Corso di Laurea in INFORMATICA

a.a. 2011-2012

Algoritmi e Strutture Dati

MODULO 7

STRUTTURE NON LINEARI

Generalità su grafi e alberi. Alberi binari: specifiche sintattiche e semantiche. Realizzazioni. Visita di alberi binari.

Abbiamo parlato di gerarchie, tassonomie, alberi utili per rappresentare:

- alberi genealogici
- classificazioni di specie animali
- organizzazione del territorio
  - continente, nazione, regione, provincia ecc
- (alcuni) organigrammi
- gerarchie di ereditarietà
  - ad es., in C^{++}, Java etc.
- gerarchie di domini Internet
- file system
- procedimento di ricerca all’interno di una tabella

SI CONSIDERI L'ORGANIZZAZIONE DI UN INDICE:

0. ***TIPI DI DATO E STRUTTURE DATI***

    0.1 STRUTTURE DI DATI: SPECIFICHE E REALIZZAZIONI

    0.2 RAPPRESENTAZIONE IN MEMORIA

1. ***LISTE***

    1.1 REALIZZAZIONE CON PUNTATORI

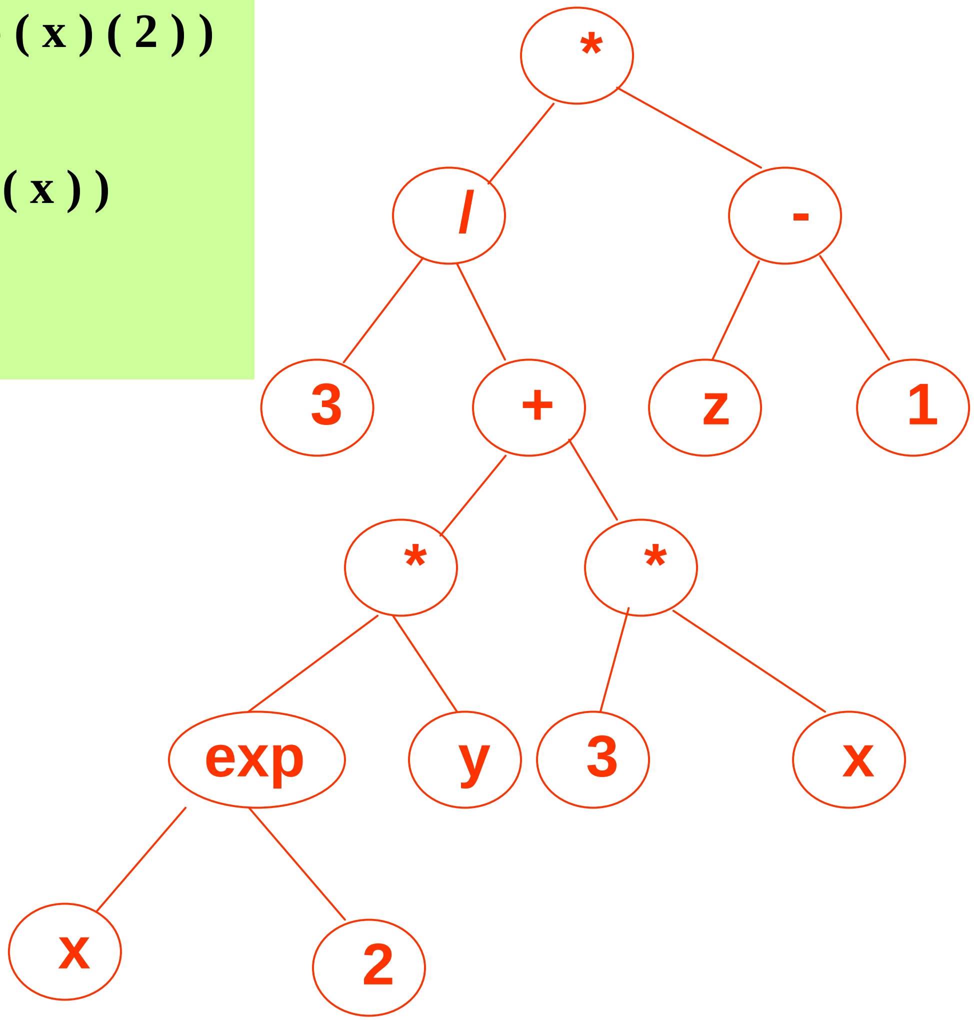
    1.2 REALIZZAZIONE CON CURSORI

    1.3 REALIZZAZIONE CON DOPPI PUNTATORI

IN QUESTA ORGANIZZAZIONE OGNI ARGOMENTO PRINCIPALE HA DIVERSI ARGOMENTI SECONDARI, OGNUNO DEI QUALI PUO' DIVIDERSI IN SOTTOARGOMENTI E COSI' VIA. QUESTO PUO' ESSERE RAPPRESENTATO MEDIANTE UN ALBERO CHE RAPPRESENTA L'ORGANIZZAZIONE GERARCHIZZATA DEI CONTENUTI

UN ESEMPIO DI ALBERO (DI PARSING) DA PERCORRERE DAL BASSO VERSO L'ALTO PER LA VERIFICA DELL'ESPRESSIONE

```
( * ( / ( 3 )
          ( + ( * ( exp ( x ) ( 2 ) )
                 ( y )
                )
               ( + ( 3 ) ( x ) )
          )
     ( - ( z ) ( 1 ) )
)
```



$(3/(x^2y+3x))*(z-1)$

IN GENERALE

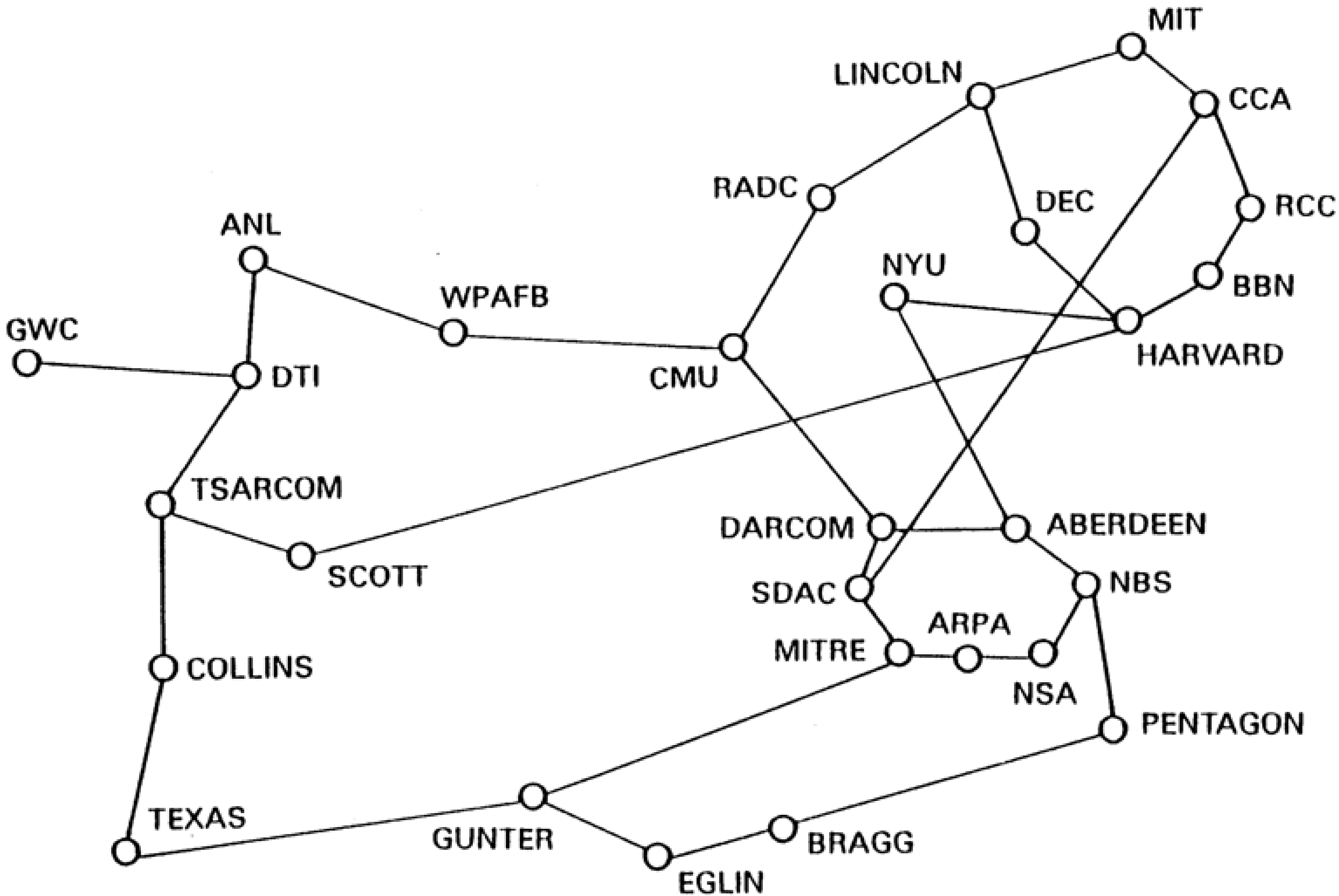
L'ALBERO E' UNA STRUTTURA INFORMATIVA FONDAMENTALE UTILE PER RAPPRESENTARE:

- **PARTIZIONI SUCCESSIVE DI UN INSIEME IN SOTTOINSIEMI DISGIUNTI**
- **ORGANIZZAZIONI GERARCHICHE DI DATI**
- **PROCEDIMENTI DECISIONALI ENUMERATIVI**

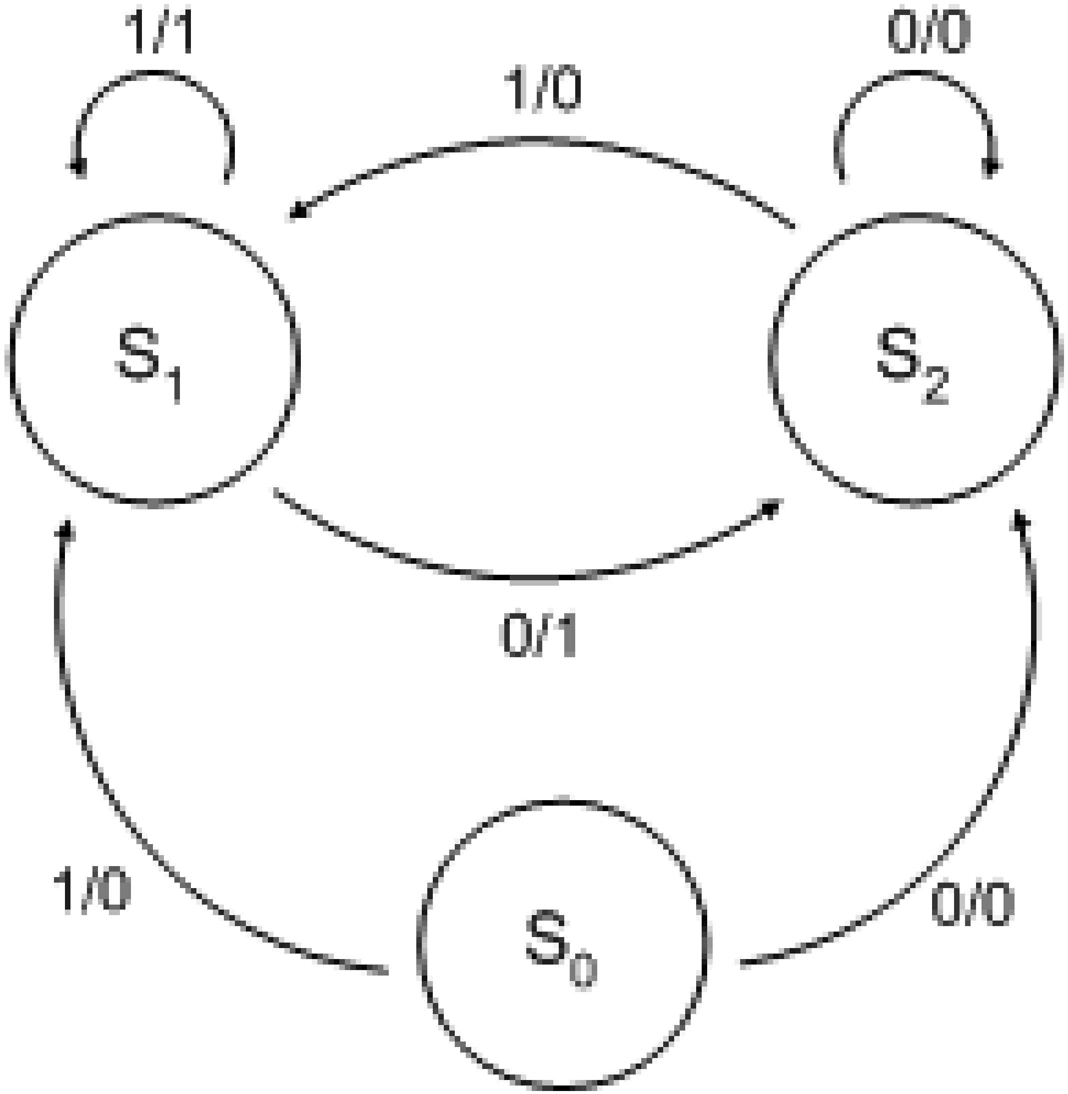
L'ALBERO E' UN CASO PARTICOLARE DI GRAFO

IL GRAFO E' UNA STRUTTURA DATI GENERALE ALLA QUALE SI POSSONO RICONDURRE STRUTTURE PIU' SEMPLICI: ANCHE LE LISTE POSSONO ESSERE CONSIDERATE UN CASO PARTICOLARE DI GRAFO.

UN ESEMPIO DI GRAFO



Parte della mappa di ARPANET.



Grafo di rappresentazione di un automa di Mealy

GRAFI E ALBERI

LA DIFFERENZA TRA LE DUE STRUTTURE:

- **GENERALMENTE NELL'ALBERO VI E' UNA DIREZIONALITA' CHE UTILIZZIAMO PER RAPPRESENTARE PARTIZIONI SUCCESSIVE O TASSONOMIE GERARCHICHE.**

- **NEL GRAFO QUESTA DIREZIONALITA', SE ESISTE, NON E' PREDEFINITA NE' UTILIZZATA PER RAPPRESENTARE UN RANGO NELLA ORGANIZZAZIONE DEI DATI MA PIUTTOSTO LA DIREZIONE DELLA RELAZIONE TRA I NODI COLLEGATI.**

TUTTI GLI ELEMENTI (*nodi*) DEL GRAFO SONO SULLO STESSO PIANO E LA STRUTTURA DATI RAPPRESENTA L'ESISTENZA DI UNA CONNESSIONE TRA ELEMENTI.

GRAFO: Definizione

Un grafo G=(N,A) consiste in:

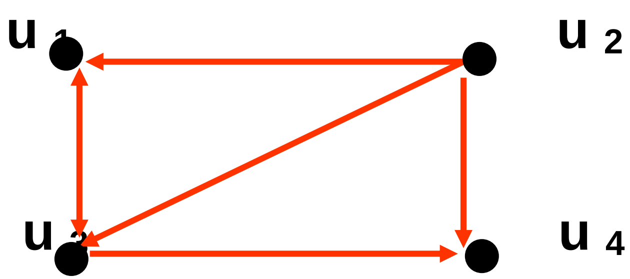
- un insieme N di vertici (o nodi)
- un insieme A di coppie di vertici, detti archi o spigoli: ogni arco connette due vertici

Esempio 1: N={persone che vivono in Italia}, A={coppie {x,y} tali che x e y si sono stretta la mano}

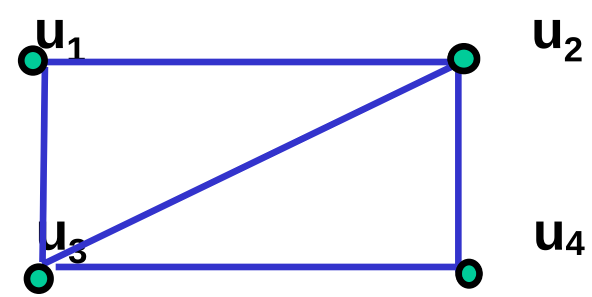
Esempio 2: N={persone che vivono in Italia}, A={coppie (x,y) tale che x ha inviato una mail a y}

GRAFI ORIENTATI E NON ORIENTATI: DEFINIZIONI

IN UN GRAFO ORIENTATO (u_i,u_j) e (u_j,u_i) INDICANO DUE ARCHI DISTINTI, IN UN GRAFO NON ORIENTATO INDICANO LO STESSO ARCO CHE INCIDE SUI DUE NODI.



1. Grafo orientato



2. Grafo non orientato

IN UN GRAFO NON ORIENTATO I NODI CONGIUNTI DA UN ARCO SONO DETTI ADIACENTI. NELL'ESEMPIO I NODI u_1 E u_3 SONO ADIACENTI MA u_1 E u_4 NON LO SONO.

