PROBLEM SOLVING = disciplina che si occupa di trovare procedimenti generali per risolvere problemi. Ovvero l'insieme delle competenze/abilità necessarie per portare a termine con successo la soluzione di un problema (generico), cioè quelle abilità che consentono di individuare un algoritmo trattabile e quindi un procedimento programmabile per il computer.

Quindi:

Dato un problema generico —obiettivo→ trovare un procedimento applicabile per poter trovare una soluzione

= non è il risultato del prodotto dell'applicazione del programma MA è il programma stesso

PROCEDIMENTO RISOLUTIVO = elenco delle formule che devono essere applicate per trovare la soluzione del problema

STATO DI PARTENZA	STATI INTERMEDI	STATO D'ARRIVO
= dati iniziali o ipotesi (viene formulato un problema da risolvere e vengono formulate tutte le premesse e i dati iniziali)	= le situazioni alle quali si perviene via via applicando le formule opportune o regole di deduzione problema in sottoproblemi	= raggiunto il quale il problema è risolto (ed è disponibile la soluzione).

RISOLUBILITA' DEI PROBLEMI = non tutti i problemi possono essere risolti da un computer

PROCEDURA EFFETTIVA = L'attività di cui si parla deve essere eseguibile senza alcuna ambiguità

PROBLEMA ALGORITMICO:

Requisiti fondamentali:
- l'insieme degli input legali per quel problema deve essere definito in modo effettivo
- l'output deve essere funzione dell'input

Il problema può essere anche di tipo decisionale (=verifica di particolari proprietà), in tal caso l'output può essere SI o NO a seconda che il particolare input soddisfi o meno quelle proprietà. In tal caso di parla di problemi DECIDIBILI o NON DECIDIBILI.

Una funzione per la quale esiste un algoritmo per calcolarne i valori si dice CALCOLABILE (in caso contrario si dice NON CALCOLABILE); i problemi si dicono COMPUTABILI o NON COMPUTABILI.

OGNI INPUT → istanza del problema

RISOLUBILE
O

→ Quando si esplicita un procedimento effettivo (=eseguibile meccanicamente) che risolve qualsiasi istanza del problema (in un tempo finito)

Tutti i problemi algoritmici con input finiti sono computabili, mentre è più problematico per i problemi algoritmici che prevedono un numero infinito di input ammissibili in tal caso si parla di problemi NON COMPUTABILI

ALGORITMO → un procedimento che risulti effettivo in ogni sua istanza (cioè una procedura finita che in tempo finito sa risolvere una qualsiasi delle possibili istanze del problema). Quando non è possibile trovare procedimenti effettivi che risolvano qualsiasi istanza si parla di INDECIDIBILITA'

La descrizione di un procedimento con le seguenti caratteristiche:
- la descrizione deve essere esplicita e non ambigua per l'interlocutore al quale è destinata
- il procedimento che viene attivato per ogni istanza, seguendo il testo della descrizione deve terminare in un tempo finito

Si presuppone quindi sempre la presenza di :

2 interlocutori (chi produce la sequenza /e chi sa eseguire tutte le operazioni =computer)
un linguaggio usato per la descrizione (=un linguaggio di programmazione)

In questo caso l'algoritmo (cioè la descrizione del procedimento) assume la forma di PROGRAMMA

1. Touring : ha definito un meccanismo detto macchina di Turino, attraverso il quale fu possibile iscrivere e cancellare dei dati su un nastro
2. Church : ha proposto il lamda – dati
3. Prost : ha definito un meccanismo di manipolazione simbolica che si chiama regole di produzione
4. Kleene e Goedel : hanno proposto la classe delle funzioni ricorsive primitive

Tutti questi meccanismi si sono dimostrati equipotenti, ovvero un problema che risultava computabile con un meccanismo, risultava esserlo anche con gli altri.

TESI DI CHURCH = un problema è risolubile, in linea di principio, se e solo se esiste un procedimento effettivo descrivibile con un linguaggio di programmazione.

ATTENZIONE: ogni algoritmo può essere formulato come programma per computer, ma non è vero che ogni programma per computer sia la descrizione di un algoritmo

FORMULARE UN PROBLEMA IN MODO ALGORITMICO:

- Descrivere tutti gli INPUT ammissibili;
- Descrivere l'OUTPUT attesi per ciascun Input.

