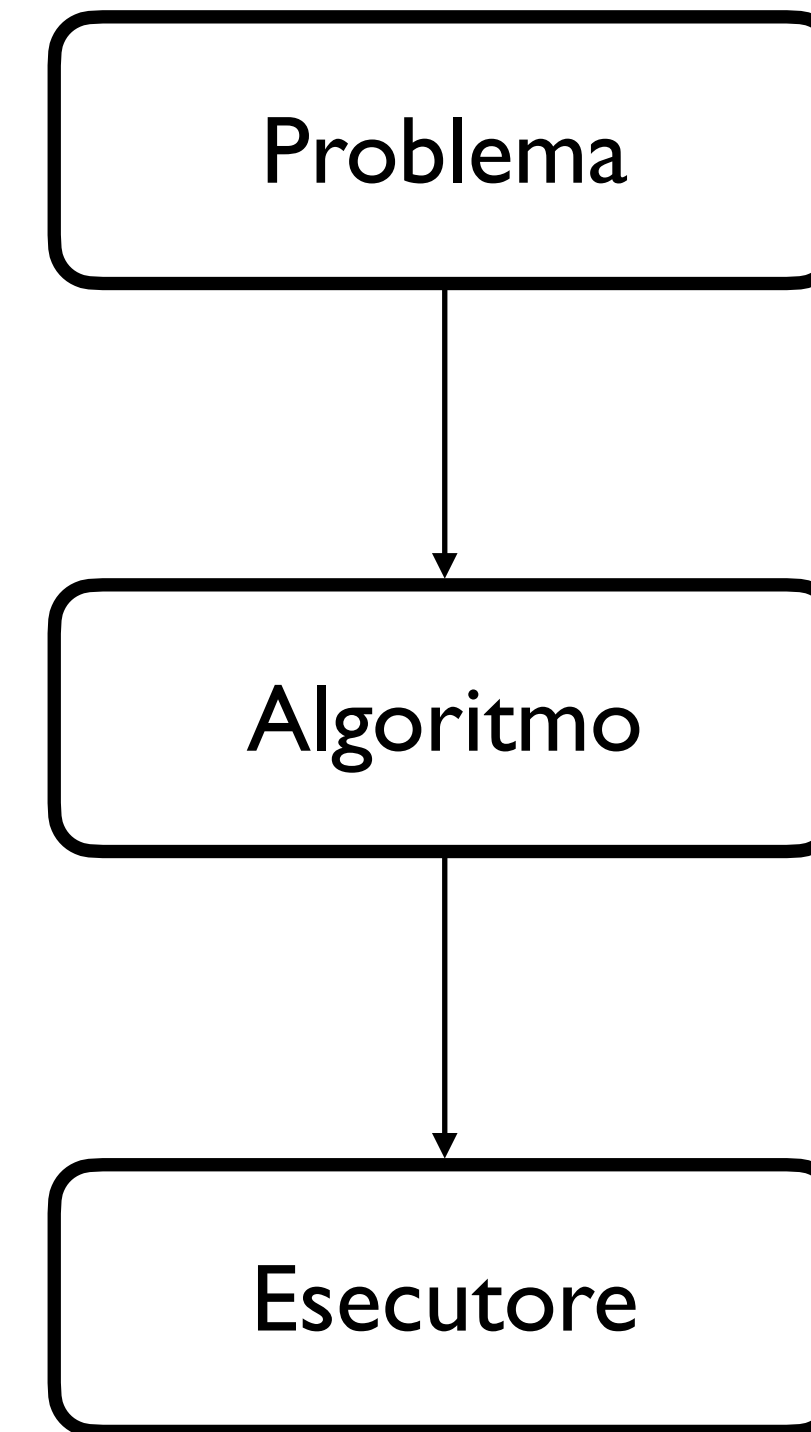
Algoritmo

- Sequenza finita di **istruzioni** che specificano come certe **operazioni elementari** debbano susseguirsi nel tempo per risolvere una data **classe di problemi**
- **Classe di problemi:** si intende che la formulazione del problema sia *indipendente* dagli specifici dati su cui opera



Algoritmo

- Sequenza finita di **istruzioni** che specificano come certe **operazioni elementari** debbano susseguirsi nel tempo per risolvere una data **classe di problemi**
- **Classe di problemi:** si intende che la formulazione del problema sia *indipendente* dagli specifici dati su cui opera

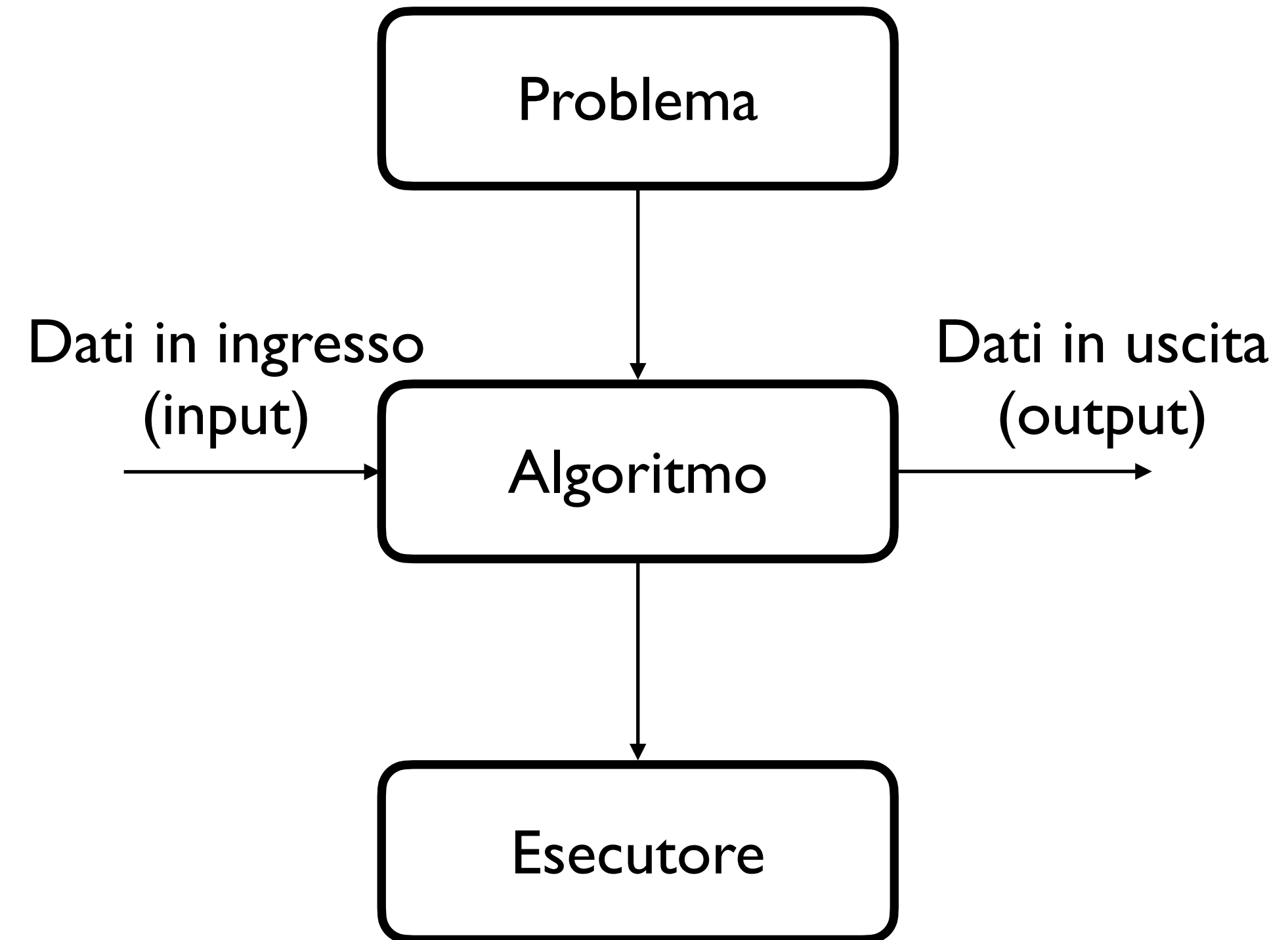


Calcolare il minimo comune multiplo fra 5 e 7 

Dati due numeri interi x e y , calcolare il minimo comune multiplo fra x e y 

Algoritmo

- Sequenza finita di **istruzioni** che specificano come certe **operazioni elementari** debbano susseguirsi nel tempo per risolvere una data **classe di problemi**
- **Classe di problemi:** si intende che la formulazione del problema sia *indipendente* dagli specifici dati su cui opera



Calcolare il minimo comune multiplo fra 5 e 7



Dati due numeri interi x e y , calcolare il minimo comune multiplo fra x e y



Algoritmo ed esecutore

Esempio

Minimo comune multiplo fra x e y

Algoritmo ed esecutore

Esempio

Minimo comune multiplo fra x e y



Algoritmo ed esecutore

Esempio

Minimo comune multiplo fra x e y



- Scomponi x in fattori primi
- Scomponi y in fattori primi
- Moltiplica fra loro tutti i fattori primi comuni e non comuni presi ciascuno una sola volta, con il più grande esponente

Algoritmo ed esecutore

Esempio

Minimo comune multiplo fra x e y



```
graph TD; A[Minimo comune multiplo fra x e y] --> B[• Scomponi x in fattori primi<br/>• Scomponi y in fattori primi<br/>• Moltiplica fra loro tutti i fattori primi comuni e non comuni presi ciascuno una sola volta, con il più grande esponente]; B --> C[ ];
```

- Scomponi x in fattori primi
- Scomponi y in fattori primi
- Moltiplica fra loro tutti i fattori primi comuni e non comuni presi ciascuno una sola volta, con il più grande esponente

Algoritmo ed esecutore

Esempio

Minimo comune multiplo fra x e y

- Scomponi x in fattori primi
- Scomponi y in fattori primi
- Moltiplica fra loro tutti i fattori primi comuni e non comuni presi ciascuno una sola volta, con il più grande esponente