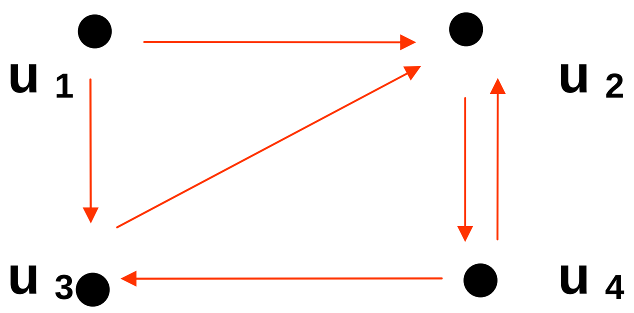
IN UN GRAFO ORIENTATO G UN CAMMINO E' UNA SEQUENZA DI NODI

u_0, u_1, …, u_k TALI CHE

$(u_i, u_{i+1}) \in A$, PER i = 0, 1, 2, … , k-1.

GRAFO CONNESSO E' UN GRAFO G = <N,A> IN CUI, DATI u E v $\in$ N, ESISTE UN CAMMINO DA u A v O UN CAMMINO DA v AD u.

UN GRAFO ORIENTATO G E' DETTO FORTEMENTE CONNESSO SE PER OGNI COPPIA DI NODI u E v ESISTE ALMENO UN CAMMINO DA u A v ED ALMENO UN CAMMINO DA v AD u.



GRAFO ORIENTATO MA NON FORTEMENTE CONNESSO (NON ESISTE UN CAMMINO DA u_4 A u_1)

UN GRAFO E' DETTO COMPLETO SE PER OGNI COPPIA DI NODI $(u_i,u_j) \in N$ ESISTE UN ARCO CHE VA DA u_i AD u_j, ($A = N$xN).

ALBERI

L’ALBERO E’ UN PARTICOLARE TIPO DI GRAFO DEFINITO MATEMATICAMENTE CON UNA COPPIA

$$T = (N,A)$$

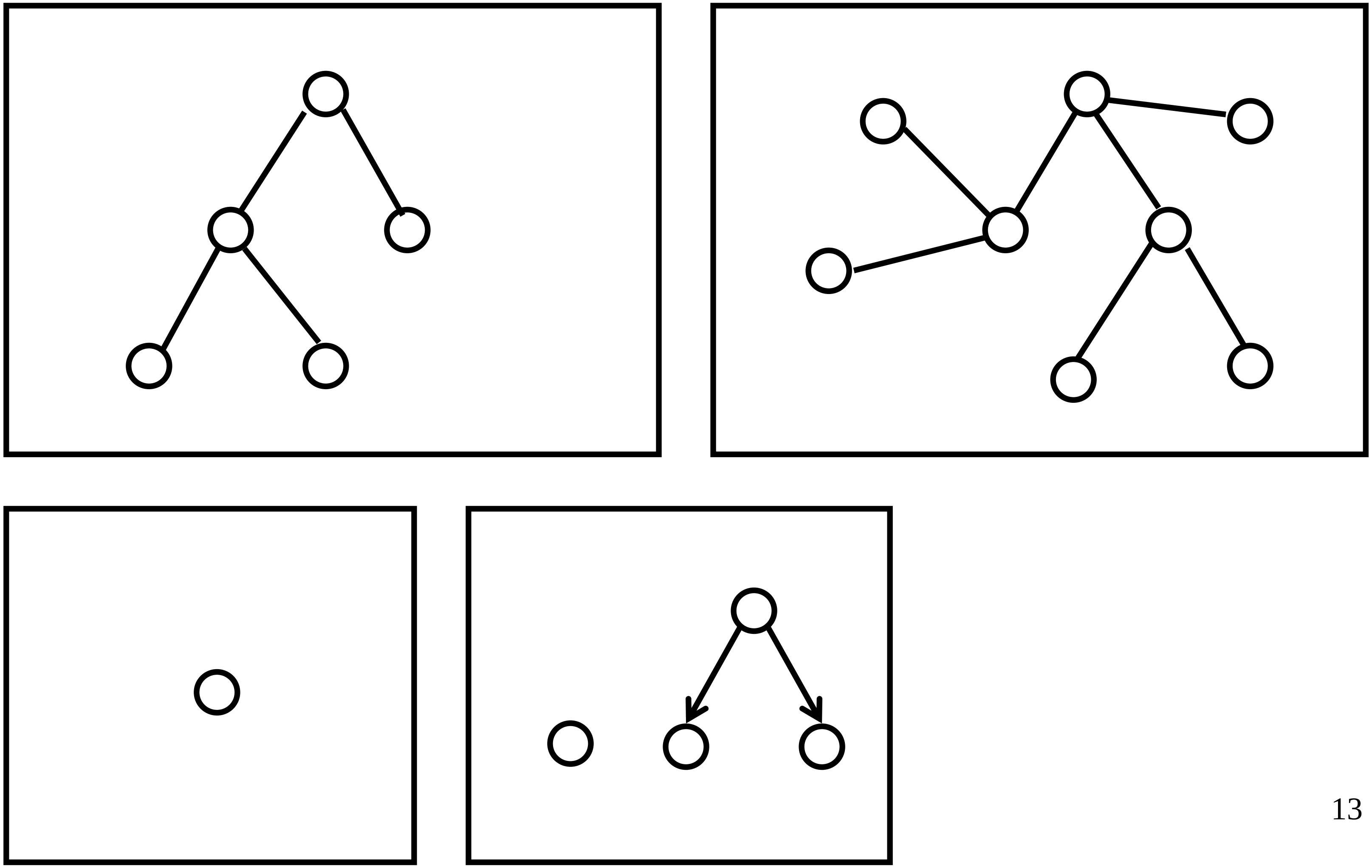
DOVE **N** E’ UN INSIEME FINITO DI NODI ED **A** E’ UN INSIEME DI COPPIE NON ORDINATE (**ALBERO LIBERO**) TALI CHE :

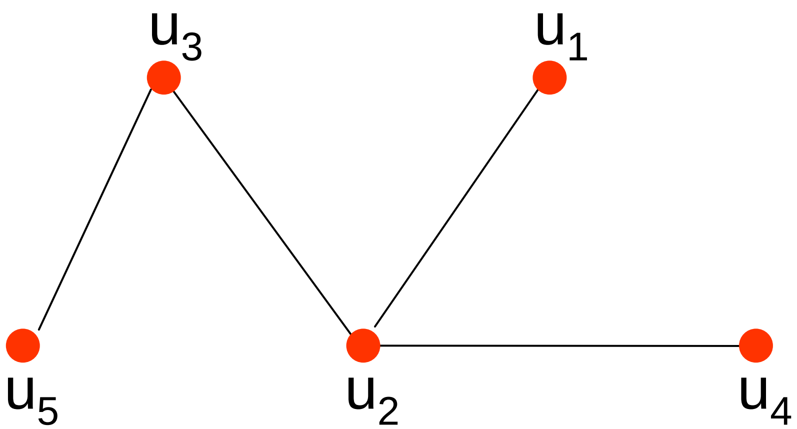
a) IL NUMERO DI ARCHI E’ UGUALE AL NUMERO DI NODI MENO UNO $|A| = |N| - 1$

b) T E’ CONNESSO, OVVERO PER OGNI COPPIA DI NODI u E v IN N, ESISTE UNA SEQUENZA DI NODI DISTINTI $u_0, u_1, \ldots, u_k$ TALI CHE $u = u_0$, $v = u_k$ E LA COPPIA $<u_i, u_{i+1}>$ E’ UN ARCO DI A, PER $i = 0, 1, \ldots, k-1$.

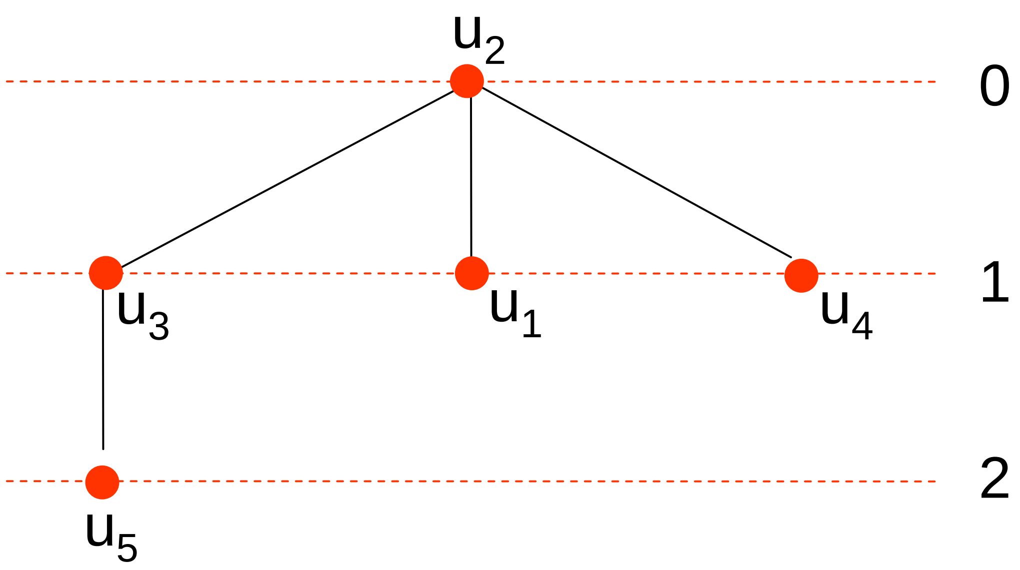
UN **ALBERO RADICATO** E’ OTTENUTO DA UN ALBERO LIBERO DESIGNANDO ARBITRARIAMENTE UN NODO **r** COME **RADICE** E DISPONENDO I NODI PER **LIVELLI**.

Sono alberi?

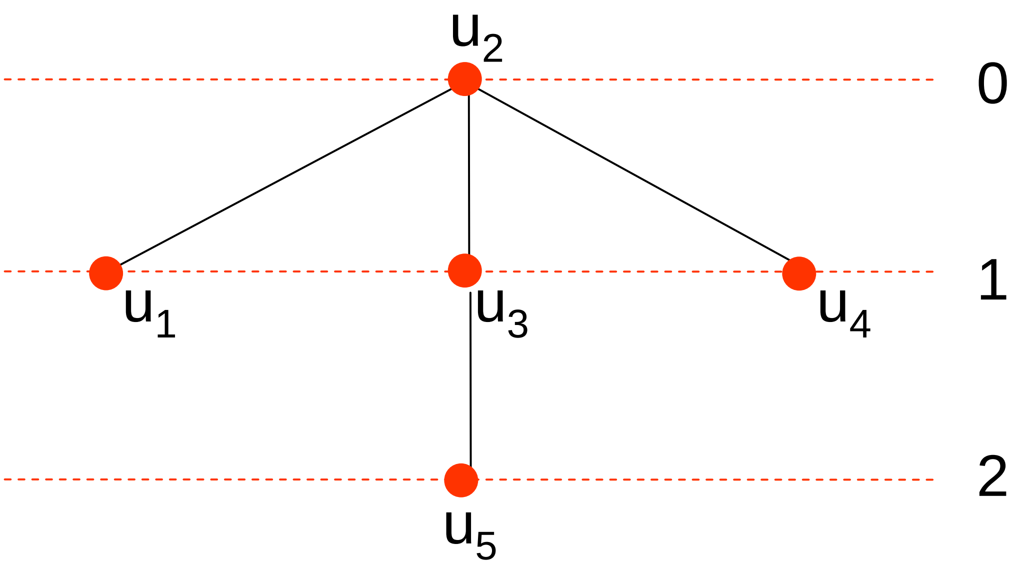




ALBERO NON RADICATO



ALBERO RADICATO



ALBERO RADICATO ORDINATO

LA RADICE r E' A LIVELLO 0 E TUTTI I NODI u, ∋′ (u, r) ∈ A, SONO FIGLI DI r E STANNO A LIVELLO 1 (r E' PADRE O GENITORE). NODI CON LO STESSO PADRE SONO FRATELLI. NODI TERMINALI SENZA FIGLI SONO DETTI FOGLIE.

UN ALBERO ORDINATO E' OTTENUTO DA UNO RADICATO STABILENDO UN *ORDINAMENTO TRA NODI ALLO STESSO LIVELLO*.

UN ALBERO RADICATO E' ACICLICO E VALGONO LE SEQUENTI PROPRIETA':

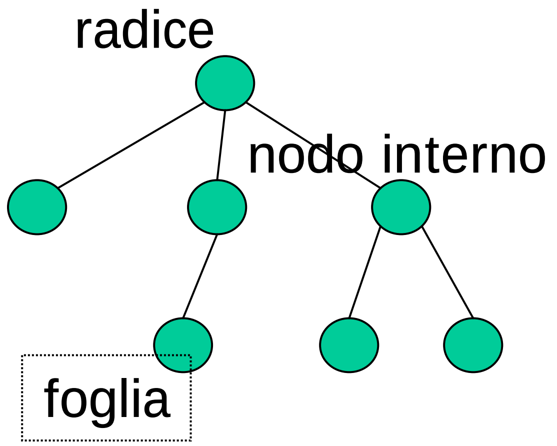
- **PER OGNI NODO C'E' UN SOLO ARCO ENTRANTE (TRANNE CHE PER LA RADICE CHE NON NE HA NESSUNO)**

- **E' DEBOLMENTE CONNESSO**
  - **SE ESISTE UN CAMMINO CHE VA DA UN NODO u AD UN ALTRO NODO v, TALE CAMMINO E' UNICO**
  - **IN UN ALBERO ESISTE UN SOLO CAMMINO CHE VA DALLA RADICE A QUALUNQUE ALTRO NODO**

- **TUTTI I NODI DI UN ALBERO T (TRANNE r) POSSONO ESSERE RIPARTITI IN INSIEMI DISGIUNTI CIASCUNO DEI QUALI INDIVIDUA UN ALBERO (DATO UN NODO u, I SUOI DISCENDENTI COSTITUISCONO UN ALBERO DETTO SOTTOALBERO DI RADICE u)**

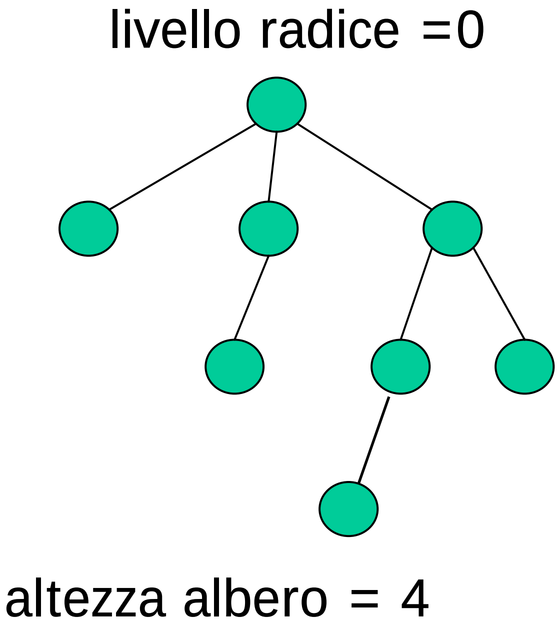
terminologia

- genitore, antenato, figlio, discendente
  - concetti intuitivi
- nodo interno
  - se ha almeno un figlio
- linea
  - connette due nodi uno dei quali è genitore dell’altro
- cammino
  - sequenza di linee che connettono due nodi uno dei quali è antenato dell’altro



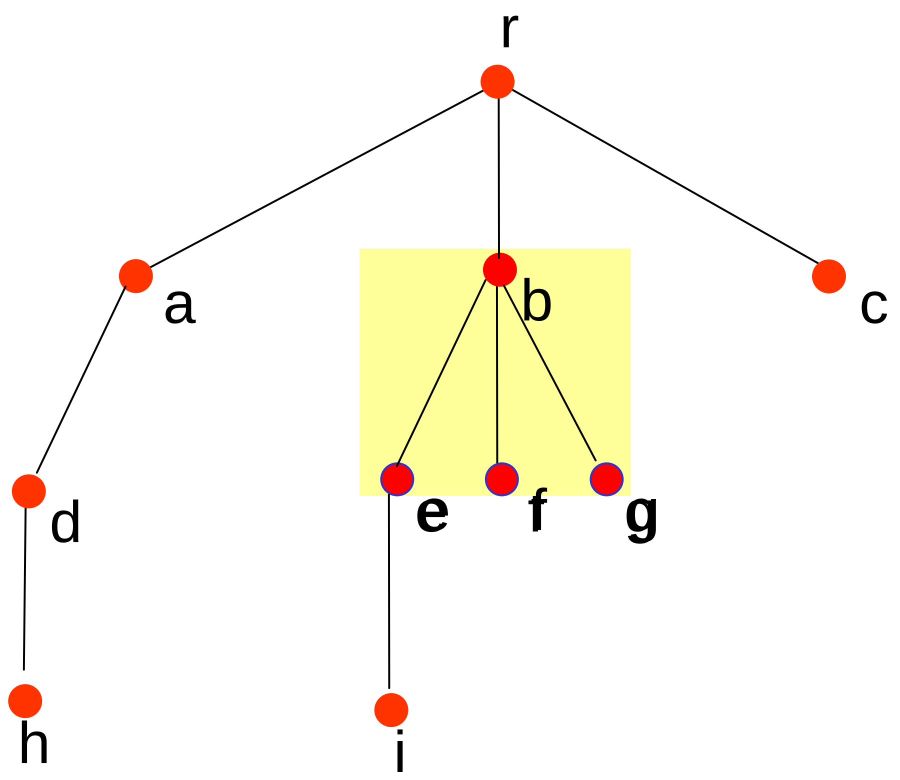
terminologia

- livello/profondità di un nodo
  - lunghezza cammino radice-nodo
- ramo
  - percorso radice-foglia
- altezza nodo
  - lunghezza del cammino più lungo da quel nodo a una foglia
- altezza (profondità) albero
  - lunghezza (n.ro nodi) ramo più lungo dell'albero