Per fare ciò è necessario eseguire le seguenti attività:

1. individuare nel testo le entità coinvolte in input e output
2. definire per ciascuna entità l'insieme da cui possono prendere i rispettivi valori;
3. esplicitare la o le relazioni tra le entità di input o output

TEORIA DELLA CALCOLABILITA' = definisce i criteri che consentono di ritenere effettivamente risolubile un problema, un problema viene definito effettivamente risolubile se è possibile eseguire (per esempio)§ una macchina di Touring (o un programma per computer) che lo risolva

TEORIA DELLA COMPLESSITA' = l'algoritmo individuato deve essere computazionalmente trattabile. Quando il tempo di soluzione di un problema dipende da un parametro n (detto dimensione del problema generico) può accadere che il numero delle operazioni da eseguire per trovare la soluzione mostri una crescita (rispetto a n) come di seguito:

- LINEARE : trovare il più grande di n numeri dati
- QUADRATICA : mettere in ordine crescente n numeri dati
- CUBICA : risolvere un sistema di equazioni lineari
- ESPONENZIALE : elencare i sottoinsiemi di un insieme di n elementi
- SUPERESPONENZIALE : elencare le permutazioni di n oggetti

SE UN PROBLEMA GENERICO PREVEDE UN NUMERO DI OPERAZIONI CHE CRESCE ESPONENZIALMENTE CON LA DIMENSIONE DELL'INPUT, IN GENERALE QUEL PROBLEMA NON E' TRATTABILE, MA POSSONO ESSERE RISOLTI SOLO CASI ASSOCIATI A DIMENSIONI ESTREMAMENTE LIMITATI.

QUINDI:

Affinché abbia senso risolvere un problema facendo ricorso un computer , non solo il problema deve essere risolubile in termini di principio (cioè calcolabile), ma deve anche essere trattabile, cioè si deve conoscere per quel problema un procedimento risolutivo (cioè un algoritmo) che richieda un tempo di esecuzione ragionevole (non esponenziale o superesponenziale)

RIASSUMENDO:

Quanto detto su calcolabilità e complessità dei problemi trattati si possono così riassumere:

- Il problema deve essere formulato in modo algoritmico (specificando le entità di input quelle di output e la relazione che le lega)

- Deve essere un algoritmo per la soluzione di tutte le istanze ammissibili di quel problema
- Il numero di operazioni da eseguire per ottenere la soluzione delle singole istanze non deve crescere in modo esponenziale o superesponenziale con il crescere delle dimensioni dell'input (e quindi non è sufficiente esibire un algoritmo e dimostrare che il procedimento ha termine)

Una volta trovato il procedimento effettivo e ottenuti i risultati, si devono sempre verificare due cose:

- Che il procedimento sia effettivamente corretto rispetto al problema (algoritmico) che si sta trattando
- Che la soluzione del problema algoritmico sia anche soddisfacente soluzione delle esigenze che lo hanno prodotto

PROGRAMMI TRADUTTORI o INTERPRETI o COMPILATORI sono programmi che eseguono traduzioni tra testi scritti in linguaggi di programmazione, esso è caratterizzato da tre linguaggi:

1. SORGENTE , il linguaggio usato per descrivere il messaggio da tradurre
2. OGGETTO , il linguaggio usato per fornire la traduzione
3. il linguaggio in cui è scritto il programma che effettua la traduzione

I computer quando escono dalla catena di montaggio hanno il linguaggio macchina.

Possiamo distinguere due famiglie di linguaggi di programmazione:

- a basso livello (o linguaggio macchina) per facilitare e rendere economica la produzione di computer
- a alto livello per facilitare la scrittura di algoritmi nelle diverse aree disciplinari e rendere efficienti i processi di elaborazione

Il problem solving utilizza linguaggi ad alto livello, che possono essere distinti in 4 categorie:

1. linguaggi procedurali, consente di fare riferimento ad un insieme di azioni elementari (primitive) che l'esecutore sa risolvere
2. linguaggi dichiarativi, consente di fare riferimento al un insieme di problemi elementari (primitivi) che l'esecutore sa risolvere
3. linguaggi funzionali
4. linguaggi ad oggetti