Algoritmo ed esecutore

Esempio

Minimo comune multiplo fra x e y

- Scomponi x in fattori primi
- Scomponi y in fattori primi
- Moltiplica fra loro tutti i fattori primi comuni e non comuni presi ciascuno una sola volta, con il più grande esponente

Studente terza
media

Algoritmo ed esecutore

Esempio

Minimo comune multiplo fra x e y



- Scomponi x in fattori primi
- Scomponi y in fattori primi
- Moltiplica fra loro tutti i fattori primi comuni e non comuni presi ciascuno una sola volta, con il più grande esponente



Studiante terza
media



Algoritmo ed esecutore

Esempio

Minimo comune multiplo fra x e y

- Scomponi x in fattori primi
- Scomponi y in fattori primi
- Moltiplica fra loro tutti i fattori primi comuni e non comuni presi ciascuno una sola volta, con il più grande esponente

Studente prima
elementare

Algoritmo ed esecutore

Esempio

Minimo comune multiplo fra x e y



- Scomponi x in fattori primi
- Scomponi y in fattori primi
- Moltiplica fra loro tutti i fattori primi comuni e non comuni presi ciascuno una sola volta, con il più grande esponente



Studente prima
elementare



Algoritmo ed esecutore

Esempio

Minimo comune multiplo fra x e y

```
#include <iostream>
using namespace std;

int mcd(int x, int y) {
    while (y > 0) {
        int r = x % y;
        x = y;
        y = r;
    }
    return x;
}

int main() {
    int a, b;

    cin >> a;
    cin >> b;

    cout << "Il m.c.m. fra " << a << " e " << b << " e` ";

    if ((a == 0) || (b == 0))
        cout << "0";
    else {
        int m_c_d = mcd(a, b);
        cout << (a * b) / m_c_d;
    }

    cout << endl;
}
```

Studente prima
elementare



Algoritmo ed esecutore

Esempio

Minimo comune multiplo fra x e y

```
#include <iostream>
using namespace std;

int mcd(int x, int y) {
    while (y > 0) {
        int r = x % y;
        x = y;
        y = r;
    }
    return x;
}

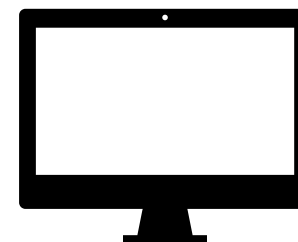
int main() {
    int a, b;

    cin >> a;
    cin >> b;

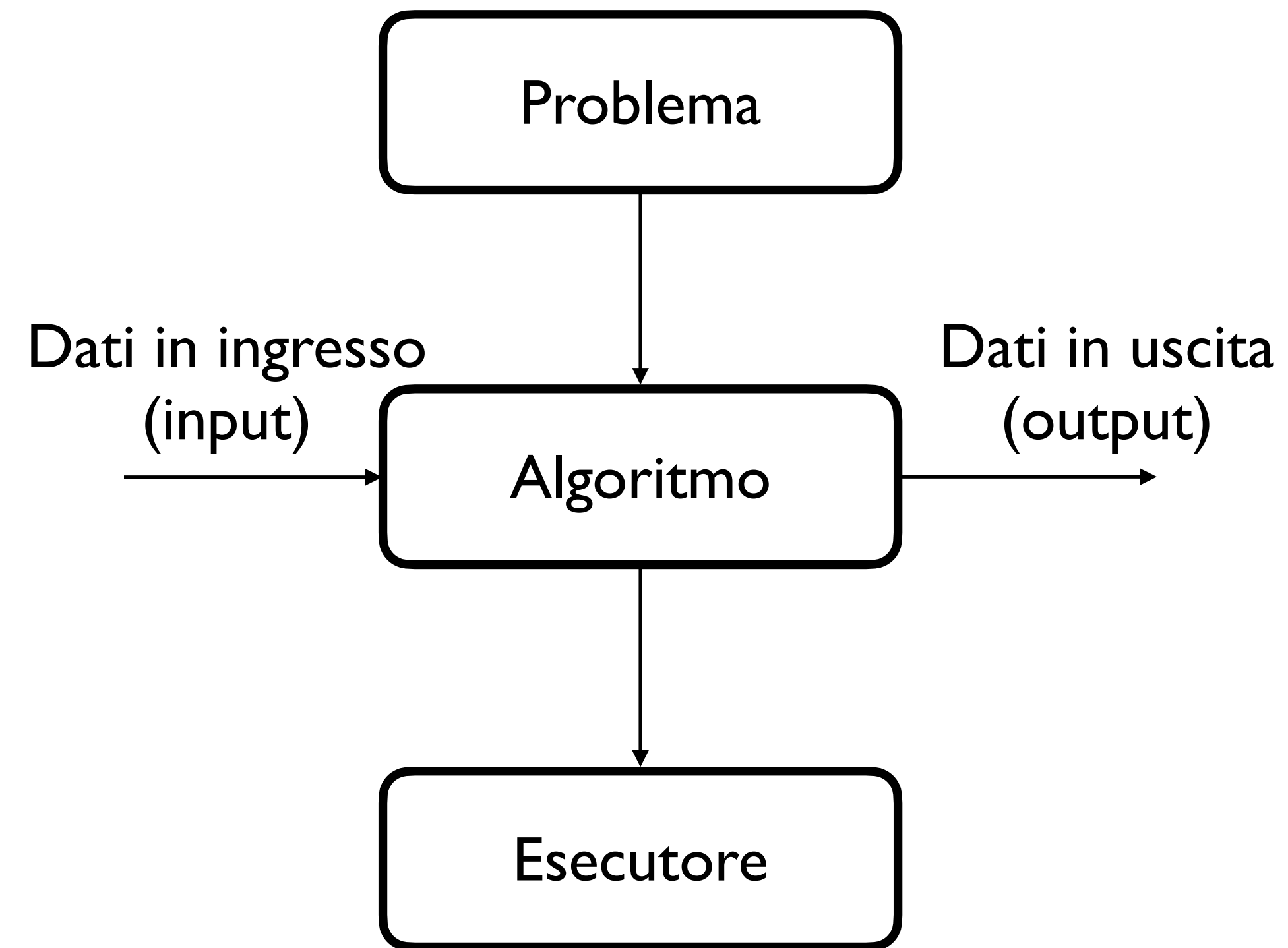
    cout << "Il m.c.m. fra " << a << " e " << b << " e` ";

    if ((a == 0) || (b == 0))
        cout << "0";
    else {
        int m_c_d = mcd(a, b);
        cout << (a * b) / m_c_d;
    }

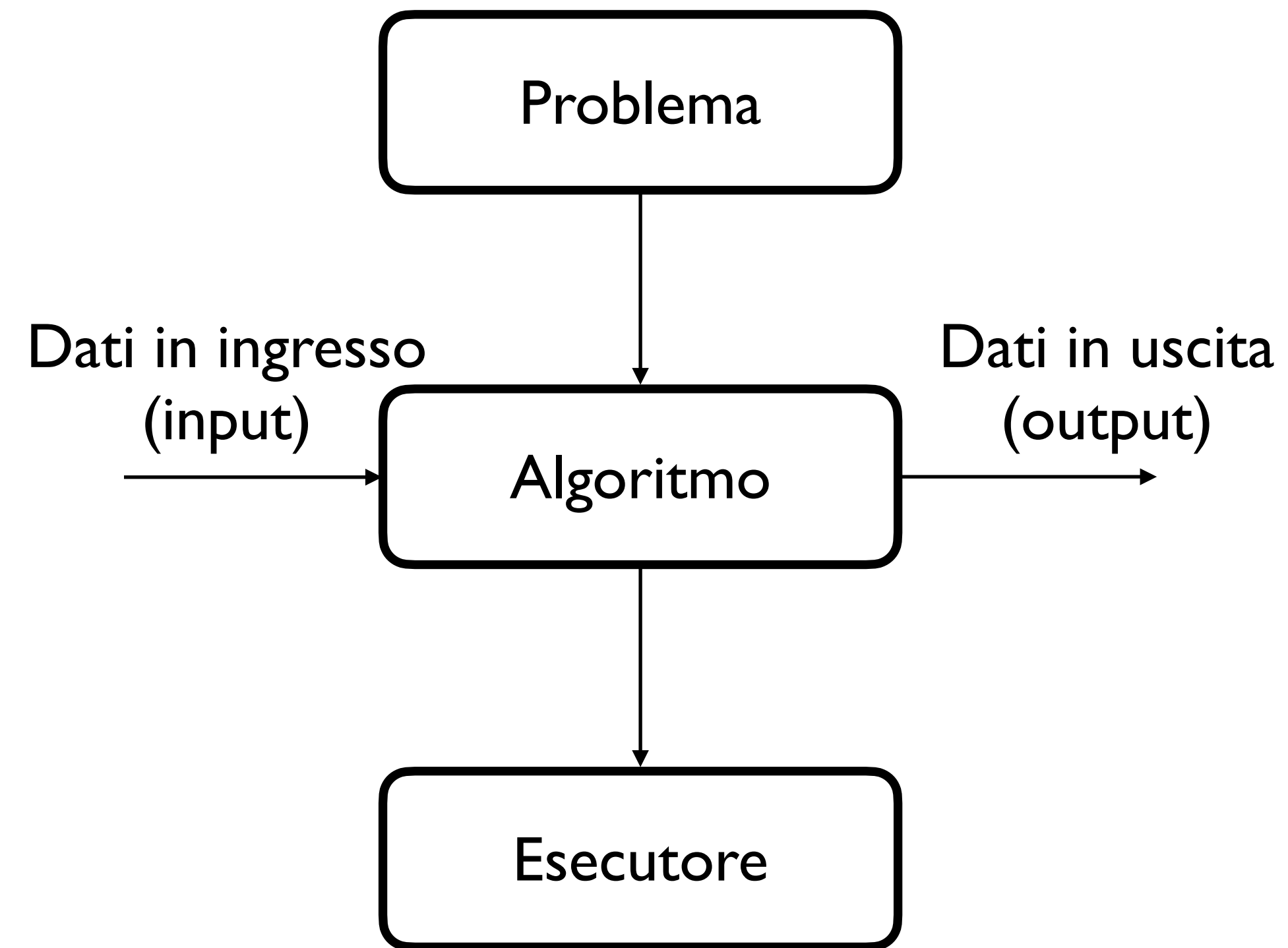
    cout << endl;
}
```