Ordine di un albero

DEFINIAMO ALBERO DI ORDINE K UN ALBERO IN CUI OGNI NODO HA AL MASSIMO K FIGLI



ALBERO TERNARIO

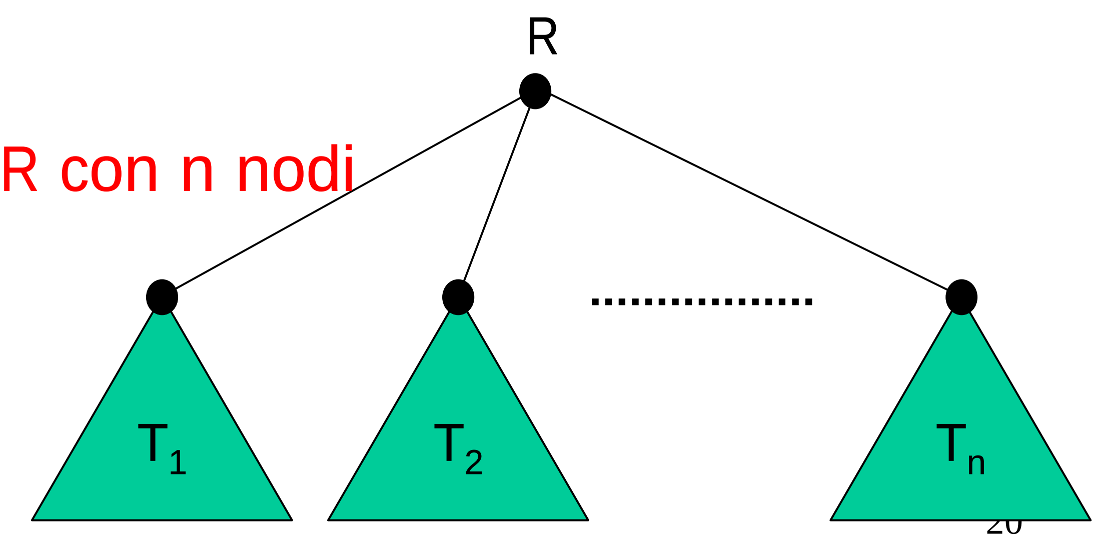
O DI ORDINE (RANGO) 3

tipo di dato astratto albero

UN ALBERO PUO' ESSERE DEFINITO RICORSIVAMENTE

- **UN ALBERO E' UN INSIEME NON VUOTO DI NODI AI QUALI SONO ASSOCIATE DELLE INFORMAZIONI**
- **TRA I NODI ESISTE UN NODO PARTICOLARE CHE E' LA RADICE (LIVELLO 0)**
- **GLI ALTRI NODI SONO PARTIZIONATI IN SOTTOINSIEMI CHE SONO A LORO VOLTA ALBERI (LIVELLI SUCCESSIVI)**

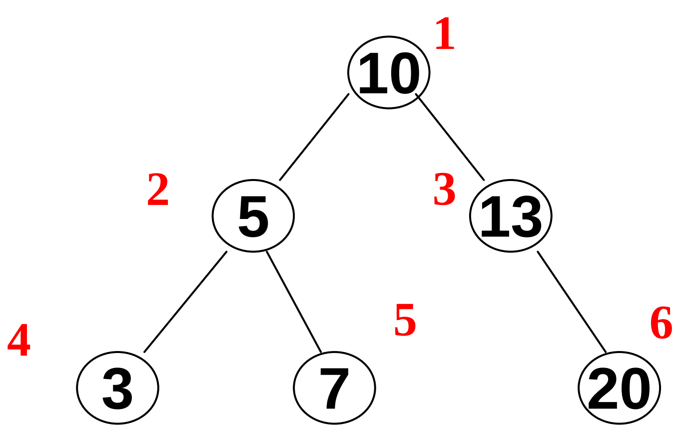
es.: radice R con n nodi

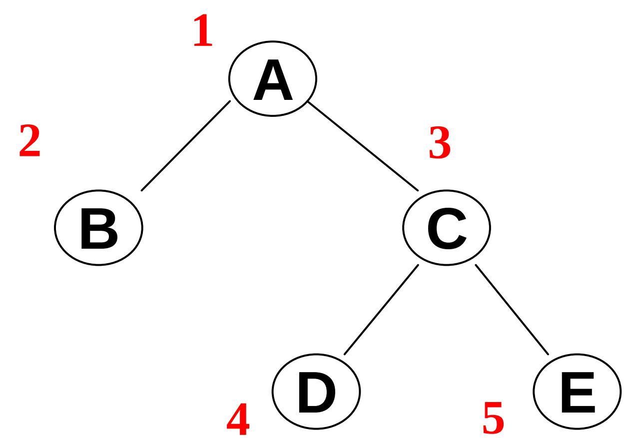


sottoalberi

ALBERI BINARI

SONO ALBERI ORDINATI IN CUI OGNI NODO HA AL PIU' DUE FIGLI (FIGLIO SINISTRO E FIGLIO DESTRO). AI NODI E' ASSOCIATO UN NUMERO D'ORDINE. I NODI HANNO UN'ETICHETTA (LABEL) ASSOCIATA (INTERI, CARATTERI etc.).





DEFINIZIONE:

UN ALBERO BINARIO E' UN GRAFO ORIENTATO CHE O E' VUOTO O E' COSTITUITO DA UN SOLO NODO O E' FORMATO DA UN NODO N (DETTO RADICE) E DA DUE SOTTOALBERI BINARI, CHE VENGONO CHIAMATI RISPETTIVAMENTE SOTTOALBERO (O FIGLIO) SINISTRO E SOTTOALBERO (O FIGLIO) DESTRO

QUALCHE CONSIDERAZIONE TEORICA CIRCA LA CORRISPONDENZA TRA “ALBERO BINARIO” E “LISTA”

OGNI ALBERO BINARIO T PUO’ ESSERE CONSIDERATO EQUIVALENTE AD UNA LISTA NEL MODO SEGUENTE:

- **SE T E’ VUOTO, LA LISTA CHE LO RAPPRESENTA E’ LA LISTA VUOTA**

- **SE T NON E’ VUOTO, LA LISTA CHE LO RAPPRESENTA E’ FORMATA DA TRE ELEMENTI:**

  - IL PRIMO E’ L’ATOMO CHE RAPPRESENTA LA RADICE DI T

  - IL SECONDO E’ UNA LISTA CHE RAPPRESENTA, CON LO STESSO METODO, IL SOTTOALBERO SINISTRO DI T

  - IL TERZO E’ UN’ALTRA LISTA CHE RAPPRESENTA IL SOTTOALBERO DESTRO DI T

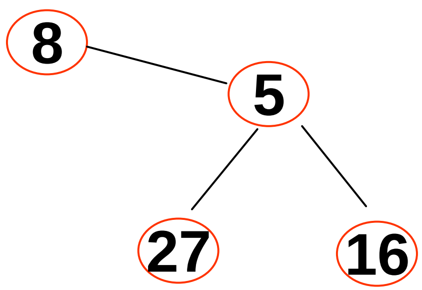
POSSIAMO USARE UNA NOTAZIONE CON PARENTESI PER RAPPRESENTARE UN ALBERO BINARIO MEDIANTE LISTA

() **ALBERO VUOTO**

(a) **ALBERO COSTITUITO DALLA SOLA RADICE**

(a () ()) **ALBERO BINARIO COSTITUITO DA RADICE a, UN FIGLIO SINISTRO VUOTO E UN FIGLIO DESTRO VUOTO**

AD ESEMPIO L’ALBERO SEGUENTE



CORRISPONDE ALLA LISTA (8 () (5 (27 () ()) (16 () ())))

LA SPECIFICA SINTATTICA

TIPI: ALBEROBIN, BOOLEANO, NODO

CREABINALBERO : ( ) → ALBEROBIN

BINALBEROVUOTO : (ALBEROBIN) → BOOLEANO

BINRADICE : (ALBEROBIN) → NODO

BINPADRE : (NODO,ALBEROBIN) → NODO

FIGLIOSINISTRO : (NODO,ALBEROBIN) → NODO

FIGLIODESTRO : (NODO,ALBEROBIN) → NODO

SINISTROVUOTO : (NODO,ALBEROBIN) → BOOLEANO

DESTROVUOTO : (NODO,ALBEROBIN) → BOOLEANO

COSTRBINALBERO : (ALBEROBIN,ALBEROBIN) → ALBEROBIN

CANCSOTTOBINALBERO : (NODO,ALBEROBIN) → ALBEROBIN

LA SPECIFICA SEMANTICA

TIPI: ALBEROBIN: insieme degli alberi binari T=(N,A), nei quali ad ogni nodo è associato un LIVELLO, BOOLEANO, NODO

CREABINALBERO = T'

POST: T' = Λ

--

BINALBEROVUOTO(T) = b

POST: b=VERO SE T = Λ; b=FALSO ALTRIMENTI

--

BINRADICE(T) = u

PRE: T $\neq \Lambda$

POST: u → RADICE DI T → LIVELLO(u) = 0

--

BINPADRE(u,T) = v

PRE: T $\neq \Lambda$, u $\in$ N, LIVELLO(u) > 0

POST: v E' PADRE DI u → (v,u) $\in$ A → LIVELLO(u)=LIVELLO(v)+1

FIGLIOSINISTRO(u,T) = v

PRE: T $\neq \Lambda$, u $\in$ N, u HA UN FIGLIO SINISTRO

POST: v E' IL FIGLIO SINISTRO DI u IN T

FIGLIODESTRO(u,T) = v

PRE: T $\neq \Lambda$, u $\in$ N, u HA UN FIGLIO DESTRO

POST: v E' IL FIGLIO DESTRO DI u IN T

SINISTROVUOTO(u,T) = b

PRE: T $\neq \Lambda$, u $\in$ N

POST: b=VERO SE u NON HA UN FIGLIO SINISTRO

b=FALSO ALTRIMENTI

DESTROVUOTO(u,T) = b

PRE: T $\neq \Lambda$, u $\in$ N

POST: b=VERO SE u NON HA UN FIGLIO DESTRO

b=FALSO ALTRIMENTI

COSTRBINALBERO(T,T') = T''

POST: T'' SI OTTIENE DA T E DA T' INTRODUCENDO AUTOMATICAMENTE UN NUOVO NODO r'' (RADICE DI T'') CHE AVRA' COME SOTTOALBERO SINISTRO T E SOTTOALBERO DESTRO T'

(*SE T = Λ E T' = Λ, L'OPERATORE INSERISCE LA SOLA RADICE r'';*

***SE T = Λ, r'' NON HA FIGLIO SINISTRO; SE T' = Λ, r'' NON HA FIGLIO DESTRO*)**

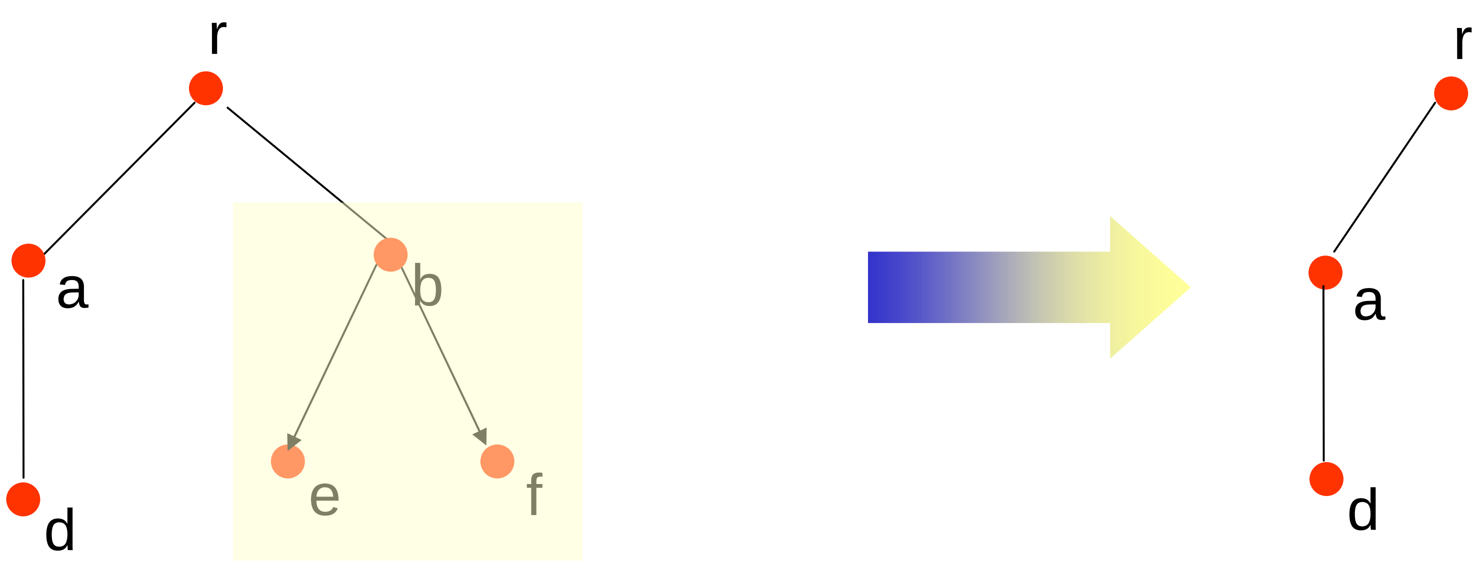
CANCSOTTOBINALBERO(u,T) = T'

PRE: T $\neq \Lambda$, u $\in$ N

POST: T' E' OTTENUTO DA T ELIMINANDO IL SOTTOALBERO DI RADICE u, CON TUTTI I SUOI DISCENDENTI

VALIDA PER ALBERI DI OGNI ORDINE AGISCE POTANDO DAL NODO u. AD ESEMPIO:

CANCSOTTOBINALBERO(b,T)



ANCORA DUE OPERATORI UTILI !!!

SPECIFICHE SINTATTICHE

Tipi: Va aggiunto TIPOELEM del tipo dell’etichetta se si vuole associare informazione ai nodi

LEGGINODO: (NODO,ALBEROBIN) → TIPOELEM

SCRIVINODO: (TIPOELEM,NODO,ALBEROBIN) → ALBEROBIN

SPECIFICHE SEMANTICHE

LEGGINODO (n,T) = a

PRE: n E’ UN NODO DI T, n $\in$ N

POST: a E’ IL VALORE ASSOCIATO AL NODO n IN T

SCRIVINODO (a,n,T) = T'

PRE: n E’ UN NODO DI T, n $\in$ N

POST: T' E’ IL NUOVO ALBERO CORRISPONDENTE AL VECCHIO T CON IL VALORE a ASSEGNATO AL NODO n

In modo simile si possono prevedere operatori per associare informazioni agli archi (peso, costo etc.)

L'ALGEBRA PRESENTATA RAPPRESENTA UNA SPECIFICA SCELTA DI PROGETTO

SI E' SCELTO DI ENFATIZZARE LA NATURA RICORSIVA DEGLI ALBERI E DI COSTRUIRE L'ALBERO BINARIO DAL BASSO VERSO L'ALTO, CIOE' DAL LIVELLO DELLE FOGLIE VERSO LA RADICE.

UNA SCELTA ALTERNATIVA:

UN'ALGEBRA CHE PREVEDA DI COSTRUIRE L'ALBERO DALL'ALTO VERSO IL BASSO, INSERENDO PRIMA LA RADICE E POI I NODI FIGLI VIA VIA. ANDREBBE SOSTITUITO L'OPERATORE DI COSTRUZIONE CON TRE OPERATORI NUOVI, UNO DEDICATO ALL'INSERIMENTO DELLA RADICE E GLI ALTRI DUE DEDICATI ALL'INSERIMENTO DEL FIGLIO SINISTRO E DEL FIGLIO DESTRO.

LA SPECIFICA SINTATTICA

INSBINRADICE: (*nodo, alberobin*) $\rightarrow$ *alberobin*

INSFIGLIOSINISTRO: *(nodo, alberobin, alberobin)* $\rightarrow$ *alberobin*

INSFIGLIODESTRO: *(nodo, alberobin, alberobin)* $\rightarrow$ *alberobin*

LA SPECIFICA SEMANTICA

INSBINRADICE(u, T) =T'

PRE: T=Λ

POST: T'=(N,A), N={u}, LIVELLO(u)=0, A=Ø

INSFIGLIOSINISTRO(u, T, T') = T''

PRE: T$\neq\Lambda$, T'$\neq\Lambda$, u $\in$ N, SINISTROVUOTO(u)= True

POST: N''= N$\cup$N', T'' E' OTTENUTO DA T AGGIUNGENDO L'ALBERO T' COME FIGLIO SINISTRO DI u

INSFIGLIODESTRO(u, T, T') = T''

PRE: T$\neq\Lambda$, T'$\neq\Lambda$, u $\in$ N, DESTROVUOTO(u)= True

POST: N''= N$\cup$N', T' E' OTTENUTO DA T AGGIUNGENDO L'ALBERO T' COME FIGLIO DESTRO DI u

PER ISPEZIONARE O ATTRAVERSARE GLI ALBERI, GARANTENDO DI AVERLI ESAMINATI COMPLETAMENTE, SI DEFINISCONO I COSIDDETTI

ALGORITMI DI VISITA

CIOE’ ALGORITMI CHE CONSENTONO DI ANALIZZARE TUTTI I NODI DELL’ALBERO IN UN ORDINE DEFINITO.