Le due alternative nella scelta di linguaggio da utilizzare nel problem solvine possono essere il linguaggio procedurale e quello dichiarativo

LINGUAGGIO NATURALE $\neq$ LINGUAGGIO DI PROGRAMMAZIONE

Perché le frasi scritte in linguaggio di programmazione hanno sempre e solo un significato, mentre questo in genere non è vero per le frasi (corrette) scritte in un linguaggio naturale (il che rende impossibile in linea di principio la traduzione di testi scritti in linguaggi naturali)

PROBLEMI PRIMITIVI = problemi che l'interlocutore sa risolvere direttamente, sono elementi tipici del linguaggio dichiarativo

AZIONI PRIMITIVE = quelle azioni che si sanno descrivere con frasi elementari del linguaggio, sono elementi tipici dei linguaggi procedurali

Sebbene le azioni primitive siano il corrispondente operativo dei problemi primitivi, l'uso delle une o degli altri cambia radicalmente la metodologia e le tecniche per arrivare alla descrizione dei procedimenti risolutivi.

METODO TOP DOWN (metodo per la risoluzione dei problemi)

Partendo dal problema si deduce se:

- problema primitivo → soluzione immediata
- problema non primitivo → scomposizione del problema in sottoproblemi semplici la cui soluzione porti alla soluzione del problema iniziale. Il linguaggio (dichiarativo) scelto per descrivere ogni passo di questo processo e il programma è rappresentato dalla descrizione finale delle operazioni

METODO BOTTOM UP

Partendo dalla conoscenza di problemi primitivi, che quindi l'esecutore sa risolvere:

- riconoscere se la soluzione di una loro semplice sequenza porta alla soluzione del problema iniziale
- se la semplice sequenza non è sufficiente, bisogna trovare una sequenza appropriata (spesso molto complessa) di problemi primitivi la cui soluzione equivalga alla soluzione del problema stesso. Il programma è rappresentato dalla descrizione della sequenza (complessa) di problemi primitivi da risolvere

Come regola generale è opportuno usare il metodo TOP DOWN e ricorrere al metodo BOTTOM UP solo in casi teoricamente molto particolari

GENERALIZZAZIONE DI UN PROBLEMA → consiste nel passaggio da un'istanza specifica del problema (calcolare l'area di un rettangolo che ha base uguale a 4 cm e altezza 5 cm) alla sua formulazione generica (calcolare l'area di un rettangolo conoscendo la base e l'altezza).
Il procedimento risolutivo deve sempre essere costruito per il problema generico ed essere quindi valido per tutte le istanze ammissibili del problema.

SCOMPOSIZIONE DI UN PROBLEMA

Per procedere alla scomposizione di un problema complesso, si procede per 4 passi:

1. formalizzazione del problema
2. elencazione di alcuni casi di prova
3. scrittura del programma
4. verifica

1. FORMALIZZARE UN PROBLEMA:

1. dare un nome al problema(= stringa che consenta l'immediata individualizzazione)

2. individuare le entità (= variabili) coinvolte e i rispettivi insiemi in cui esse prendono valori;
3. comprendere le relazioni tra le variabili

Esempio: calcolare il volume della sfera, conoscendo il suo raggio

{volume – sfera(Raggio, Volume)} entità, iniziali maiuscole, separate da virgola, tra parentesi tonde

stringa, iniziali minuscole, precedute da parentesi graffe

2. I CASI DI PROVA

I casi di prova sono alcune istanze del problema, e la descrizione esplicita dei valori assunti dalle entità per quelle istanze, i valori devono rispettare le relazioni esistenti tra le entità individuate.

Nell'ipotesi esista un caso di prova che non si riesce a descrivere, significa che la formalizzazione adottata non è appropriata o che la relazione tra input e output non è stata compresa correttamente.

Al posto delle variabili troviamo dei valori effettivi, che devono essere riportati nel medesimo ordine in cui sono elencate le rispettive entità.

Esempio (sempre riferito al caso precedente)

I casi di prova sono evidenziati dai caratteri ?- posti a sinistra del termine e, come formalizzazione, sono racchiusi tra parentesi graffe.