Problemi, algoritmi, esecutori

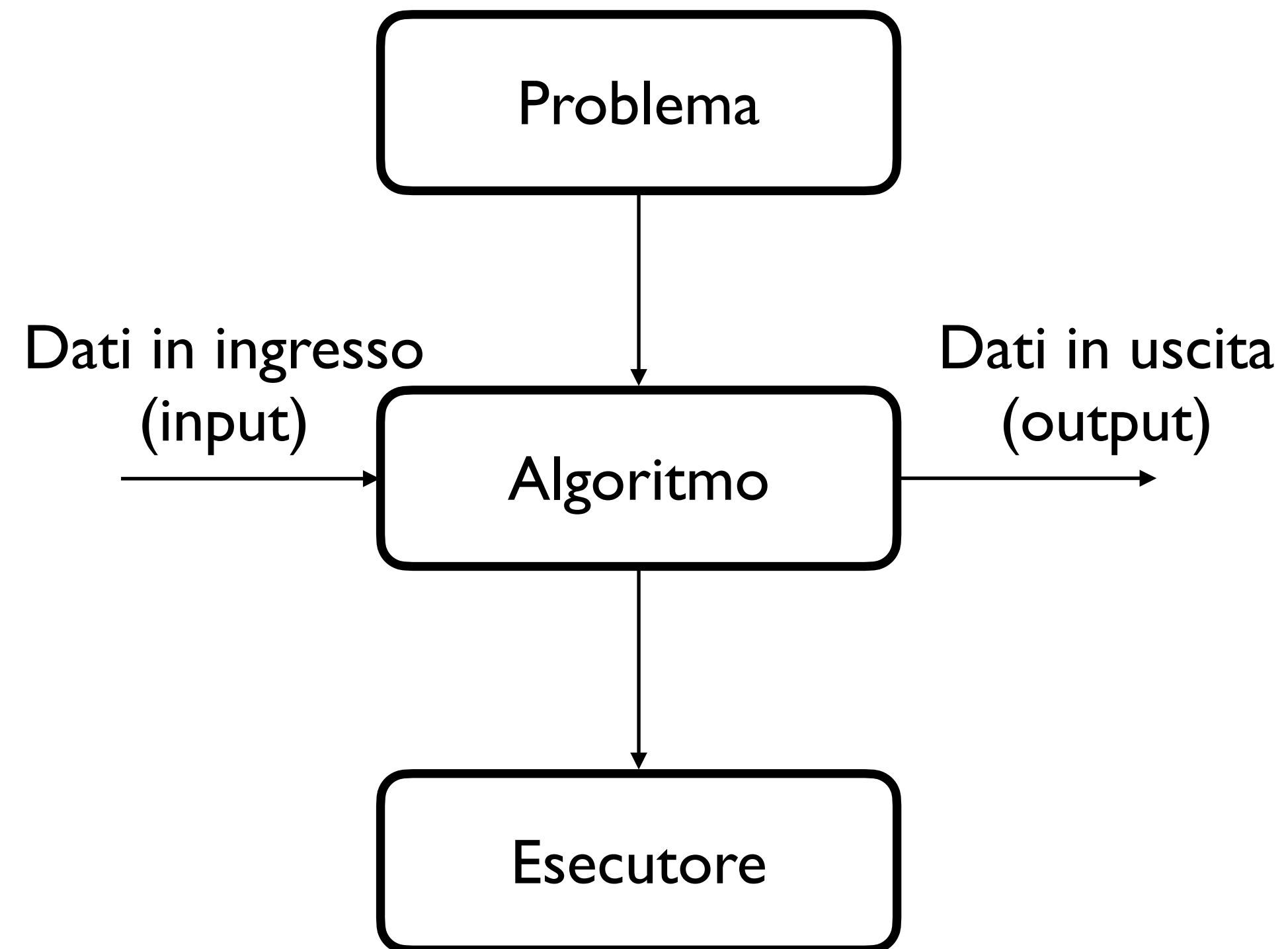


Problemi, algoritmi, esecutori



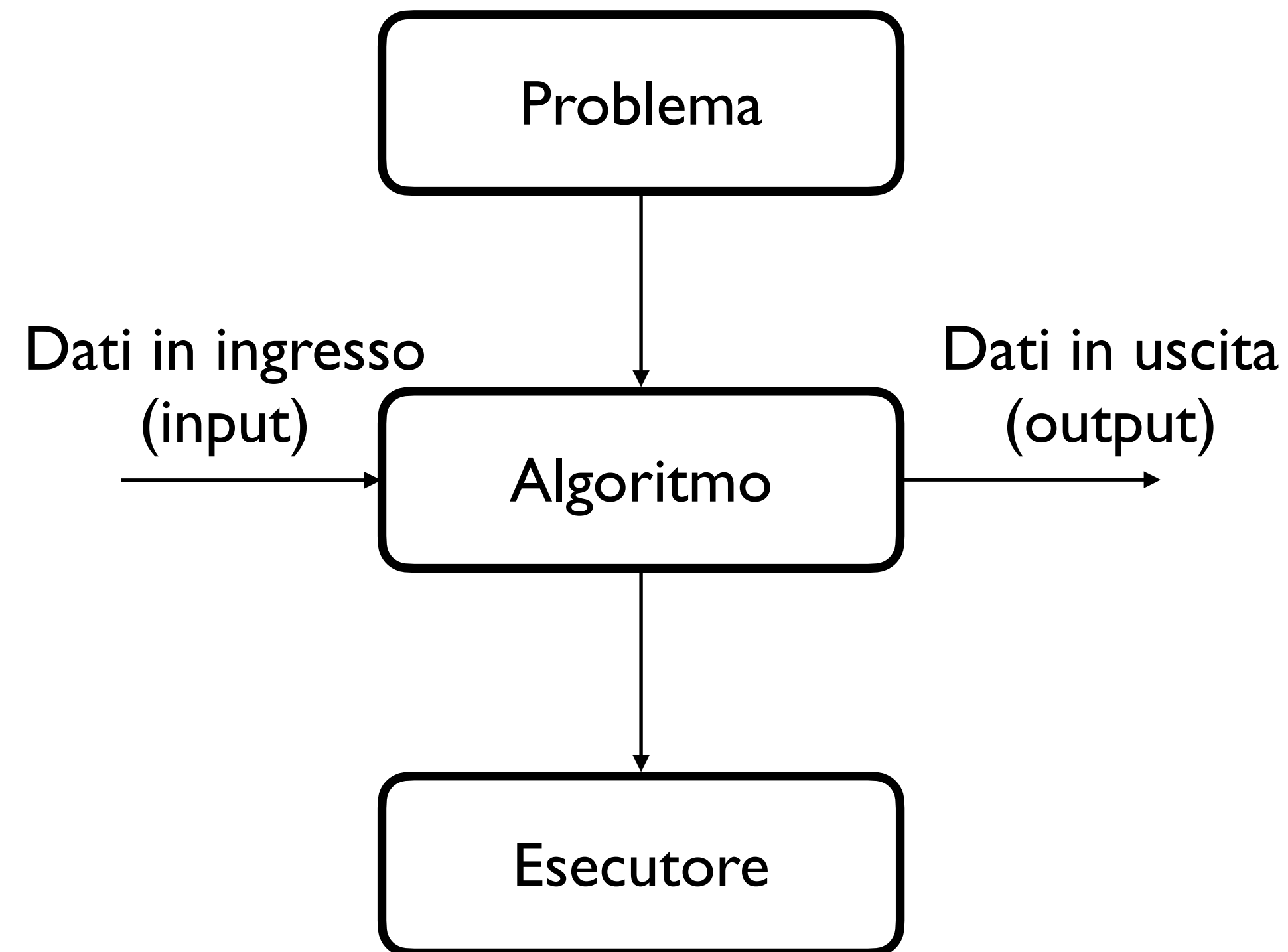
- L'esecutore esegue l'algoritmo in maniera *“acritica”*