RISULTANO PARTICOLARMENTE IMPORTANTI IN PROBLEMI PER I QUALI, AD ESEMPIO, E’ RILEVANTE DETERMINARE LA ALTEZZA DELL’ALBERO OPPURE SI DEBBA RICERCARE IN QUALE NODO E’ CONTENUTO IN ETICHETTA UN VALORE DATO IN INPUT E SI VOGLIA VERIFICARE LA PROFONDITA’ DEL NODO.

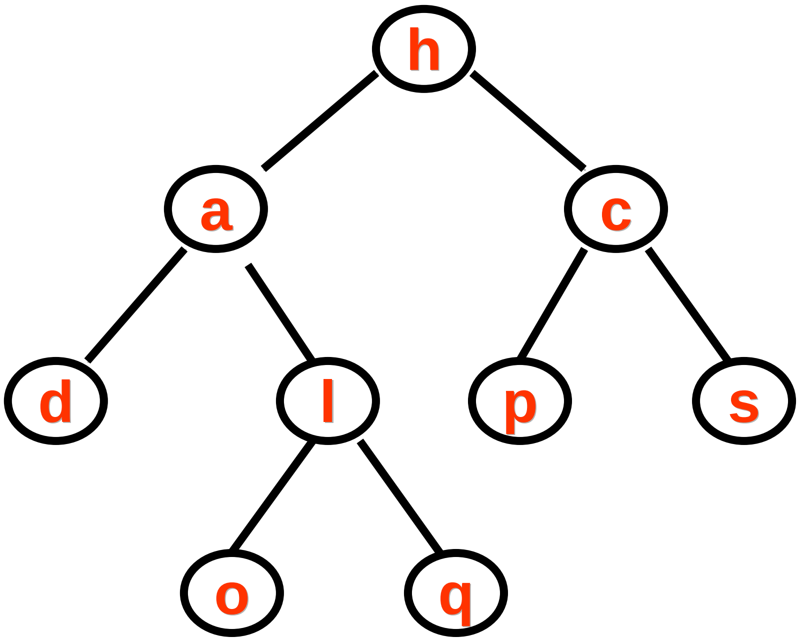
LA VISITA DI UN ALBERO CONSISTE NEL SEGUIRE UNA ROTTA DI VIAGGIO CHE CONSENTA DI ESAMINARE OGNI NODO DELL’ALBERO ESATTAMENTE UNA VOLTA.

PER GLI ALBERI BINARI I PIU' COMUNI ALGORITMI DI VISITA SONO TRE:

- **VISITA IN PRE-ORDINE: SI APPLICA AD UN ALBERO NON VUOTO E RICHIEDE DAPPRIMA L'ANALISI DELLA RADICE DELL'ALBERO E, POI, LA VISITA, EFFETTUATA CON LO STESSO METODO, DEI DUE SOTTOALBERI, PRIMA IL SINISTRO, POI IL DESTRO**
- **VISITA IN POST-ORDINE: SI APPLICA AD UN ALBERO NON VUOTO E RICHIEDE DAPPRIMA LA VISITA, EFFETTUATA CON LO STESSO METODO, DEI SOTTOALBERI, PRIMA IL SINISTRO E POI IL DESTRO, E, IN SEGUITO, L'ANALISI DELLA RADICE DELL'ALBERO**
- **VISITA SIMMETRICA: RICHIEDE PRIMA LA VISITA DEL SOTTOALBERO SINISTRO (EFFETTUATA SEMPRE CON LO STESSO METODO), POI L'ANALISI DELLA RADICE, E POI LA VISITA DEL SOTTOALBERO DESTRO**

ESEMPIO:

SIA UN ALBERO BINARIO CHE HA DEI CARATTERI NEI NODI



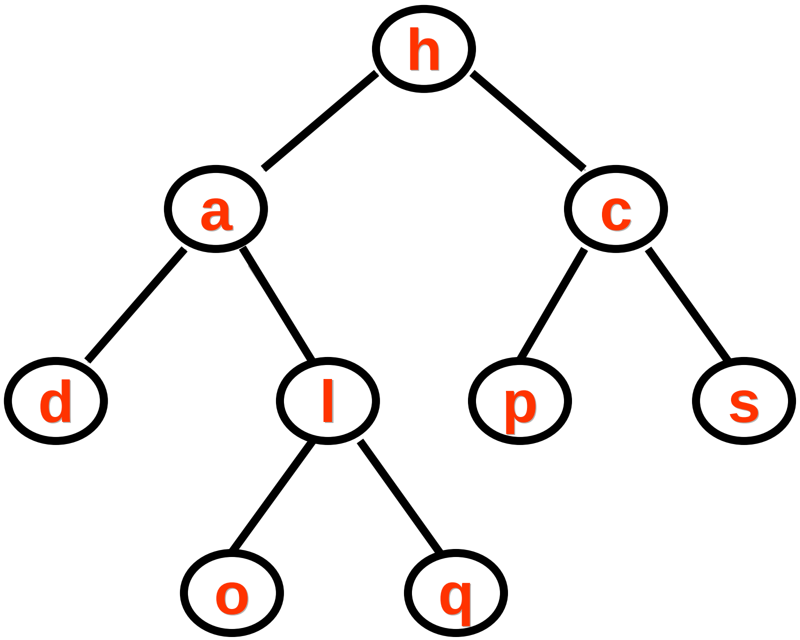
LA VISITA IN PREORDINE: h a d l o q c p s

LA VISITA IN POSTORDINE: d o q l a p s c h

LA VISITA SIMMETRICA: d a o l q h p c s

ESEMPIO:

SIA UN ALBERO BINARIO CHE HA DEI CARATTERI NEI NODI



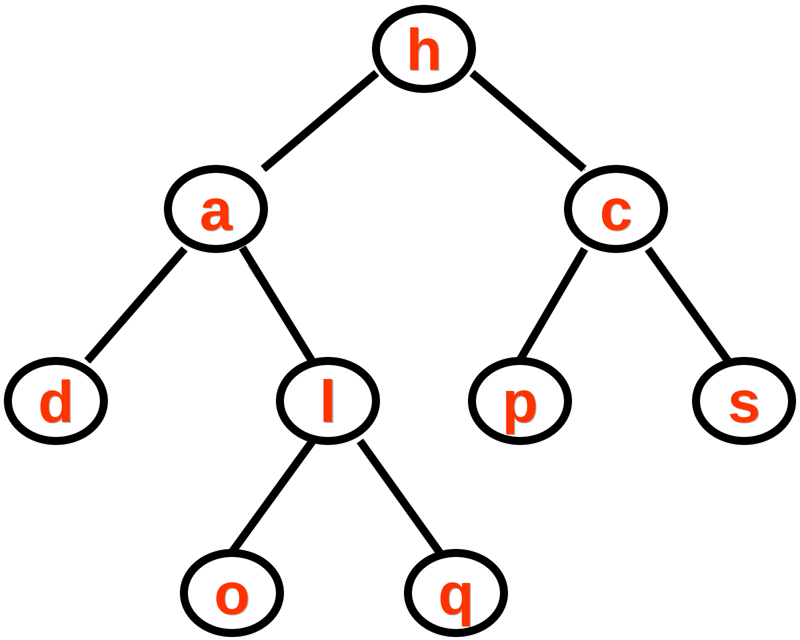
LA VISITA IN PREORDINE: h a d l o q c p s

LA VISITA IN POSTORDINE: d o q l a p s c h

LA VISITA SIMMETRICA: d a o l q h p c s

ESEMPIO:

SIA UN ALBERO BINARIO CHE HA DEI CARATTERI NEI NODI



LA VISITA IN PREORDINE: h a d l o q c p s

LA VISITA IN POSTORDINE: d o q l a p s c h

LA VISITA SIMMETRICA: d a o l q h p c s

LA FORMULAZIONE DEGLI ALGORITMI DI VISITA

GLI ALGORITMI SI POSSONO FACILMENTE FORMULARE IN MODO RICORSIVO.

AD ESEMPIO

LA VISITA IN PREORDINE L'ALBERO BINARIO T

SE L'ALBERO NON E' VUOTO

ALLORA

ANALIZZA LA RADICE DI T

VISITA IN PREORDINE IL SOTTOALBERO SINISTRO DI T

VISITA IN PREORDINE IL SOTTOALBERO DESTRO DI T

FINE

IN GENERALE

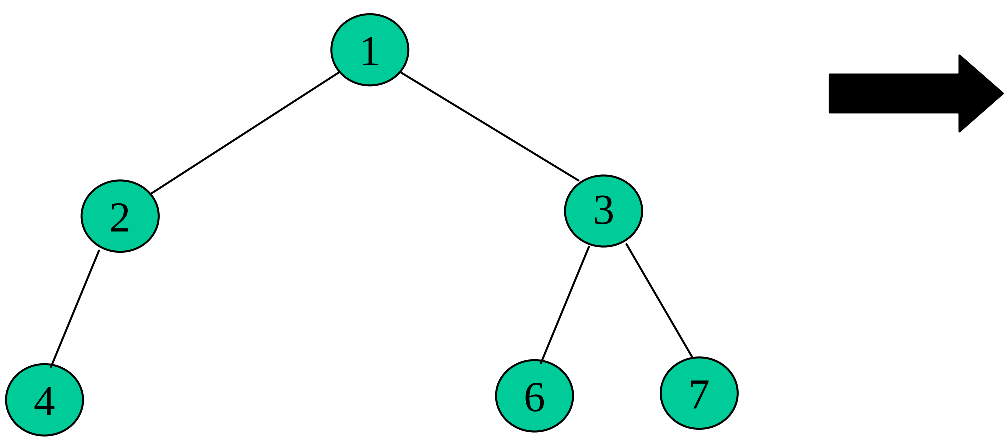
```
PROCEDURE BINVISITA (u tiponodo, T tipoalbero)
     IF BINALBEROVUOTO(T)=FALSE
     THEN
(1)                 {ESAMINA u};
          IF NOT SINISTROVUOTO(u,T) THEN
                    BINVISITA(FIGLIOSINISTRO(u,T),T)
(2)       IF NOT DESTROVUOTO(u,T) THEN
                    BINVISITA(FIGLIODESTRO(u,T),T)
(3)
          FINE
```

LA PREVISITA SI OTTIENE ESAMINANDO IL NODO u SOLTANTO NELL'ISTRUZIONE (1), MENTRE LA VISITA SIMMETRICA SI HA ESAMINANDO IL NODO u SOLO IN (2) E LA POSTVISITA ESAMINANDO IL NODO u SOLO IN (3).

RAPPRESENTAZIONE SEQUENZIALE

UNA POSSIBILE RAPPRESENTAZIONE DI UN ALBERO BINARIO E' QUELLA SEQUENZIALE MEDIANTE VETTORE.

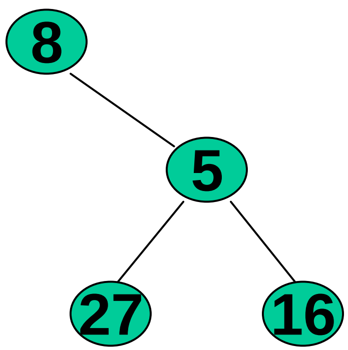
- ogni nodo v è memorizzato in posizione p(v)
  - se v è la radice allora p(v)=1 (indice 0 in Java, C, C++)
  - se v è il figlio sinistro di u allora p(v)=2p(u)
  - se v è il figlio destro di u allora p(v)=2p(u)+1



1
2
3
4
-
6
7
-

SI POSSONO USARE GLI INDICI PER DENOTARE I NODI E GLI ELEMENTI DELL'ARRAY PER DENOTARE L'ETICHETTA CHE PUO' ESSERE QUALUNQUE (carattere, stringa, record etc.).

SE L'ALBERO E' INCOMPLETO RIMANGONO ELEMENTI VUOTI



1	8
2	-
3	5
4	-
5	-
6	27
7	16

RAPPRESENTAZIONE SEQUENZIALE

Realizzazione statica: è necessaria una stima del numero massimo di nodi dell'albero

- può portare a spreco di risorse
- nel caso peggiore, un albero con n nodi richiede un vettore con 2^n-1 elementi (se l'albero degenera in una catena)

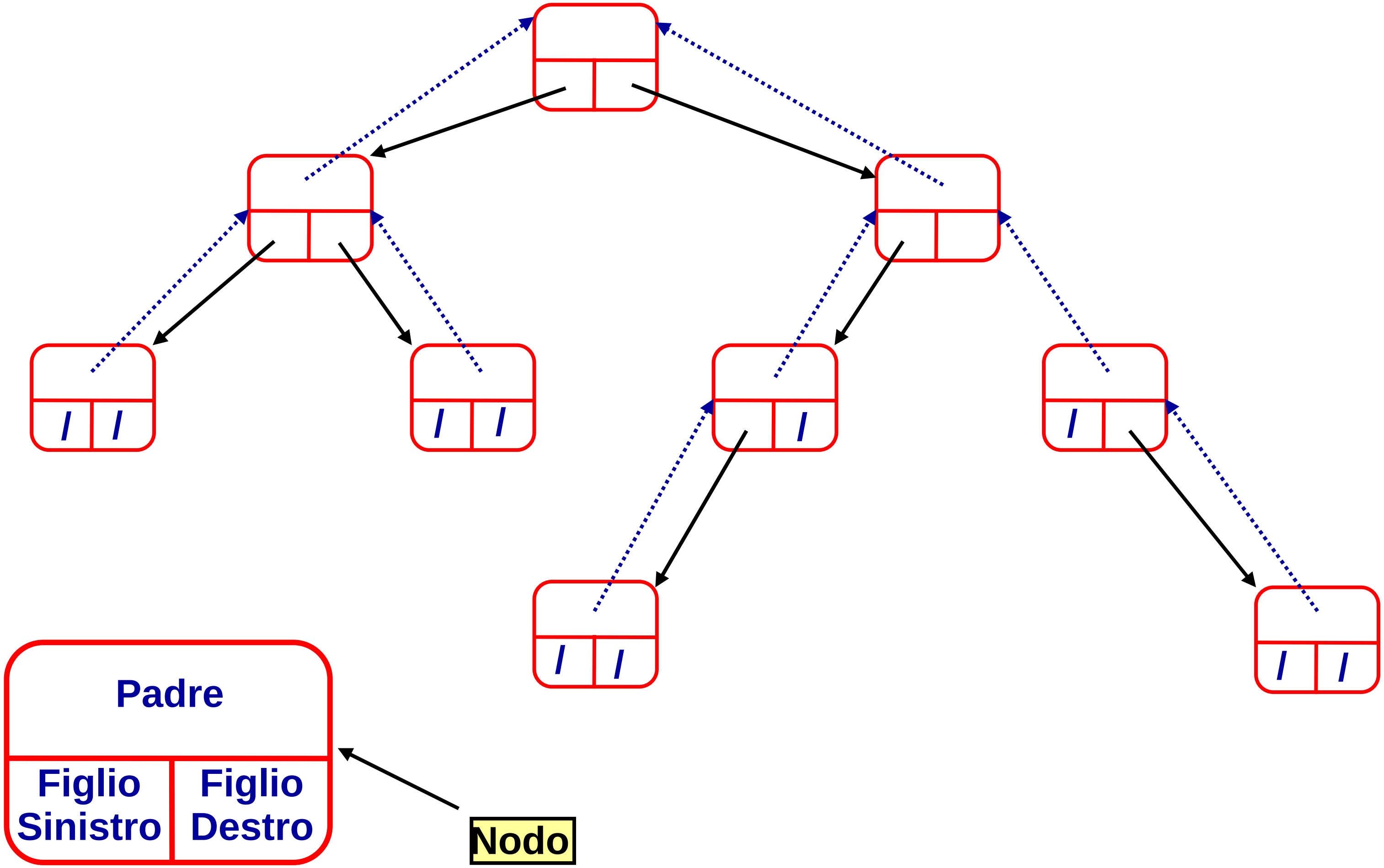
Inoltre poiché alcune componenti del vettore non corrispondono ad alcun nodo dell'albero, in caso di realizzazione con linguaggi a tipizzazione forte, non avendo modo di avvalorare elementi mancanti con un carattere tipo ~~b~~, si sarebbe costretti ad usare un vettore con elementi a due campi, uno indicante il valore l'altro un booleano per attestare la validità.