{ ?- volume – sfera (0,0)}

{ ?- volume – sfera (1,4.1887893)}

{ ?- volume – sfera (2,33.5103144)}

Formalizzazione + casi prova = fase di comprensione del problema (il dialogo tra il primo ed il secondo attore, chi ha il problema e chi deve capire come risolverlo)

3. SCRITTURA DEL PROGRAMMA

Descrivibile come un soliloquio del secondo attore che consegnerà un testo al terzo attore, l'esecutore.

I programmi per risolvere problemi primitivi consistono nella trascrizione o della tabella da consultare o nella scrittura della (semplice) formula da applicare.

Esempio del volume della sfera

Volume-sfera(Raggio,Volume) :- Volume = 4x3.141592xRaggio**3/3

TESTA DELLA REGOLA — CORPO DELLA REGOLA — REGOLA

TESTA DELLA REGOLA = termine usato per la formalizzazione senza le parentesi graffe, perché indica che questa scrittura è diretta all'esecutore

CORPO DELLA REGOLA = formula che descrive la relazione tra le entità

:- = termine separatore la testa della regola dal corpo della regola

** = quando la formula contiene una potenza, si inserisce tra la base e l'esponente (Raggio**3)

I problemi primitivi sono risolubili mediante consultazione di una tabella (es. quelli che prevedono l'uso di un elenco telefonico, la consultazione della propria agenda…)

La tabella avrà un certo numero di colonne, ogni tabella deve avere un nome che deve ricordare il contenuto.

Il nome della tabella è una stringa che comincia con lettera minuscola, mentre le stringhe che designano le colonne iniziano con la maiuscola.

Descrizione implicita: serve per definire la struttura della tabella e per specificare il nome ed il contenuto delle singole colonne.
Es.

{agenda(nome,cognome,telefono,professione)}

Descrizioni esplicite: sono necessarie per la soluzione del problema, rappresenta il programma
Es.

agenda(mario,rossi,3334445551, avvocato

Es. di risoluzione di un problema

?- agenda(lucia, bini, Tel, impiegata)
?- agenda(Nome,rossi, _ , farmacista)

qnd il testo del problema non fa riferimento ad una entità che è invece prevista nella formalizzazione della tabella che si deve usare

UNIFICAZIONE = il procedimento con il quale l'esecutore risponde alle domande

Può accadere che nella domanda siano presenti solo costanti (in questo caso il termine si dice GROUND), per cui l'esecutore non dovrà trovare delle informazioni ma semplicemente verificare se nella tabella è presente una data riga: in questi casi la risposta dell'esecutore sarà semplicemente SI o NO

Anche con problemi di tipo aritmetico la risposta può essere SI o NO
Es.

?- area – rettangolo (2,4,5)

la risposta è NO perché non è vero che l'area di un rettangolo che ha base 2 e altezza 4 è 5

BACKTRACKING = è un metodo che viene usato quando si deve risolvere una sequenza di sottoproblemi. I sottoproblemi vengono risolti nell'ordine; quando la soluzione di un sottoproblema fallisce si cerca una nuova soluzione al precedente; risolto questo, si riprendono i tentativi di risolvere il successivo.

=/= significa che l'entità di sinistra deve essere diversa dall'entità di destra (es. Nome=/=Fratello.)

GRAFO = insieme di nodi (città che congiungono i vari percorsi stradali) e archi (i collegamenti stradali)
GRAFO NON ORIENTATO = qnd gli archi sono tutti a doppio senso
GRAFO ORIENTATO = qnd gli archi sono percorribili in un unico senso di marcia

Un grafo può essere rappresentato da una tabella,ove nella prima colonna si colloca il nodo di origine dell'arco e nella seconda quello di arrivo dell'arco

CICLO = un percorso che partendo da un certo nodo dell'arco vi fa ritorno dopo essere passato per altri nodi

ALBERO = è un arco orinentato nel quale ogni nodo destinazione è raggiungibile da uno e un solo nodo origine. In un albero deve esistere un nodo che non è destinazione di alcun arco (RADICE DELL'ALBERO); i nodi dell'albero che non sono origine di alcun arco si dicono FOGLIE DELL'ALBERO