Problemi, algoritmi, esecutori



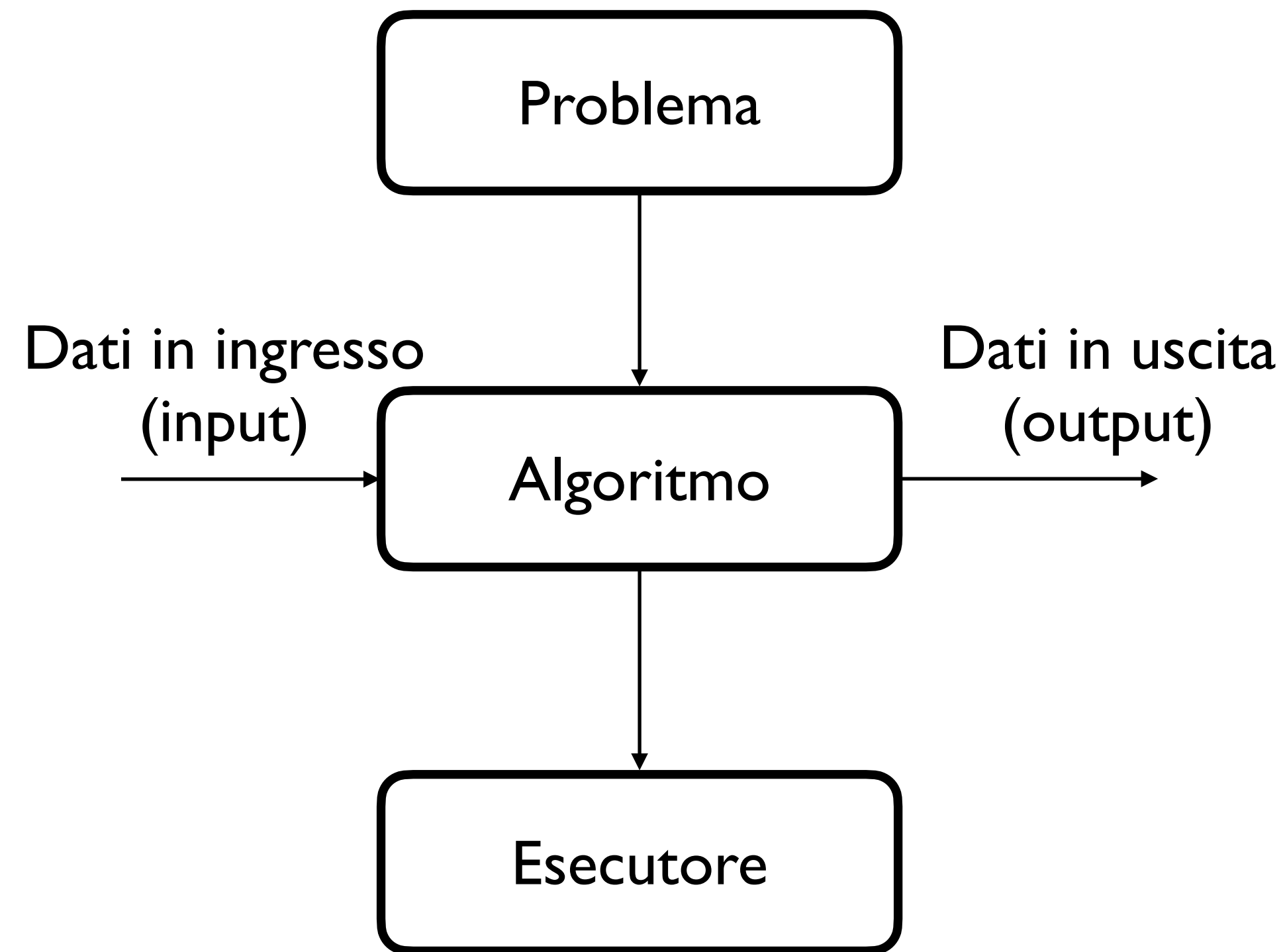
- L'esecutore esegue l'algoritmo in maniera *“acritica”*
- L'algoritmo deve essere **non ambiguo**

Problemi, algoritmi, esecutori



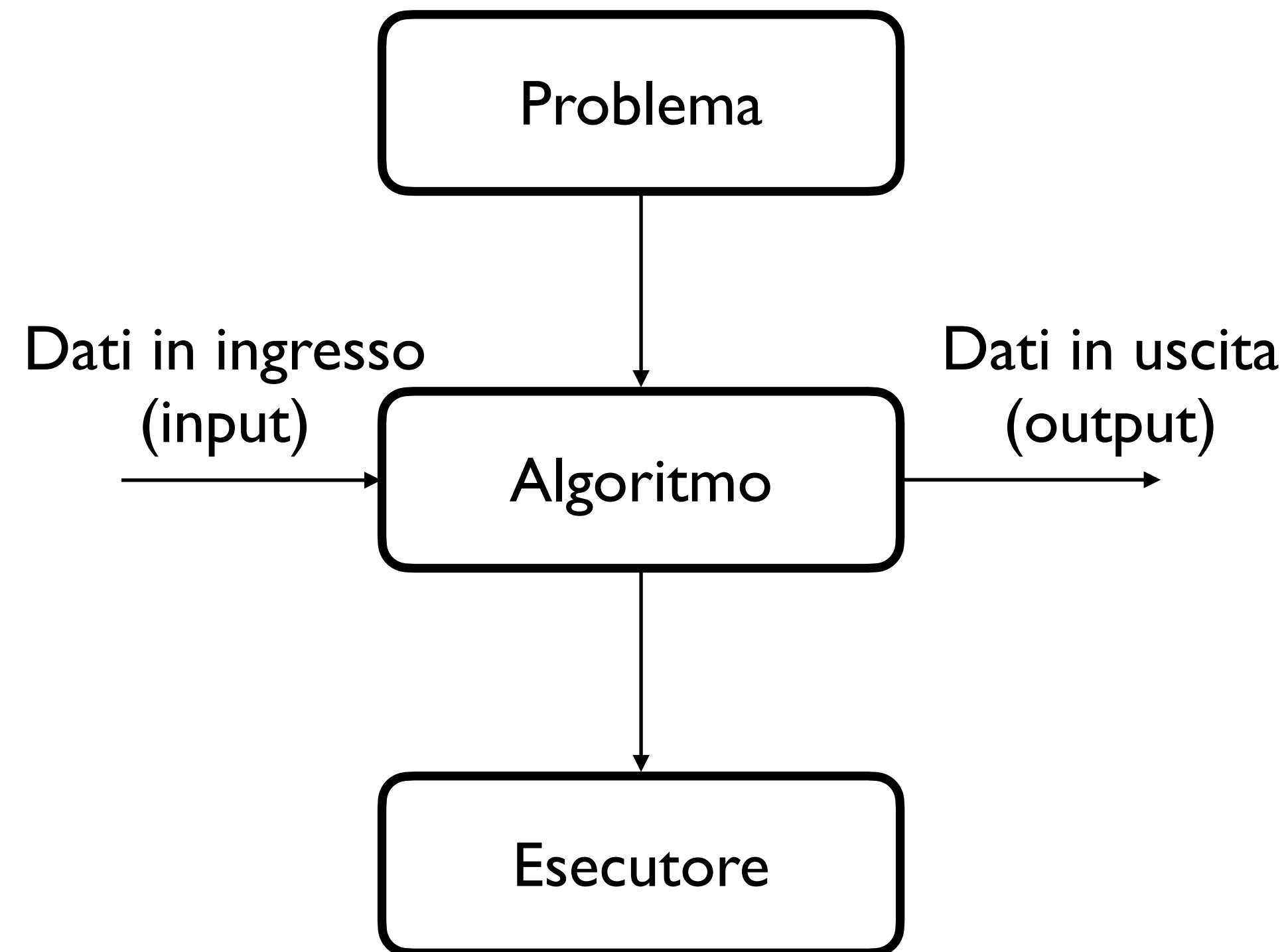
- L'esecutore esegue l'algoritmo in maniera *“acritica”*
- L'algoritmo deve essere **non ambiguo**
 - “aggiungi sale quanto basta”

Problemi, algoritmi, esecutori



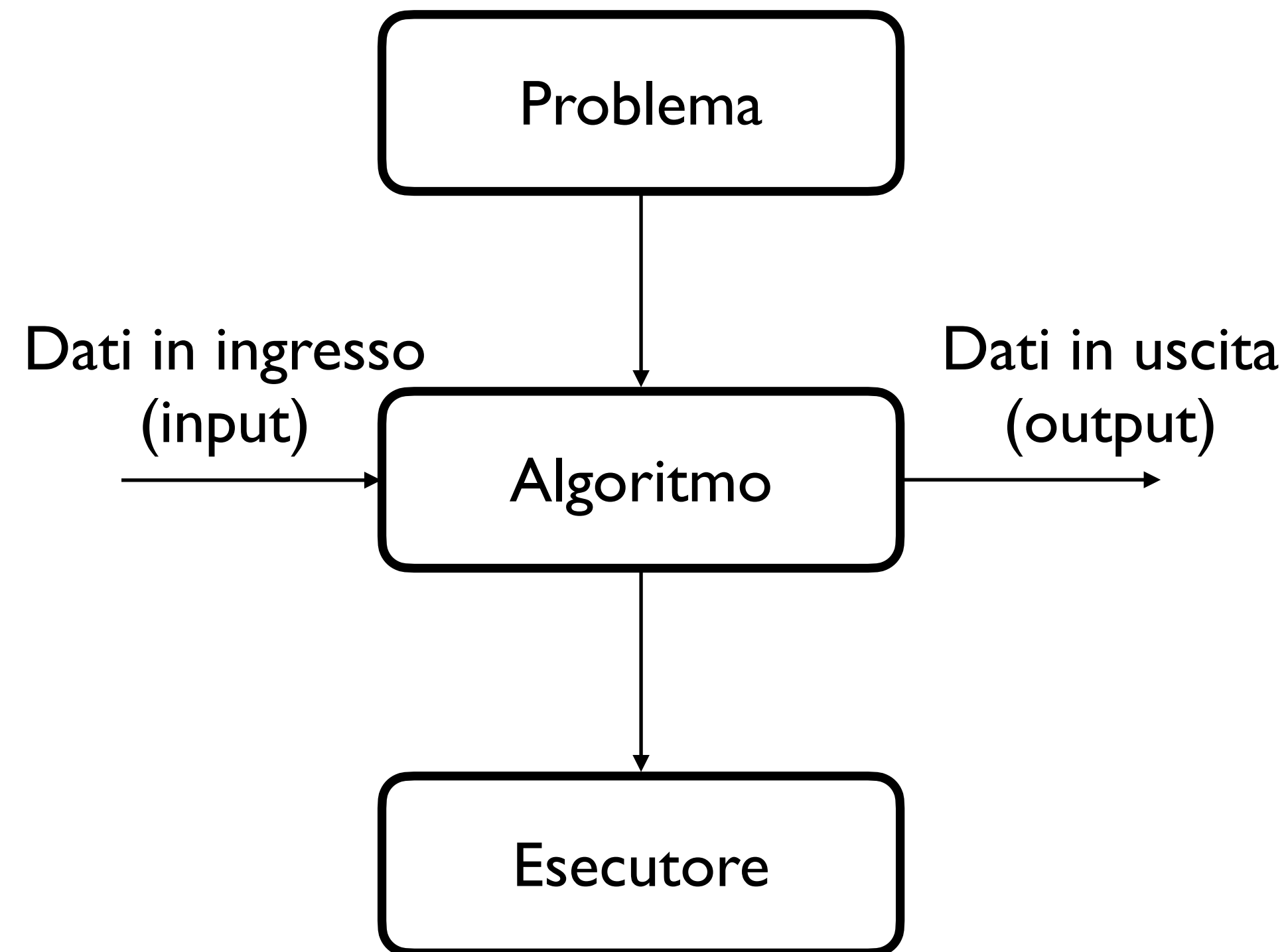
- L'esecutore esegue l'algoritmo in maniera *“acritica”*
- L'algoritmo deve essere **non ambiguo**
 - “aggiungi sale quanto basta” ❌