1	VERO	8
2	FALSO	24
3	VERO	5
4	FALSO	62
5	FALSO	3
6	VERO	27
7	VERO	16

UNA POSSIBILE REALIZZAZIONE IN C

```
#define MaxNodiAlbero…
Typedef ..TipoInfoAlbero;
Strucy StructAlbero {
   TipoInfoAlbero info;
   bool esiste;
}
typedef struct StructAlbero   TipoNodoAlbero;
Typedef TipoNodoAlbero TipoAlbero[MaxNodiAlbero];
```

Alberi binari: rappresentazione collegata

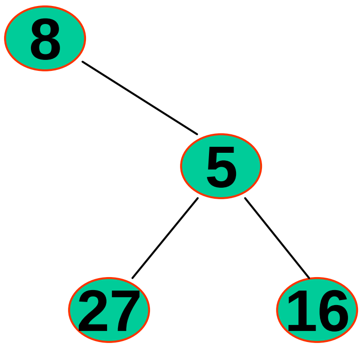


LA RAPPRESENTAZIONE COLLEGATA CON ARRAY

3

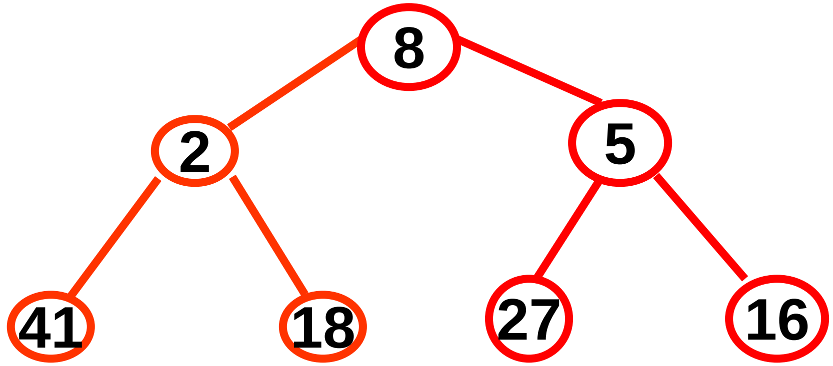
INIZIO

	sinistro	nodo	destro	padre
1	0	16	0	7
2				
3	0	8	7	0
4				
5	0	27	0	7
6				
7	5	5	1	3



SI UTILIZZA UN ARRAY: AD OGNI NODO DELL'ALBERO CORRISPONDE UNA COMPONENTE DELL'ARRAY IN CUI SONO MEMORIZZATE LE INFORMAZIONI (NODO/ETICHETTA, RIFERIMENTO A FIGLIO SINISTRO, RIFERIMENTO A FIGLIO DESTRO, EVENTUALMENTE, RIFERIMENTO A PADRE). SE IL NODO NON ESISTE IL RIFERIMENTO HA VALORE ZERO.

SE VOLESSIMO COMPLETARE L'ALBERO



3

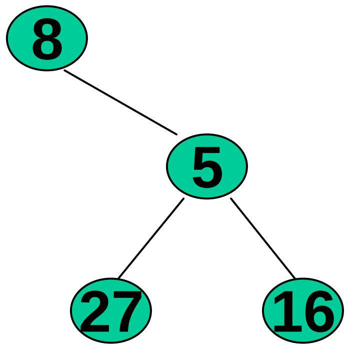
INIZIO

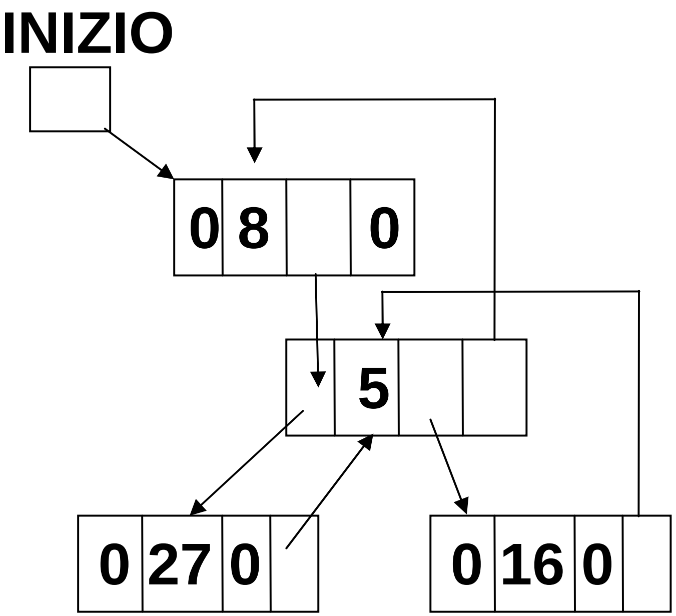
1	0	16	0	7
2	0	18	0	6
3	4	8	7	0
4	6	2	2	3
5	0	27	0	7
6	0	41	0	6
7	5	5	1	3

LA RAPPRESENTAZIONE COLLEGATA CON USO DI VARIABILI DINAMICHE

OVVIAMENTE E' POSSIBILE USARE PUNTATORI INVECE CHE CURSORI E LA MANCANZA DI UN FIGLIO VIENE INDICATA COL VALORE nil NELL'APPOSITO CAMPO. PREVEDIAMO UN CAMPO PER IL FIGLIO DESTRO, UNO PER IL FIGLIO SINISTRO E, PER RAGIONI DI EFFICIENZA UN CAMPO PER IL PADRE

(8 () (5 (27 () ()) (16 () ())))





PROBLEMA: NUMERO NODI PER SOTTOALBERO

DATO UN ALBERO BINARIO T, NON VUOTO, SI MEMORIZZI NELL'ETICHETTA DI OGNI NODO u IL NUMERO DI NODI CHE SI TROVANO NEL SOTTOALBERO CON RADICE IN u.

```
CONTANODI(U:nodo; T:binalbero)
  if (SINISTROVUOTO(U,T)) and (DESTROVUOTO(U,T))
then
      CONTO ← 1
      SCRIVINODO(CONTO,U,T)
  else
      if not SINISTROVUOTO(U,T) then
         CONTANODI(FIGLIOSINISTRO(U,T),T)
         SOMMASIN ← LEGGINODO(FIGLIOSINISTRO(U,T),T)
      else
         SOMMASIN ← 0
      if not DESTROVUOTO(U,T) then
         CONTANODI(FIGLIODESTRO(U,T),T)
         SOMMADES ← LEGGINODO(FIGLIODESTRO(U,T),T)
      else
         SOMMADES ← 0
      CONTO ← SOMMASIN+SOMMADES+1
      SCRIVINODO(CONTO,U,T)
```

ANCORA SU VISITE DI ALBERI

LA VISITA IN PRE-ORDINE E' DI FATTO UNA VISITA IN PROFONDITA', VALE A DIRE CHE L'ALBERO DI OGNI ORDINE VIENE VISITATO DALLA RADICE FINO AI NODI TERMINALI, DA SINISTRA VERSO DESTRA.

QUESTA DENOMINAZIONE DI VISITA E' USATA PER I GRAFI, MA ESSENDO L'ALBERO UN TIPO PARTICOLARE DI GRAFO POSSIAMO UTILIZZARE LA MEDESIMA DIZIONE.

SEMPRE PER I GRAFI E' DEFINITA LA VISITA IN AMPIEZZA, VALIDA ANCHE PER GLI ALBERI: L'ALBERO VIENE VISITATO A PARTIRE DALLA RADICE PER LIVELLI SUCCESSIVI, CON L'IMPOSIZIONE CHE OGNI NODO VENGA VISITATO UNA SOLA VOLTA.

LA VISITA IN AMPIEZZA DI UN ALBERO BINARIO

Il metodo proposto è di tipo iterativo e richiede per la visita l’utilizzo di una coda di appoggio in cui vengono memorizzati i riferimenti ai nodi dell’albero dei quali si vuole esaminare le etichette (leggere il contenuto).

PROCEDURA DI VISITA IN AMPIEZZA

L’algoritmo può essere descritto come segue:

Incoda radice albero;

Fino a quando la coda non è vuota ESEGUI

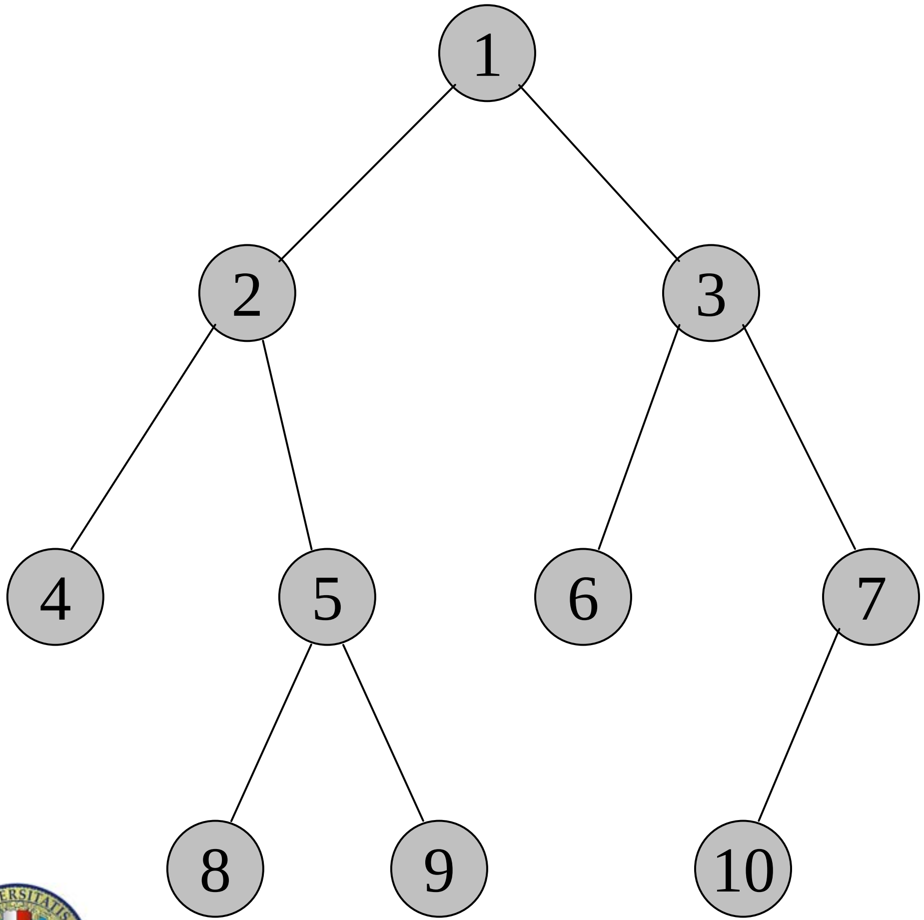
Nodo = Leggicoda e fuoricoda;

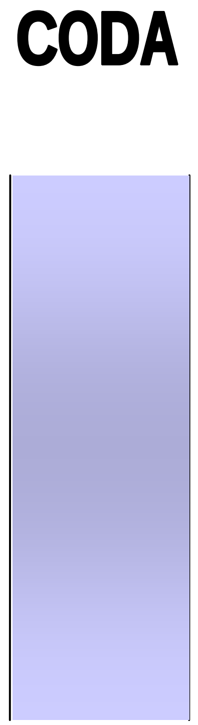
Legginodo e stampane l'etichetta;

Se c’è figliosinistro => incoda figliosinistro;

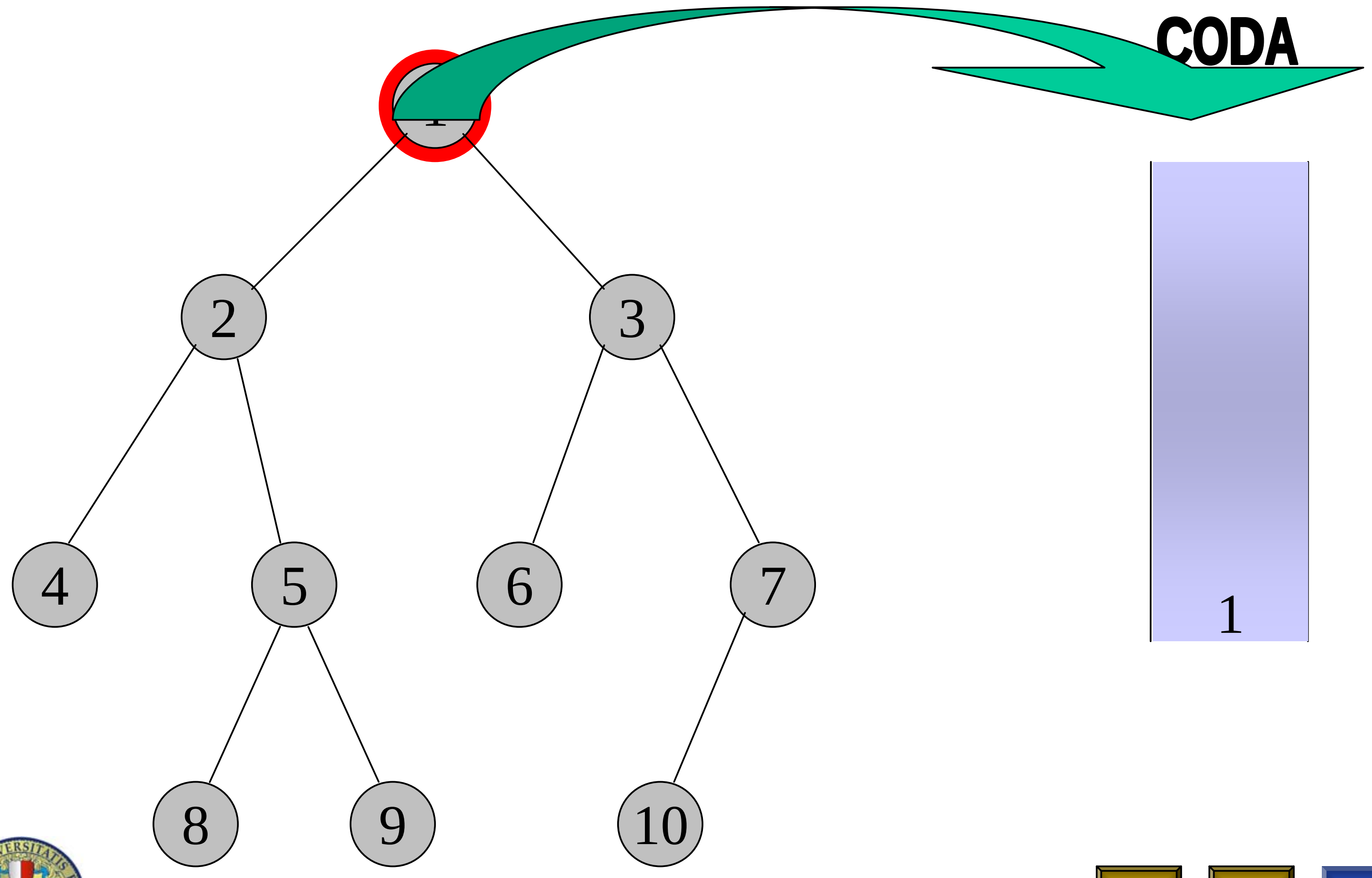
Se c’è figliodestro => incoda figliodestro;