Problemi, algoritmi, esecutori



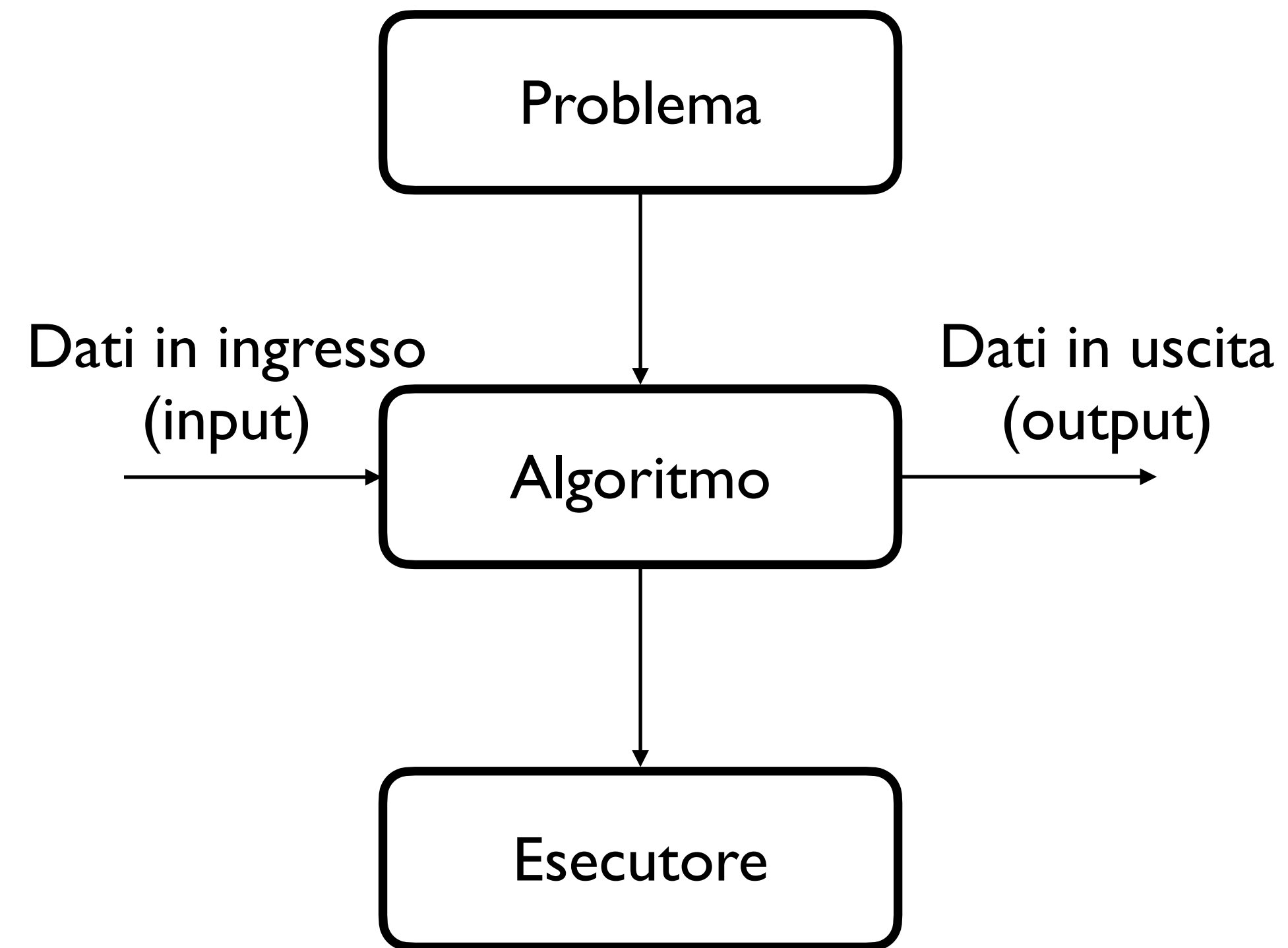
- L'esecutore esegue l'algoritmo in maniera *“acritica”*
- L'algoritmo deve essere **non ambiguo**
 - “aggiungi sale quanto basta” ❌
 - “aggiungi 5 grammi di sale”

Problemi, algoritmi, esecutori



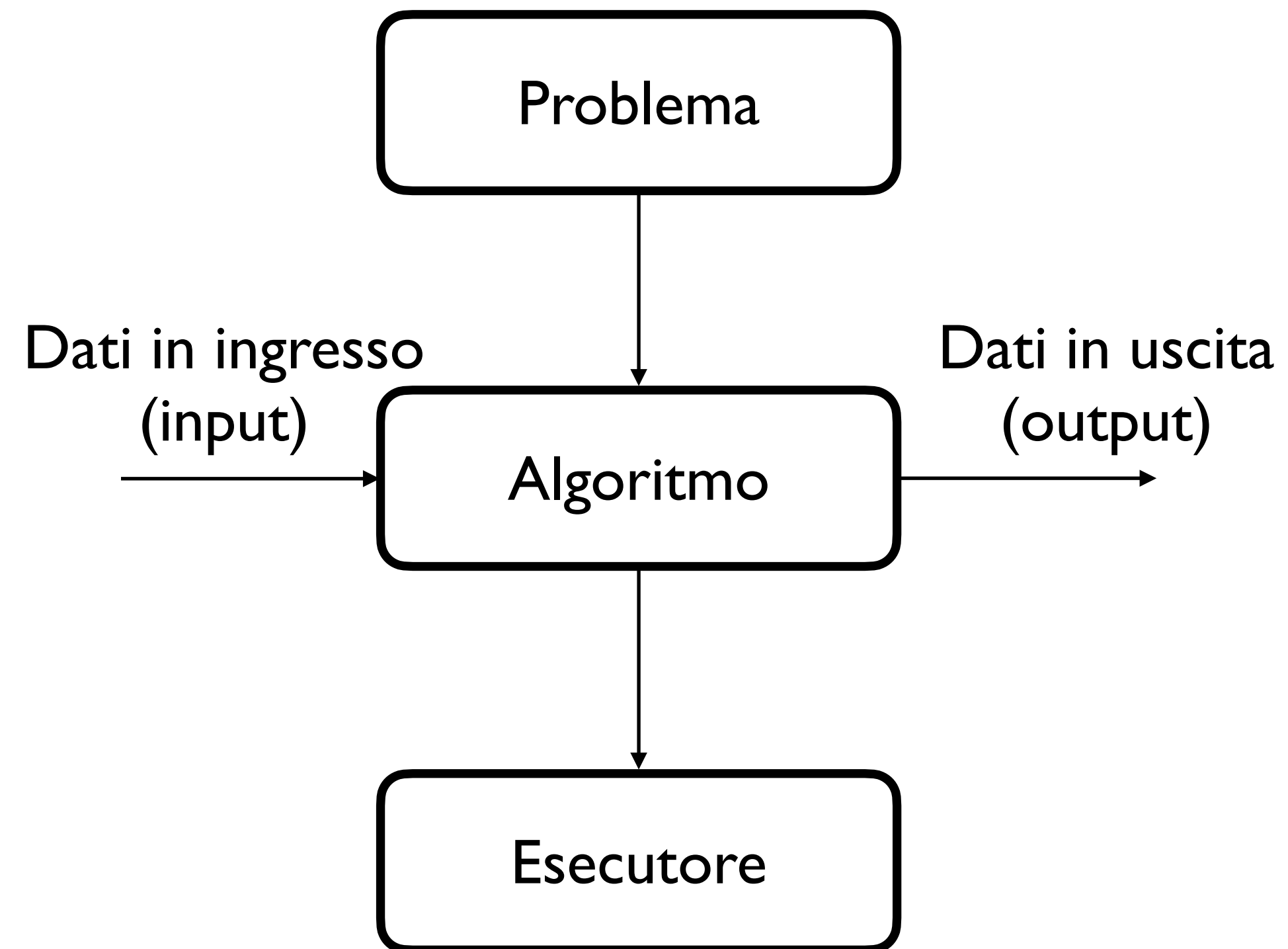
- L'esecutore esegue l'algoritmo in maniera *“acritica”*
- L'algoritmo deve essere **non ambiguo**
 - “aggiungi sale quanto basta” ❌
 - “aggiungi 5 grammi di sale” ✅

Problemi, algoritmi, esecutori

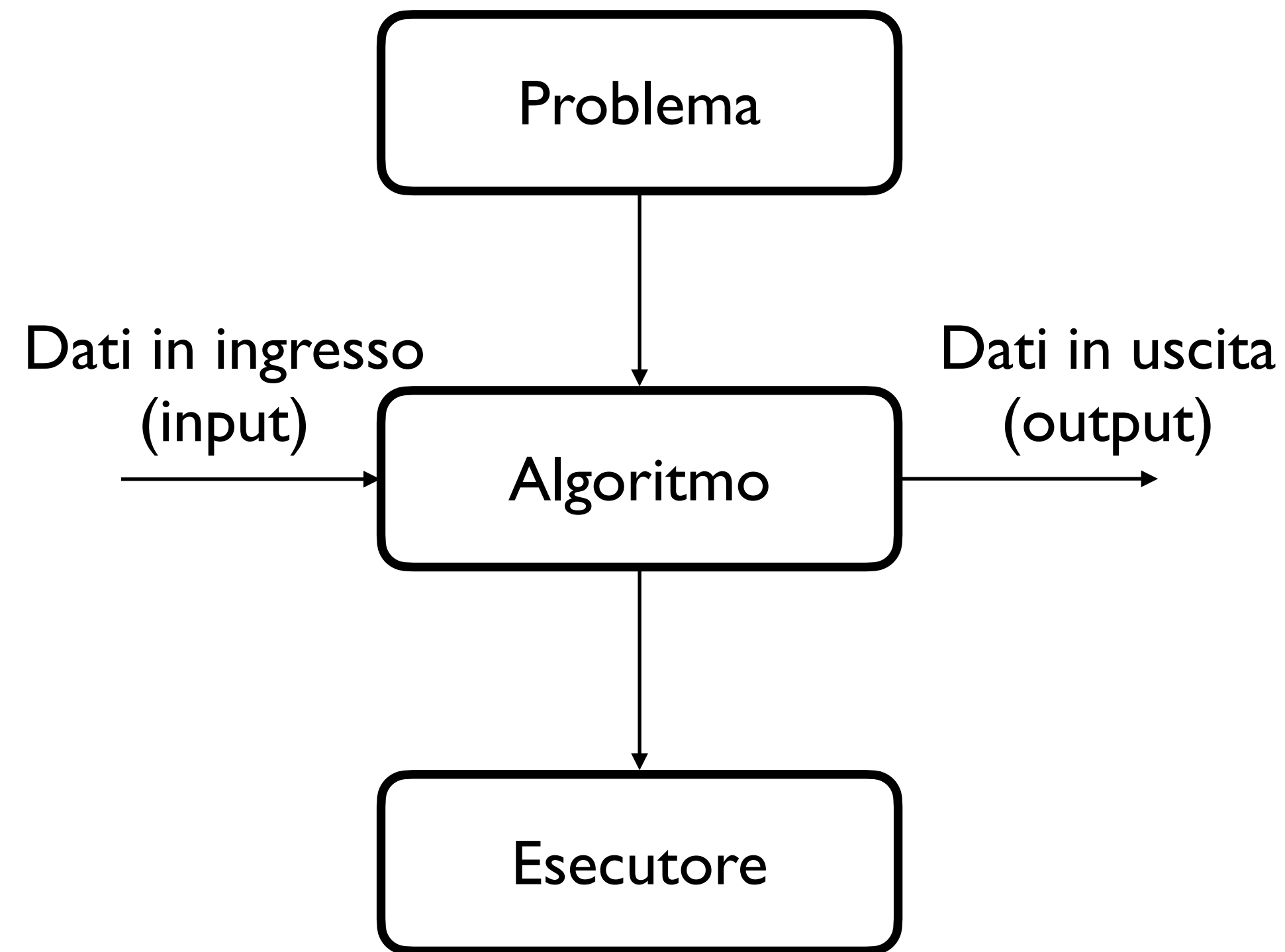


Problemi, algoritmi, esecutori

- Gli algoritmi possono garantire varie proprietà

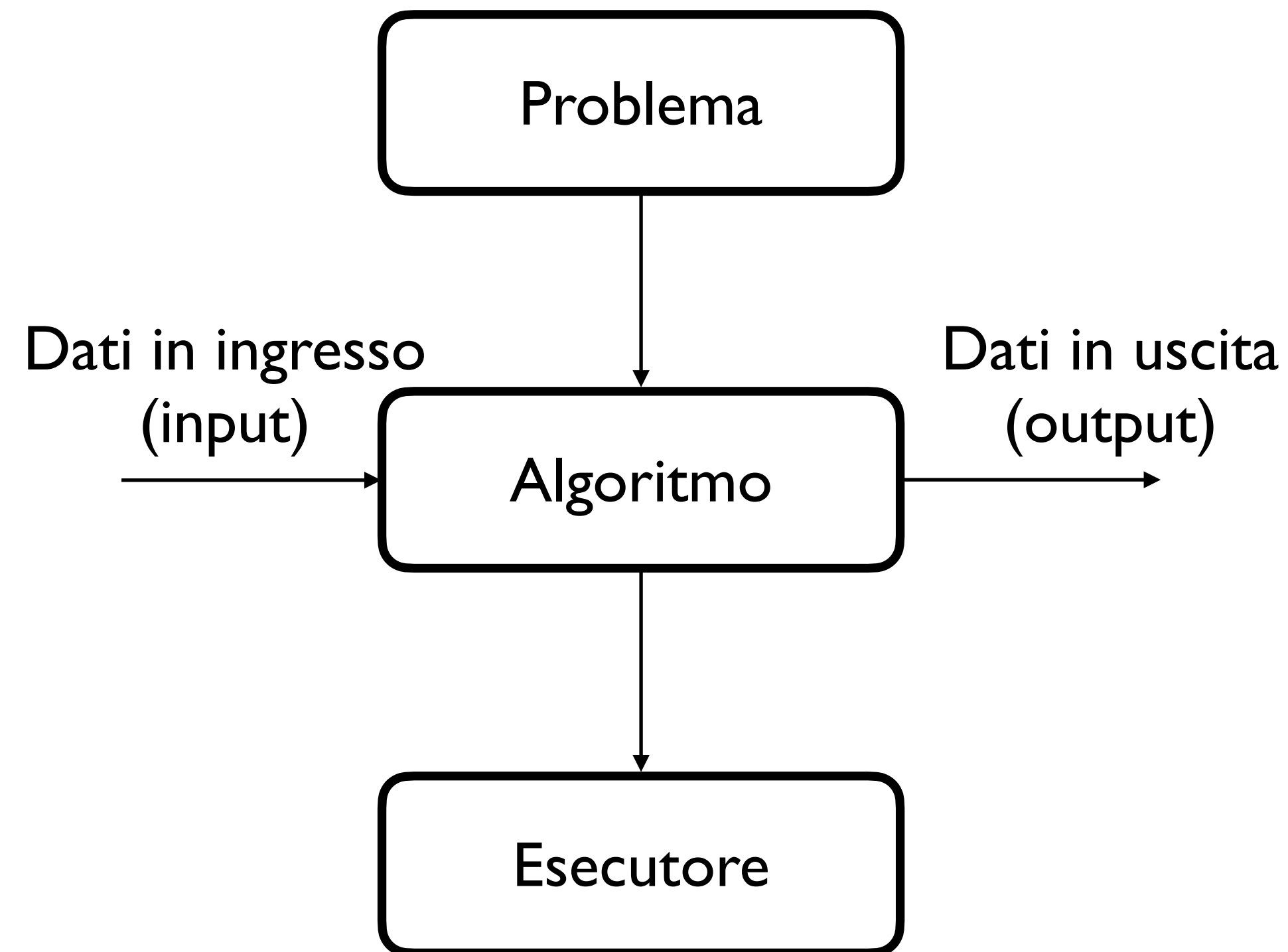


Problemi, algoritmi, esecutori



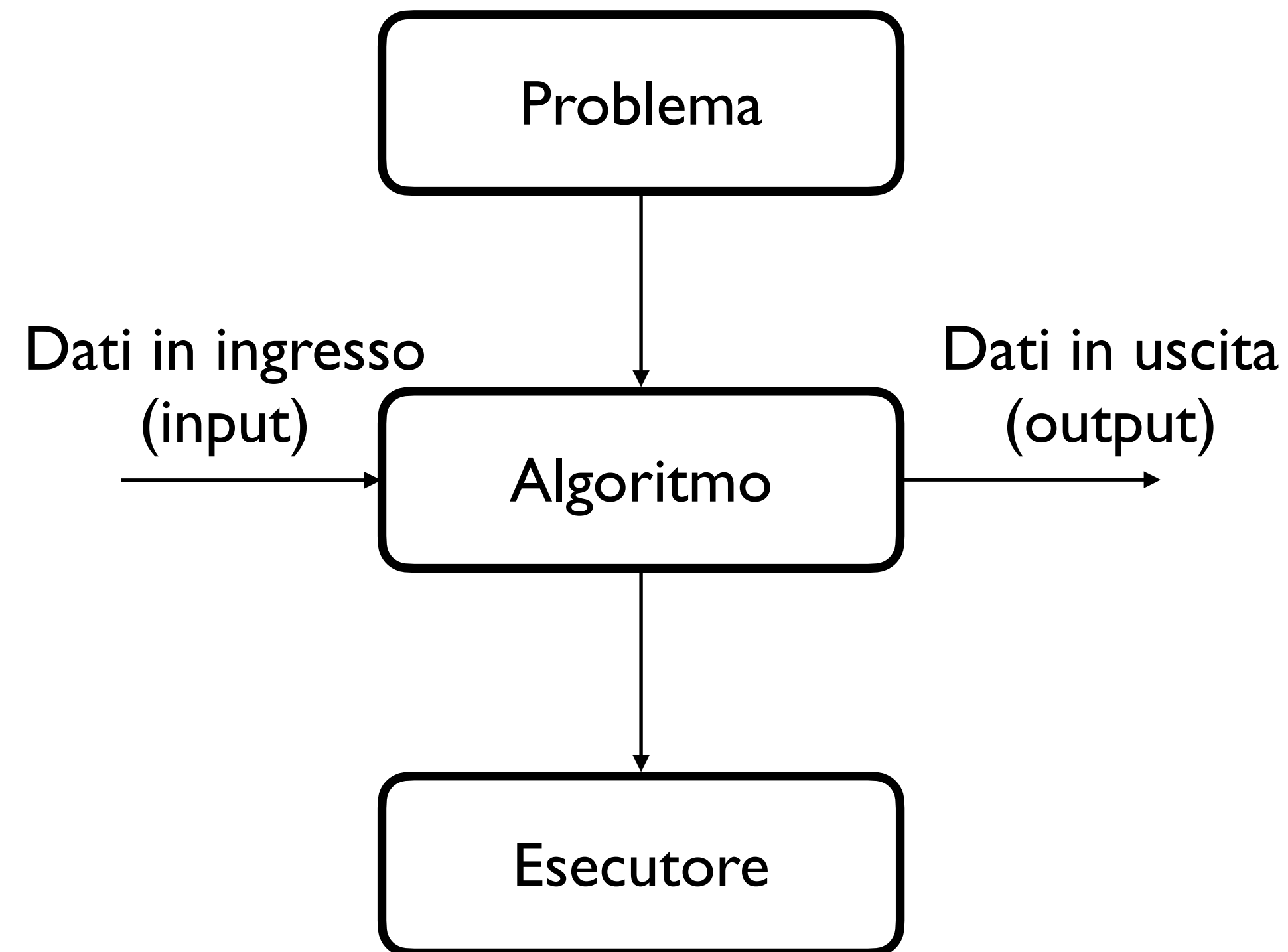
- Gli algoritmi possono garantire varie proprietà
- **Correttezza:** l'esecuzione dell'algoritmo produce i risultati attesi