PROCEDURA DI VISITA IN AMPIEZZA

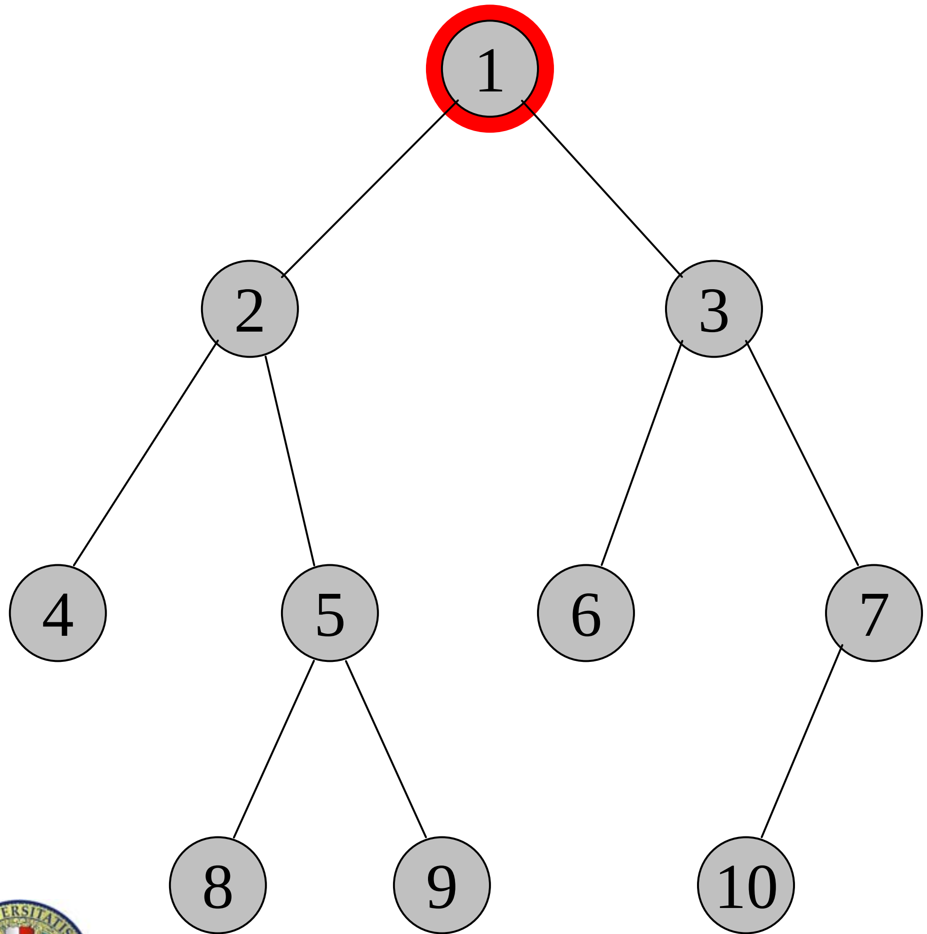


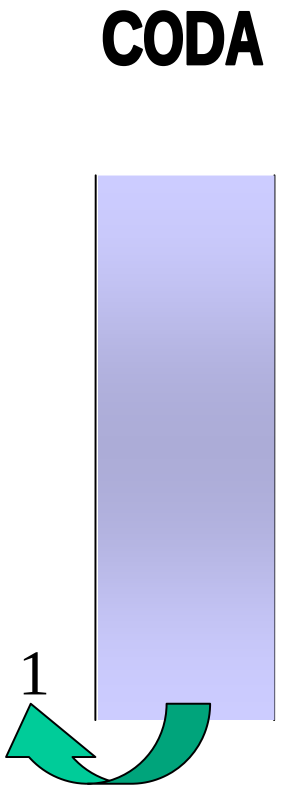


PROCEDURA DI VISITA IN AMPIEZZA

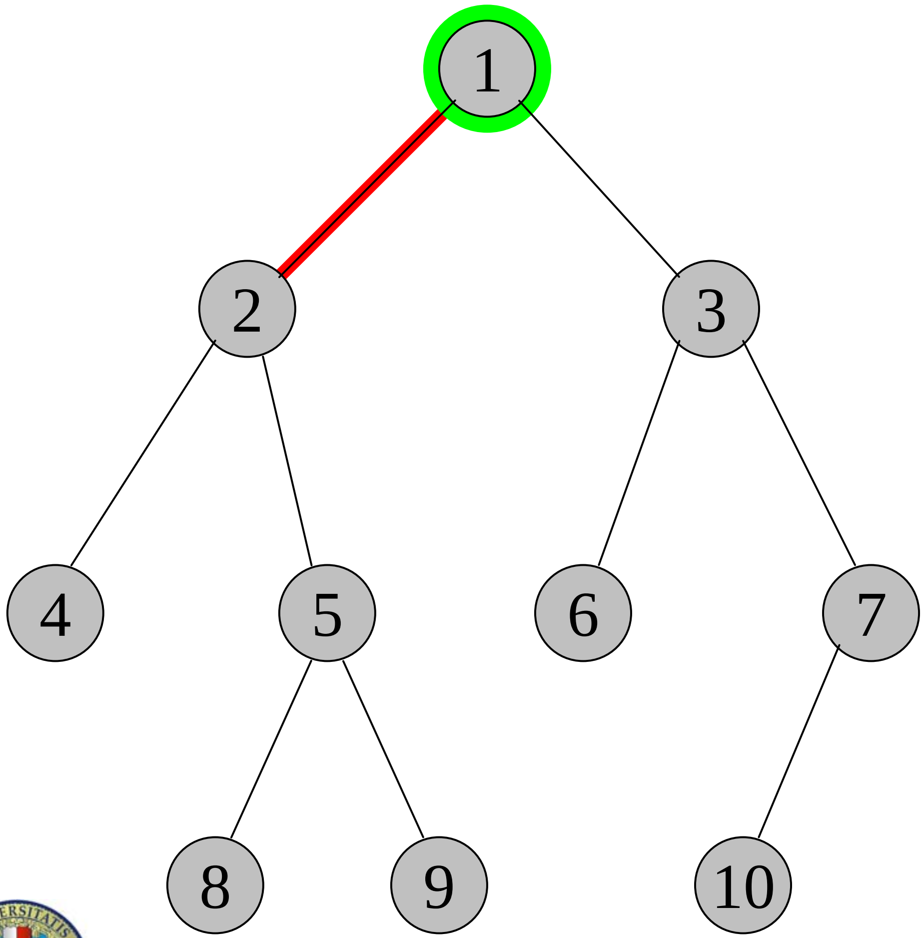


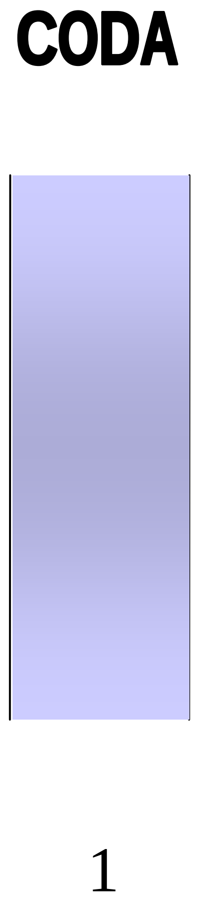
PROCEDURA DI VISITA IN AMPIEZZA



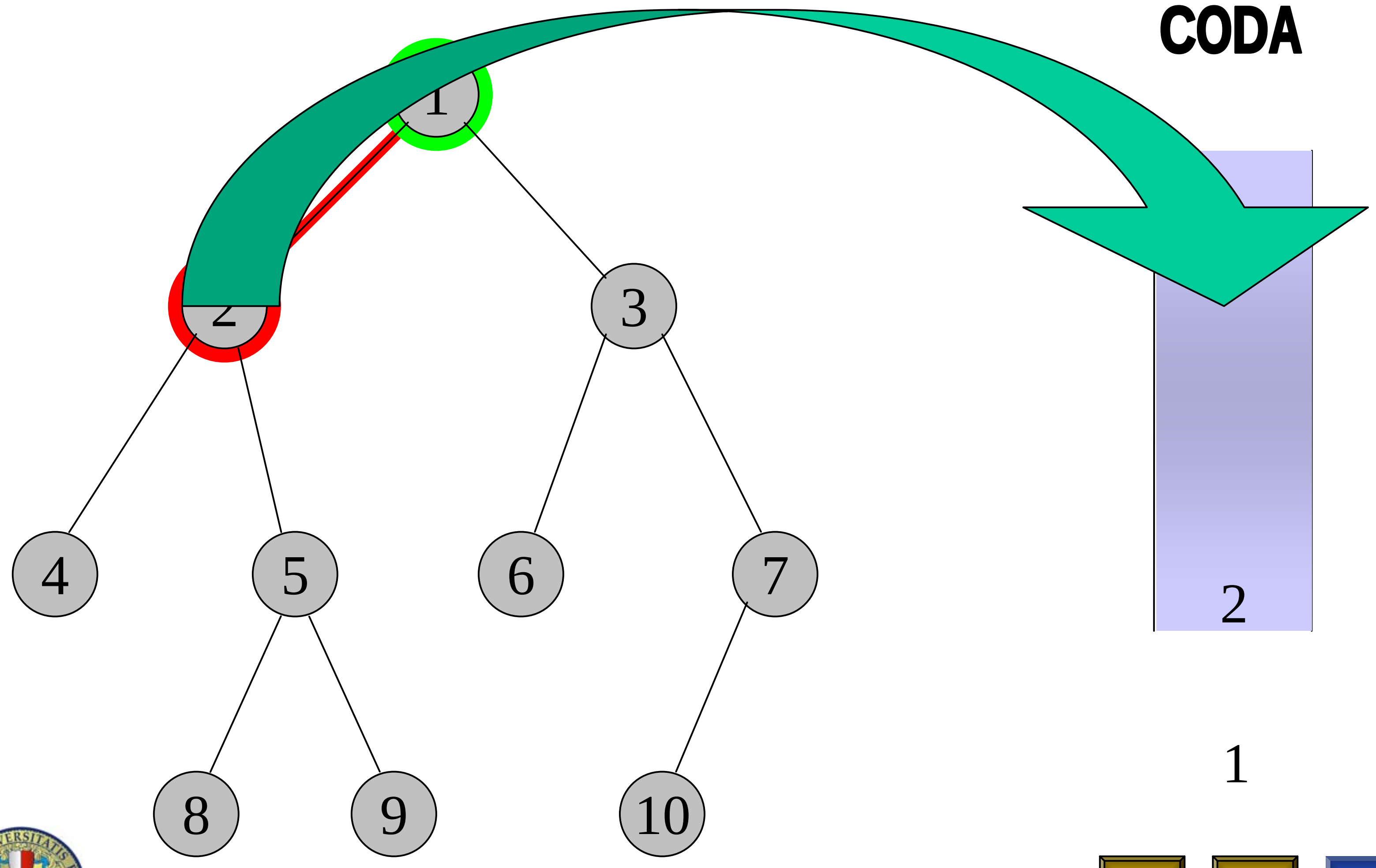


PROCEDURA DI VISITA IN AMPIEZZA

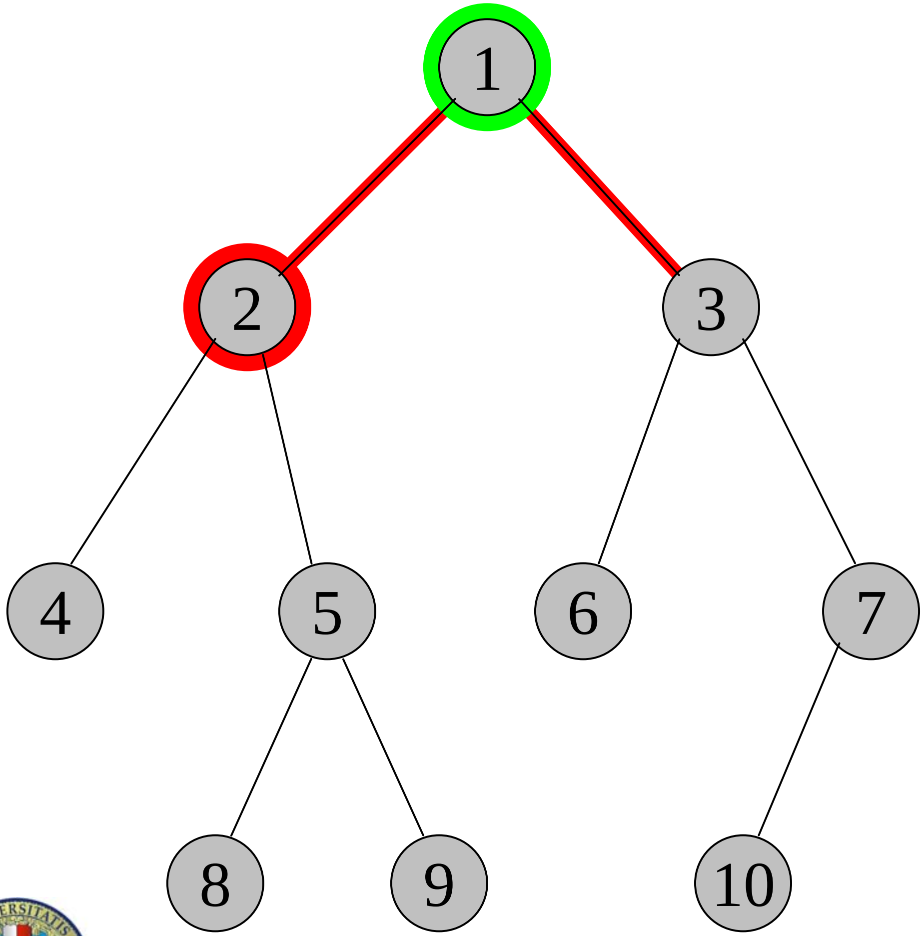


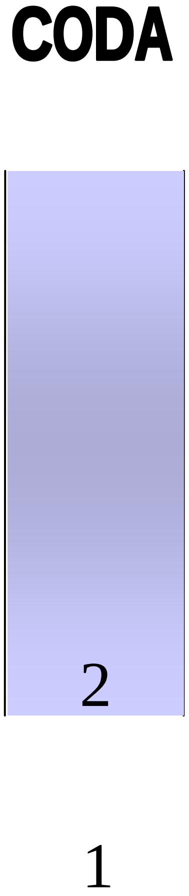


PROCEDURA DI VISITA IN AMPIEZZA

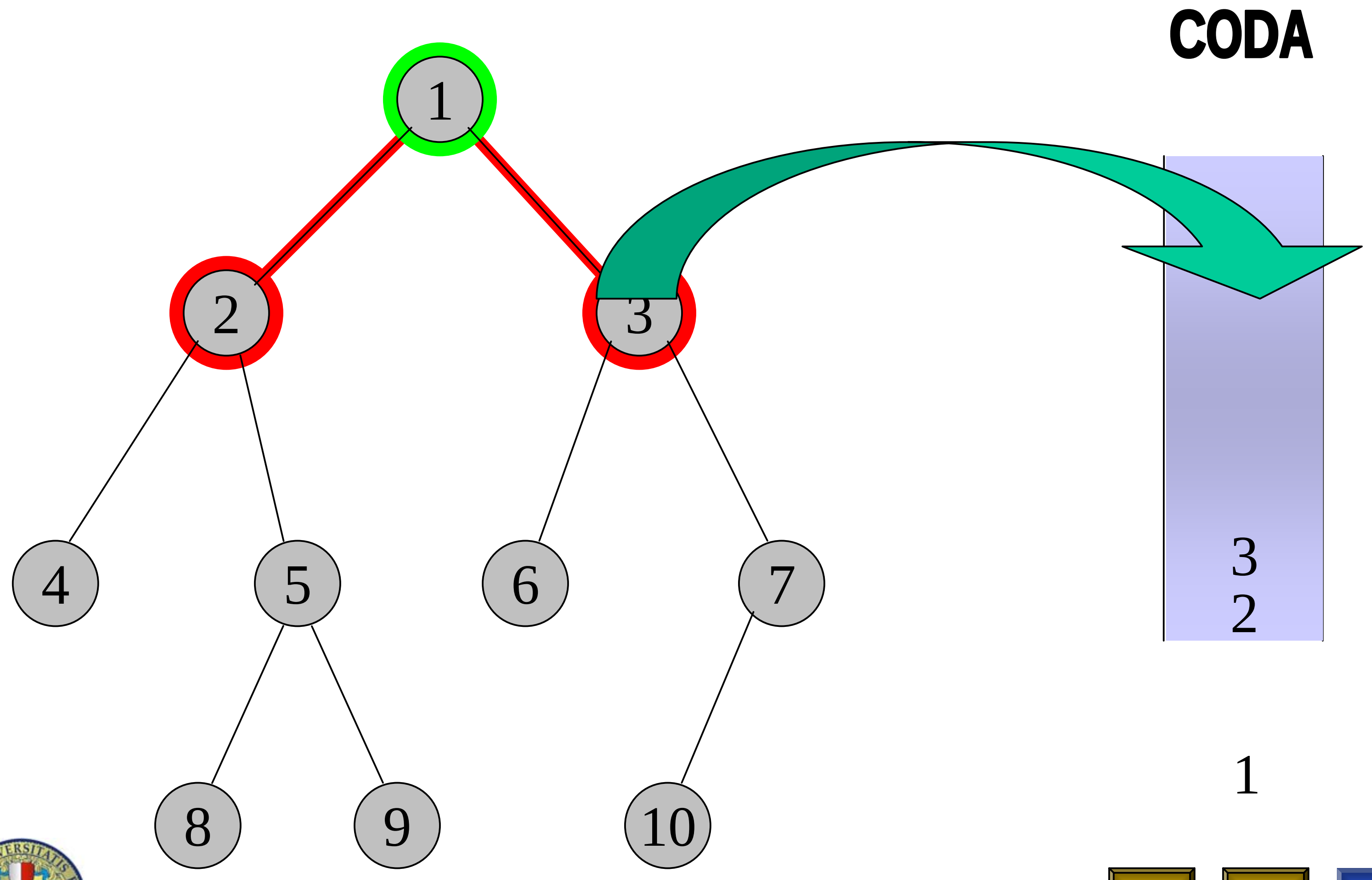