Problemi, algoritmi, esecutori



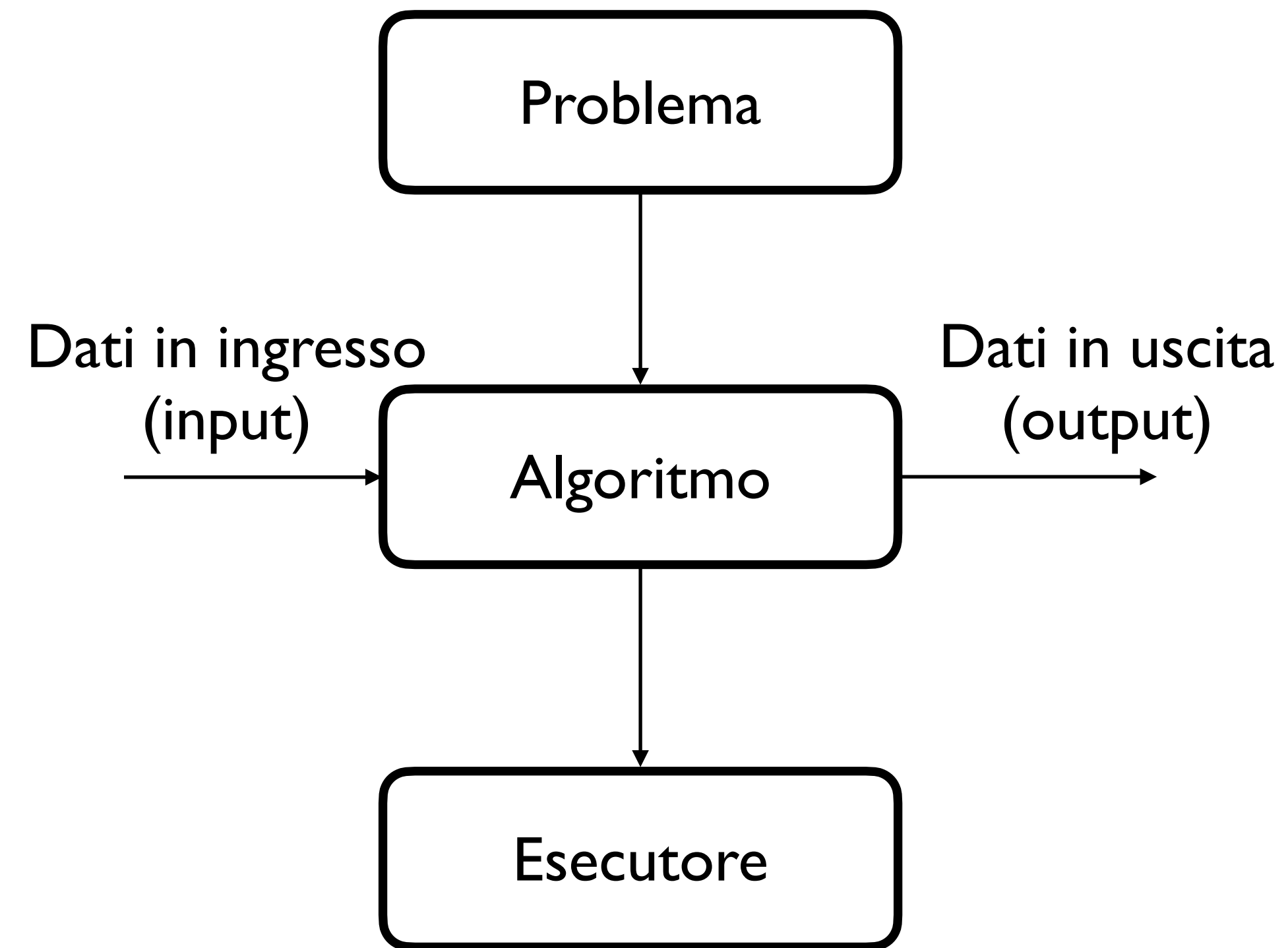
- Gli algoritmi possono garantire varie proprietà
- **Correttezza:** l'esecuzione dell'algoritmo produce i risultati attesi
- **Efficienza:** per esempio, il tempo d'esecuzione

Problemi, algoritmi, esecutori



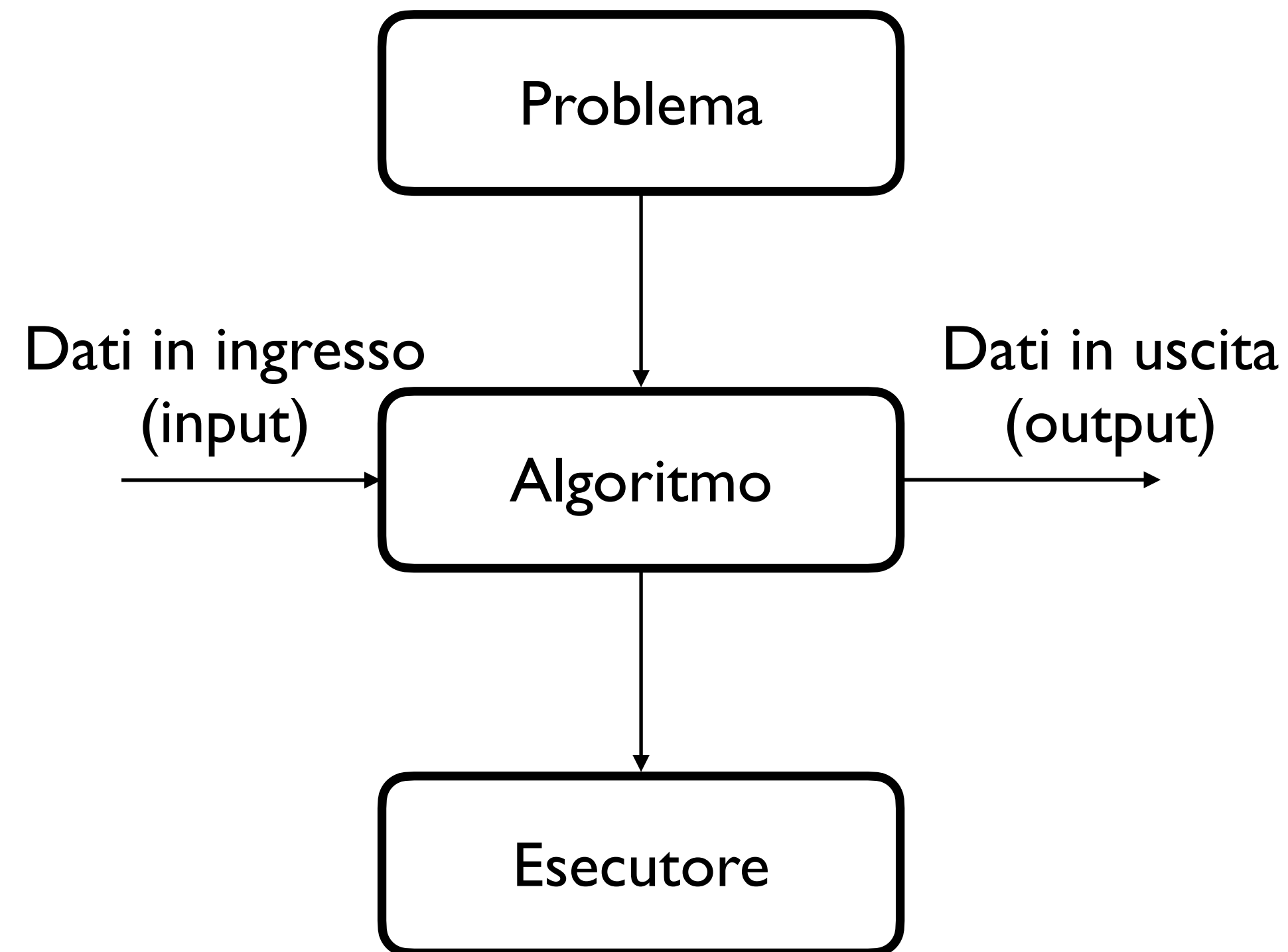
- Gli algoritmi possono garantire varie proprietà
 - **Correttezza:** l'esecuzione dell'algoritmo produce i risultati attesi
 - **Efficienza:** per esempio, il tempo d'esecuzione
 - **Terminazione:** l'algoritmo termina in un tempo finito

Problemi, algoritmi, esecutori

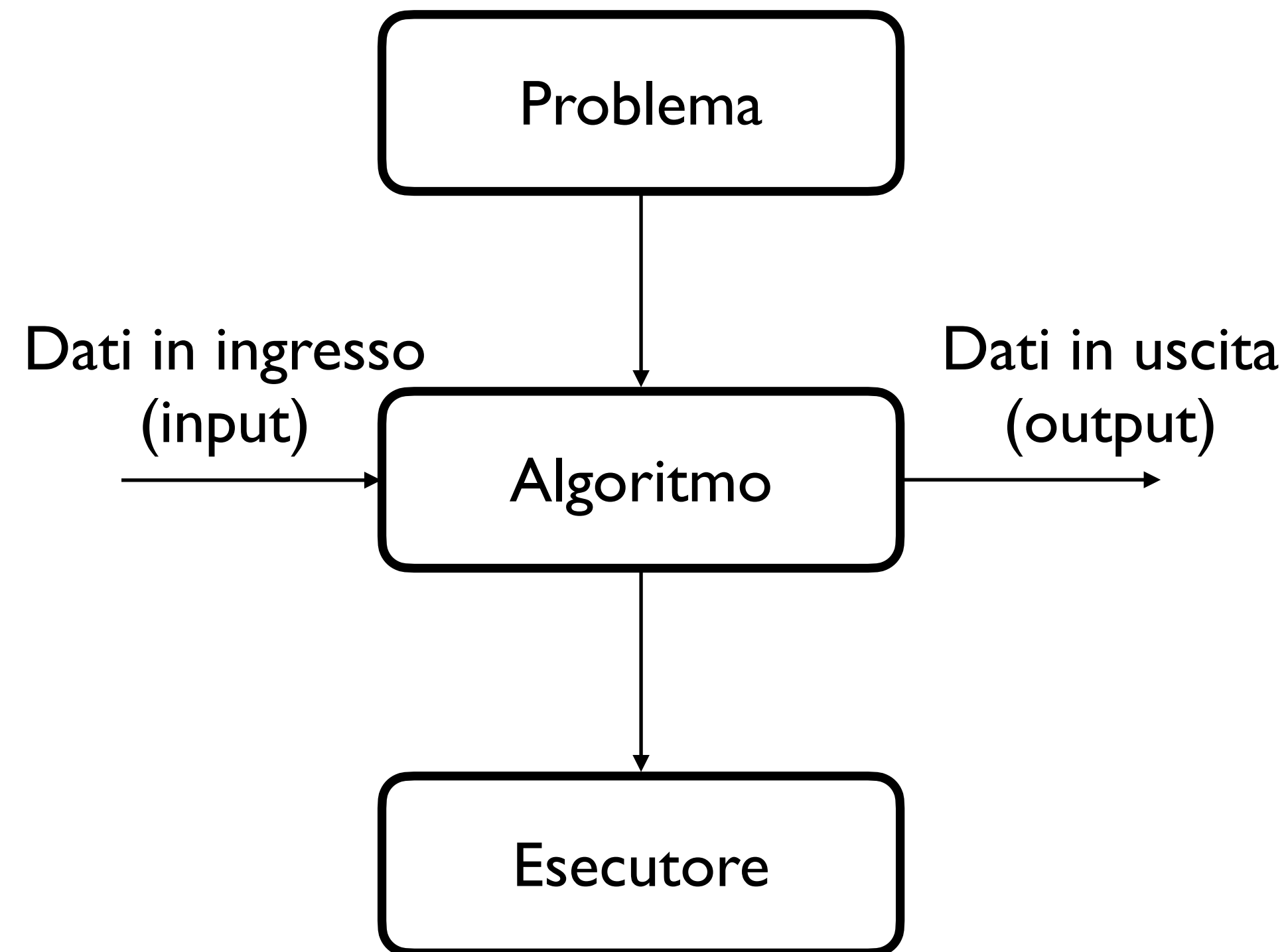


Problemi, algoritmi, esecutori

- Dato un problema, quanti sono gli algoritmi che risolvono quel problema?

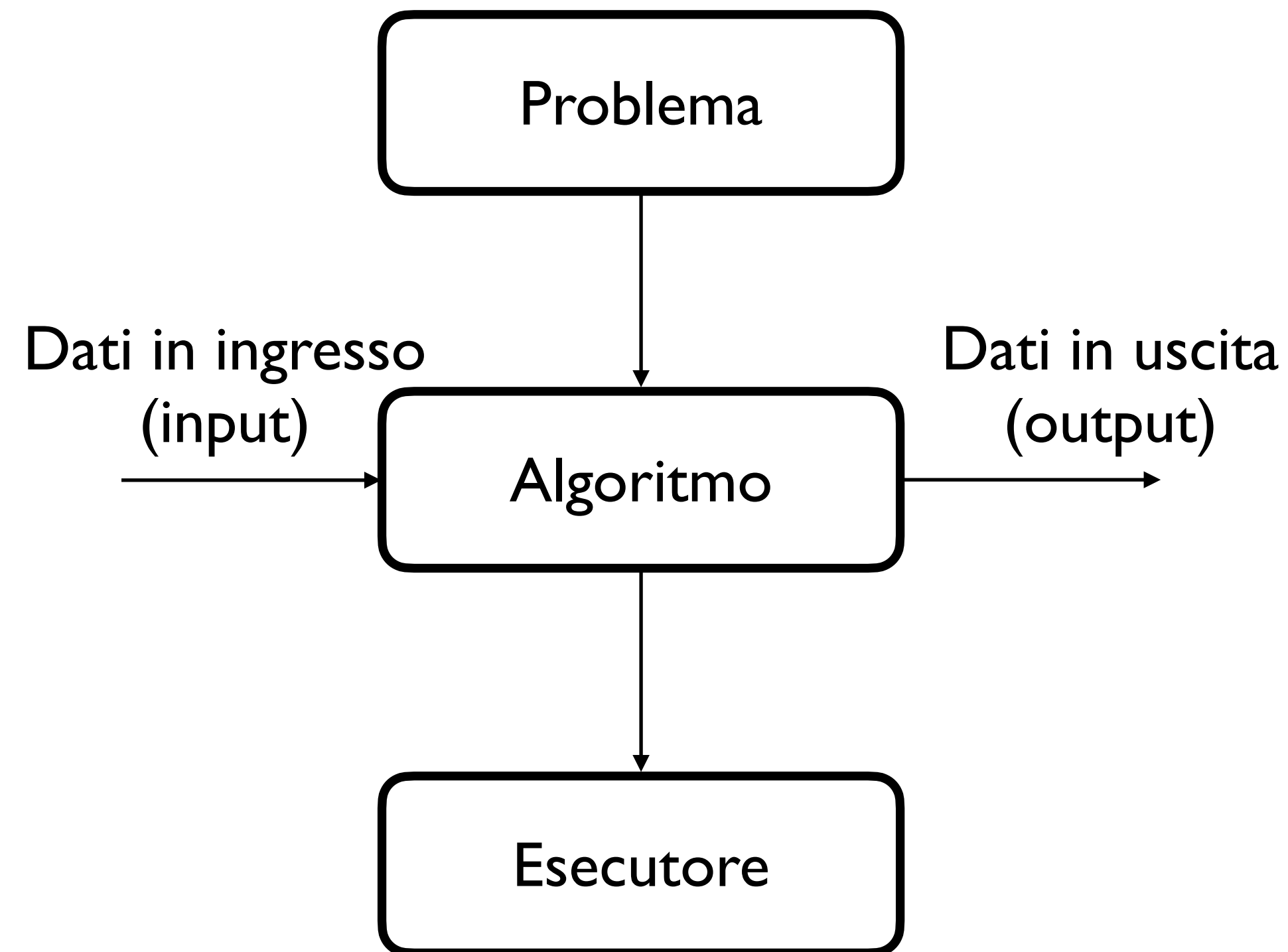


Problemi, algoritmi, esecutori



- Dato un problema, quanti sono gli algoritmi che risolvono quel problema?
- Dato un problema, esiste sempre un algoritmo che lo risolve?

Problemi, algoritmi, esecutori



- Dato un problema, quanti sono gli algoritmi che risolvono quel problema?
- Dato un problema, esiste sempre un algoritmo che lo risolve?
- Dato un problema “*risolvibile*”, qual è l’algoritmo “*più efficiente*” (in termini di tempo, risorse utilizzate, utilizzo della rete)?

Algoritmi

Linguaggio di descrizione

Algoritmi

Linguaggio di descrizione

- Formalismo costituito da un'insieme di **istruzioni primitive** (elementi propri del linguaggio), un'insieme di **tipi di dato** che il linguaggio può manipolare (interi, reali, caratteri,...) un'insieme di **operazioni primitive** su tali dati

Algoritmi

Linguaggio di descrizione

- Formalismo costituito da un'insieme di **istruzioni primitive** (elementi propri del linguaggio), un'insieme di **tipi di dato** che il linguaggio può manipolare (interi, reali, caratteri,...) un'insieme di **operazioni primitive** su tali dati
- Linguaggio naturale strutturato

Algoritmi

Linguaggio di descrizione

- Formalismo costituito da un'insieme di **istruzioni primitive** (elementi propri del linguaggio), un'insieme di **tipi di dato** che il linguaggio può manipolare (interi, reali, caratteri,...) un'insieme di **operazioni primitive** su tali dati
- Linguaggio naturale strutturato
- Linguaggi grafici: diagrammi di flusso (flow chart), UML (Unified Modeling Language)

Algoritmi

Linguaggio di descrizione

- Formalismo costituito da un'insieme di **istruzioni primitive** (elementi propri del linguaggio), un'insieme di **tipi di dato** che il linguaggio può manipolare (interi, reali, caratteri,...) un'insieme di **operazioni primitive** su tali dati
 - Linguaggio naturale strutturato
 - Linguaggi grafici: diagrammi di flusso (flow chart), UML (Unified Modeling Language)
 - Linguaggi di programmazione

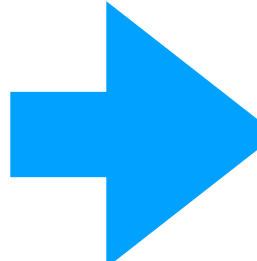
Algoritmi

Linguaggio di descrizione

- Formalismo costituito da un'insieme di **istruzioni primitive** (elementi propri del linguaggio), un'insieme di **tipi di dato** che il linguaggio può manipolare (interi, reali, caratteri,...) un'insieme di **operazioni primitive** su tali dati
 - Linguaggio naturale strutturato
 - Linguaggi grafici: diagrammi di flusso (flow chart), UML (Unified Modeling Language)
 - Linguaggi di programmazione
- I primi due sono pensati per un esecutore umano, gli ultimi sono pensati per esecutori automatici

Algoritmi

Linguaggio di descrizione

- Formalismo costituito da un'insieme di **istruzioni primitive** (elementi propri del linguaggio), un'insieme di **tipi di dato** che il linguaggio può manipolare (interi, reali, caratteri,...) un'insieme di **operazioni primitive** su tali dati
 - Linguaggio naturale strutturato
 -  • Linguaggi grafici: diagrammi di flusso (flow chart), UML (Unified Modeling Language)
 - Linguaggi di programmazione
- I primi due sono pensati per un esecutore umano, gli ultimi sono pensati per esecutori automatici