PROCEDURA DI VISITA IN AMPIEZZA

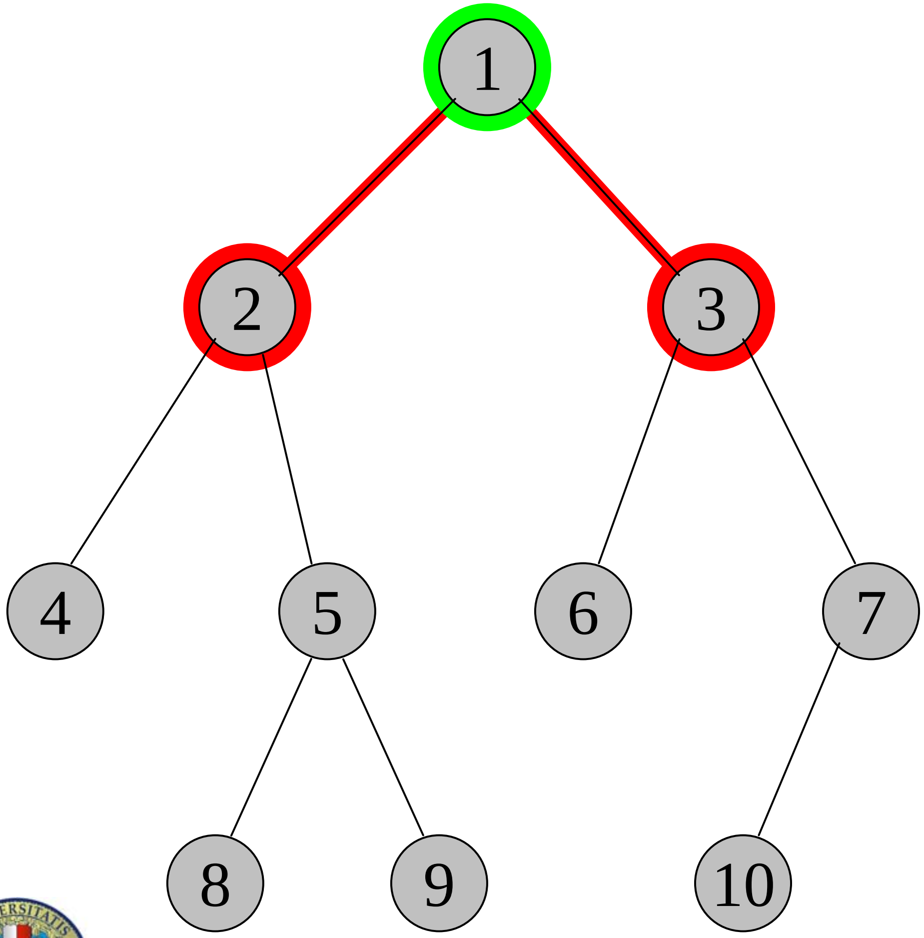


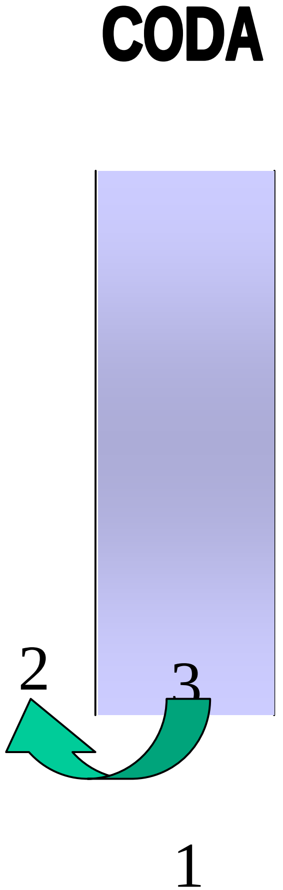


PROCEDURA DI VISITA IN AMPIEZZA

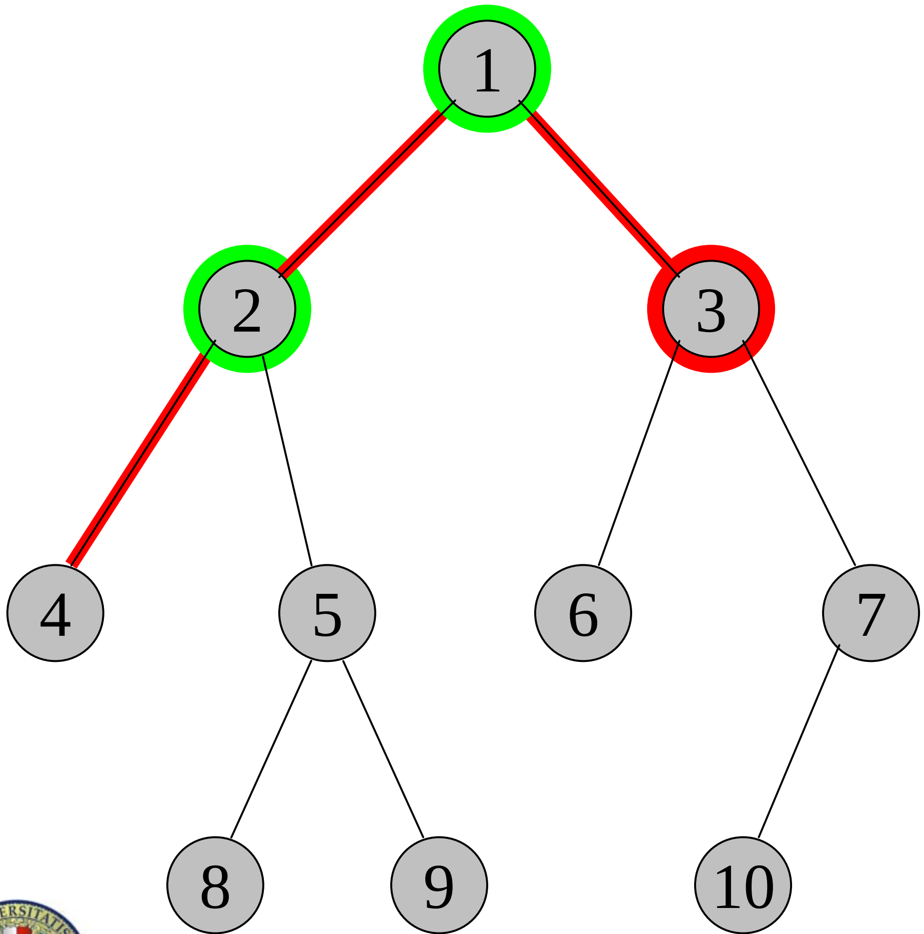


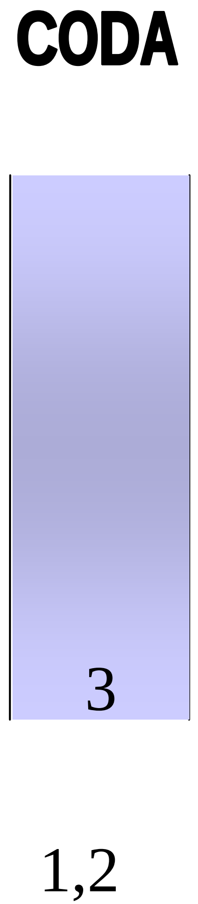
PROCEDURA DI VISITA IN AMPIEZZA



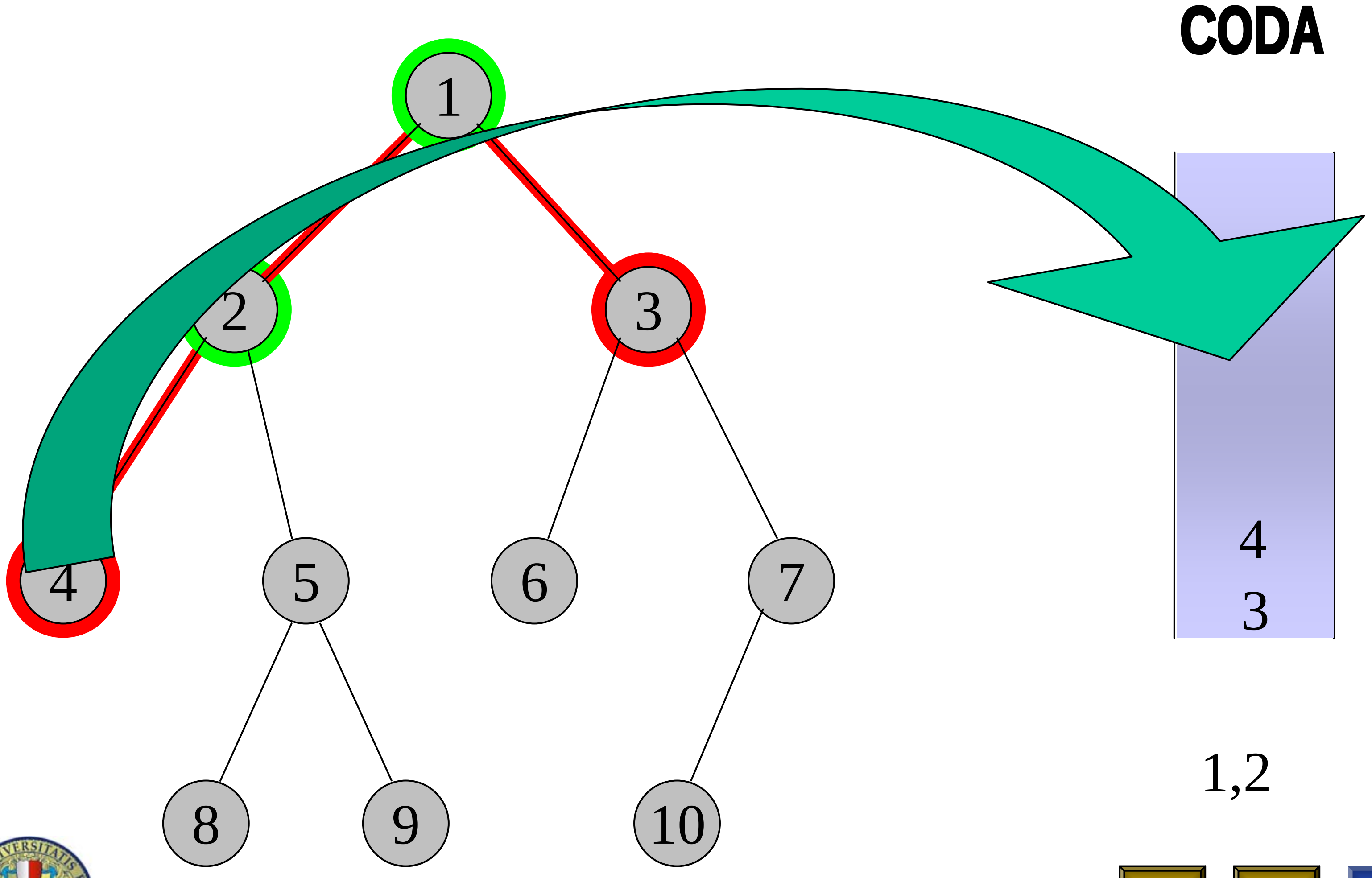


PROCEDURA DI VISITA IN AMPIEZZA

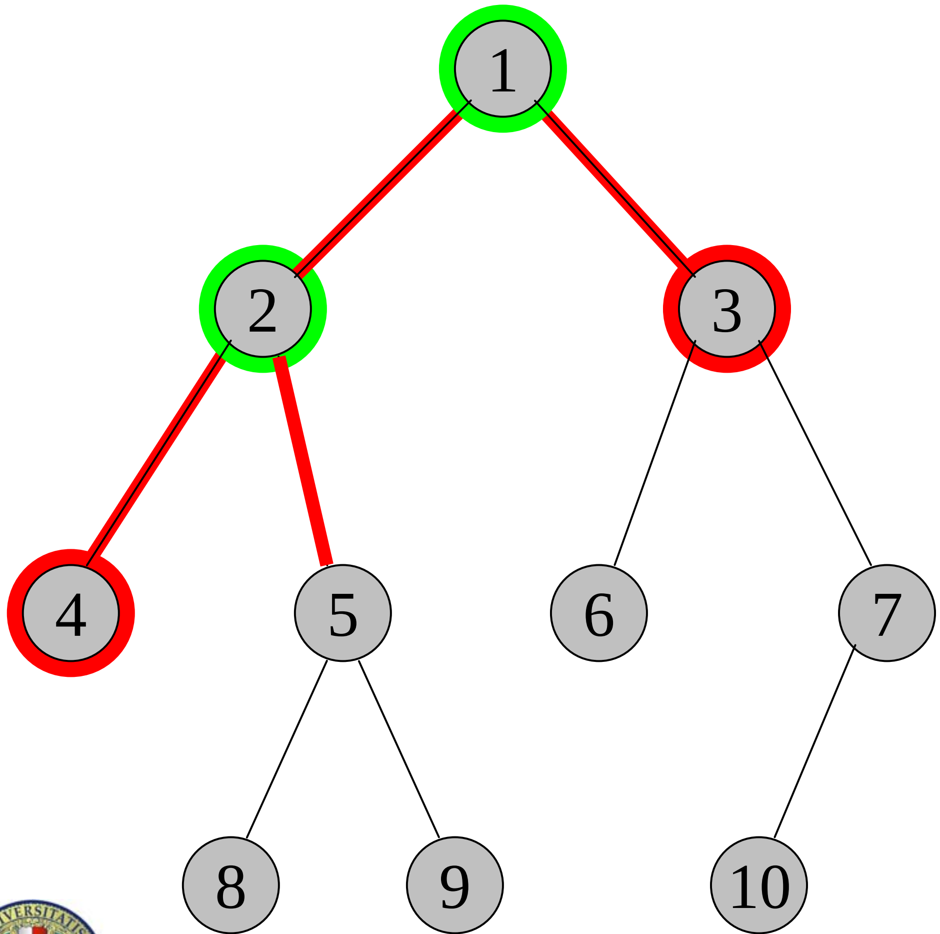


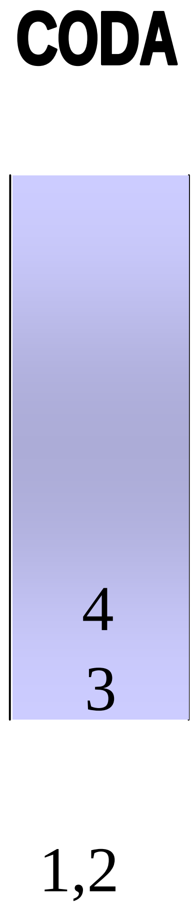


PROCEDURA DI VISITA IN AMPIEZZA

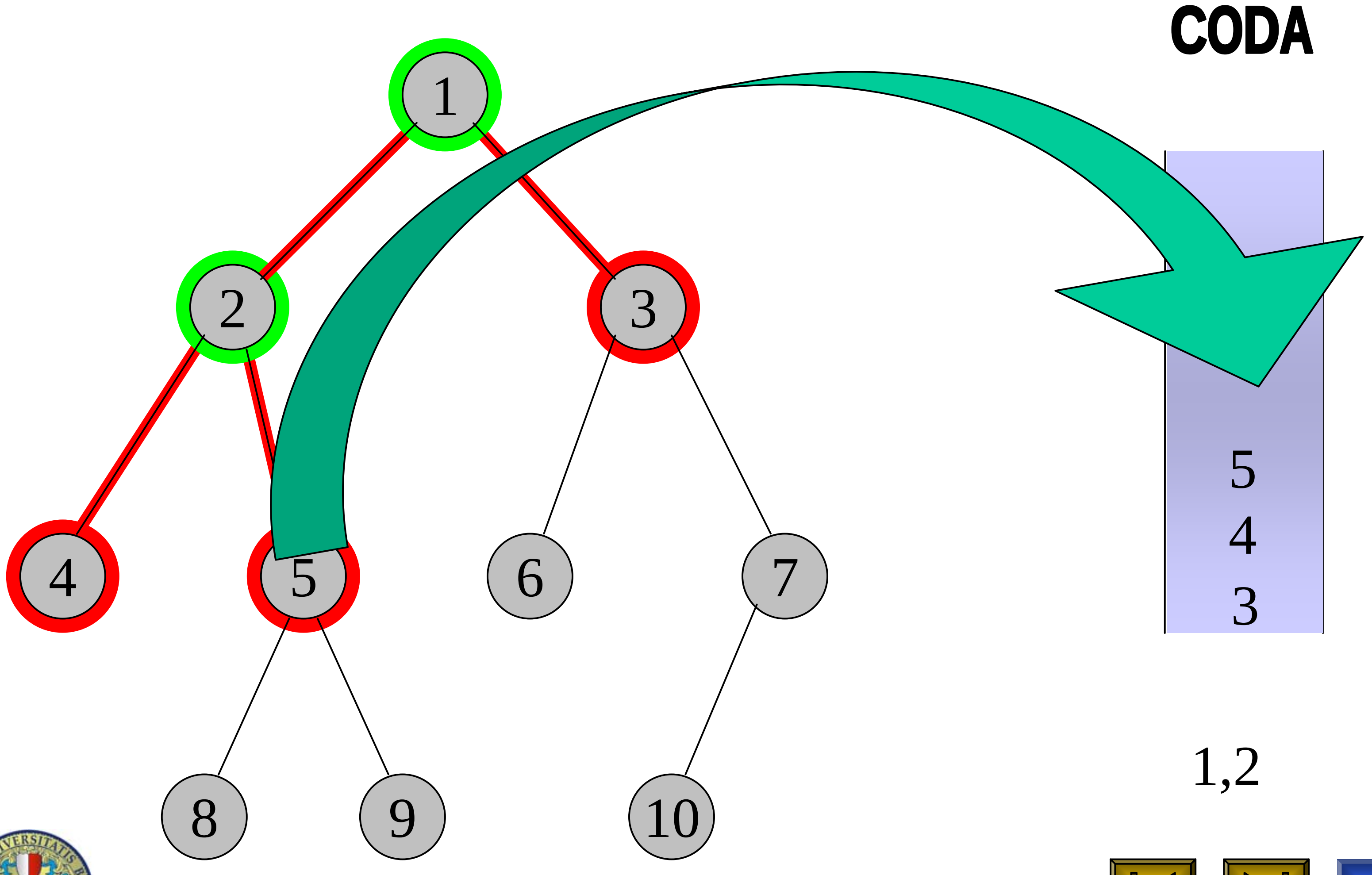


PROCEDURA DI VISITA IN AMPIEZZA

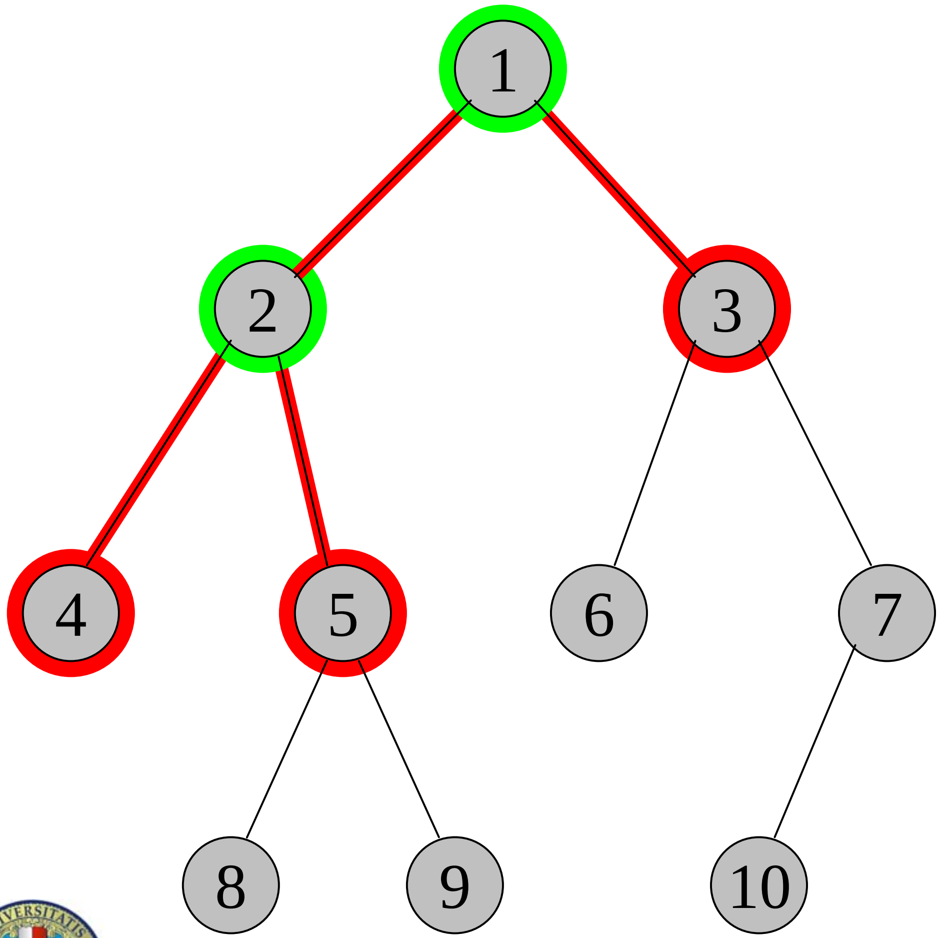


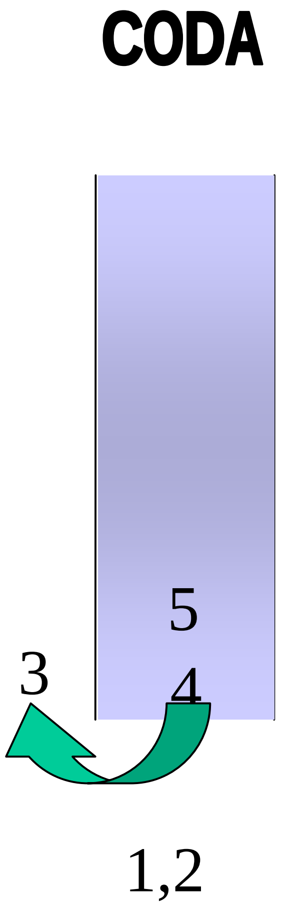


PROCEDURA DI VISITA IN AMPIEZZA

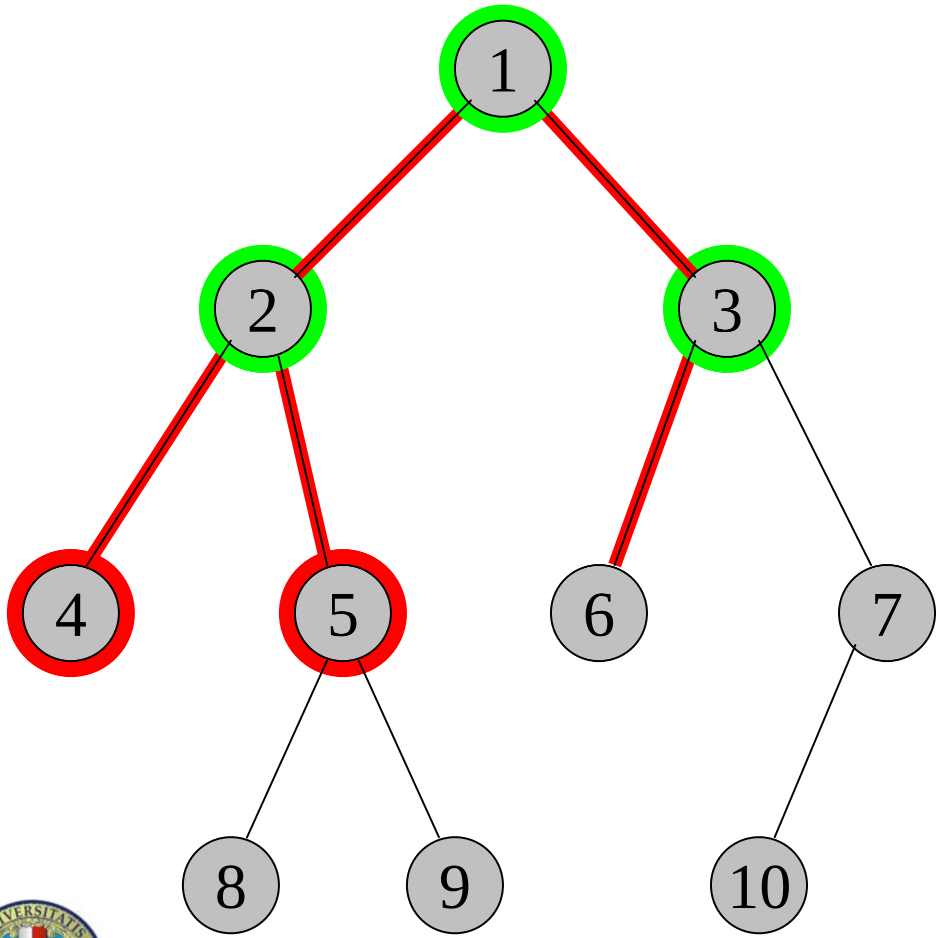


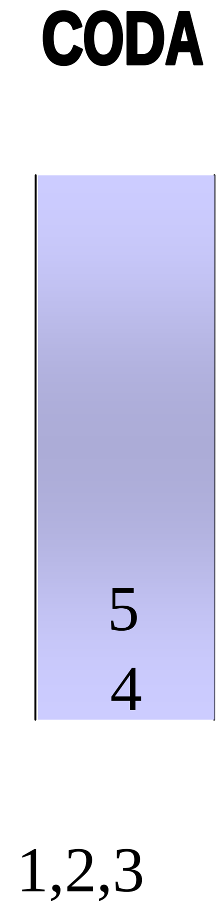
PROCEDURA DI VISITA IN AMPIEZZA



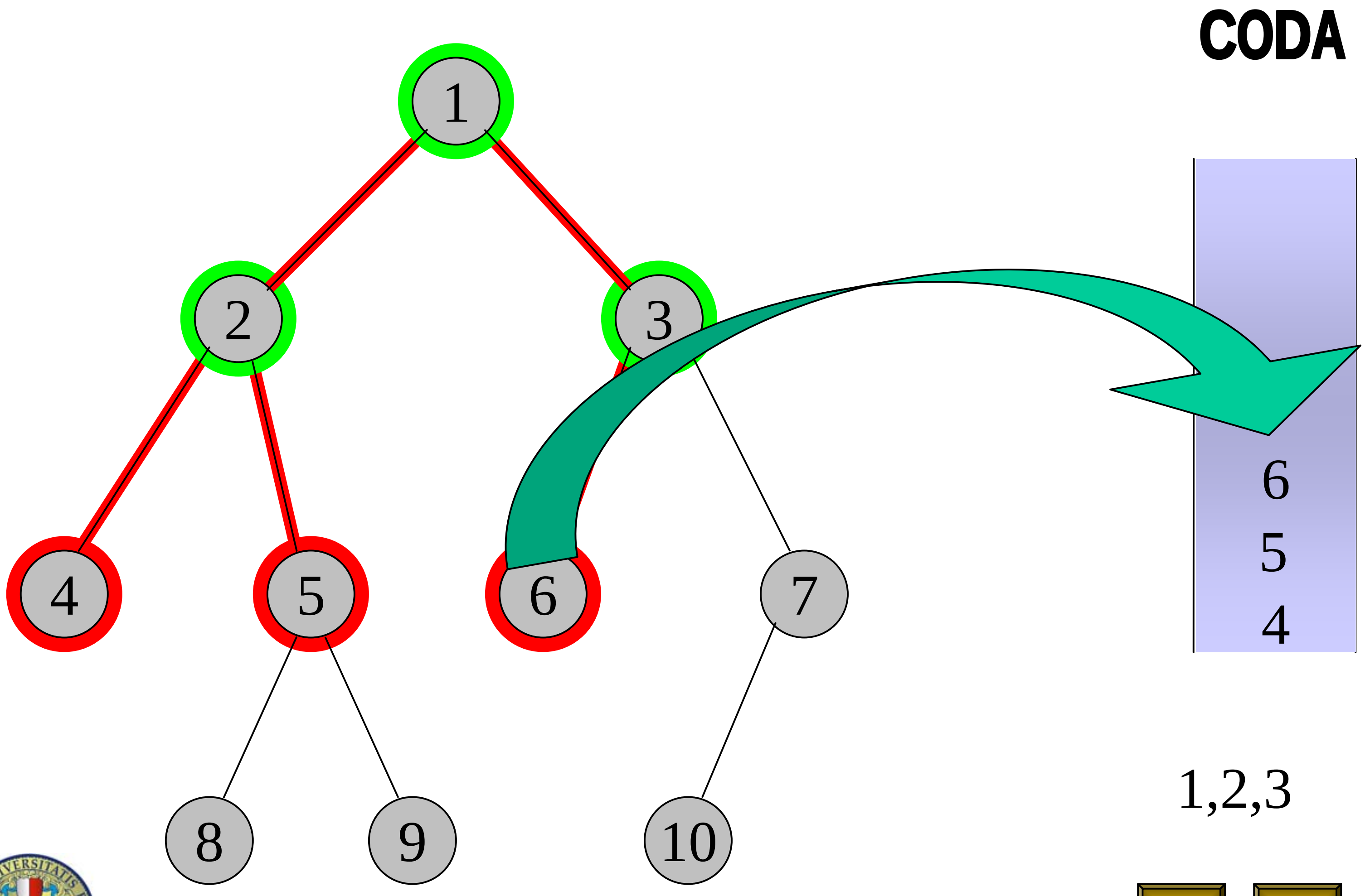


PROCEDURA DI VISITA IN AMPIEZZA

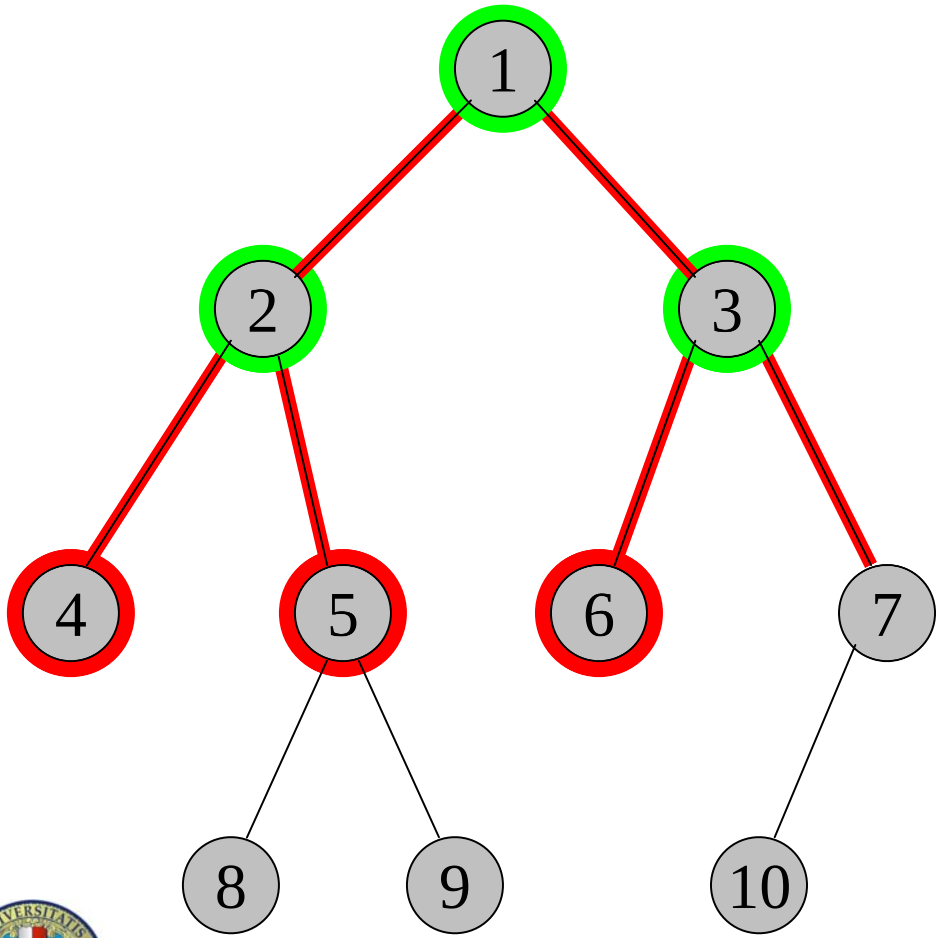


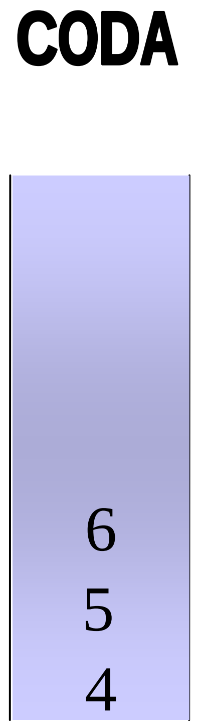


PROCEDURA DI VISITA IN AMPIEZZA



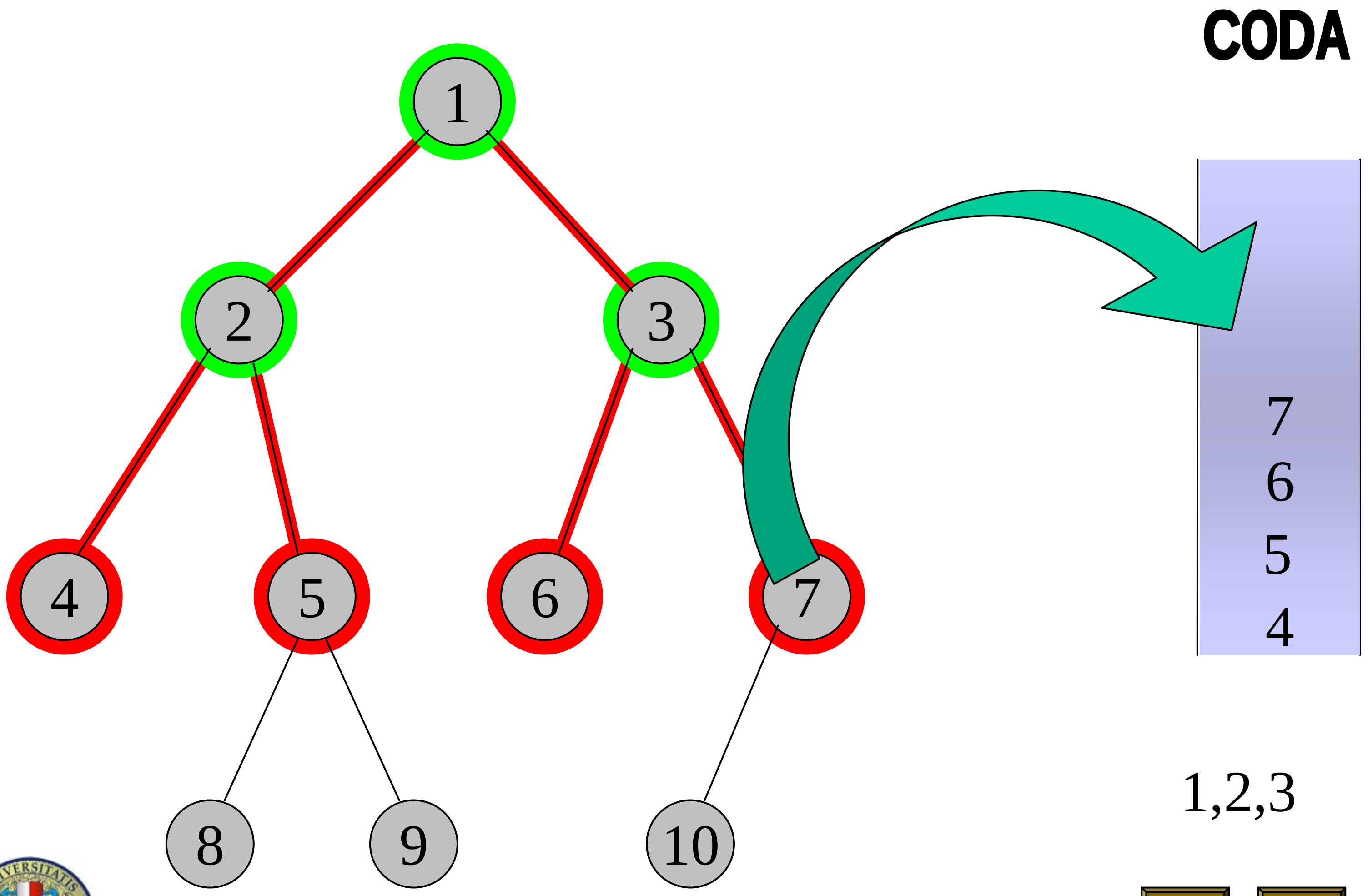
PROCEDURA DI VISITA IN AMPIEZZA



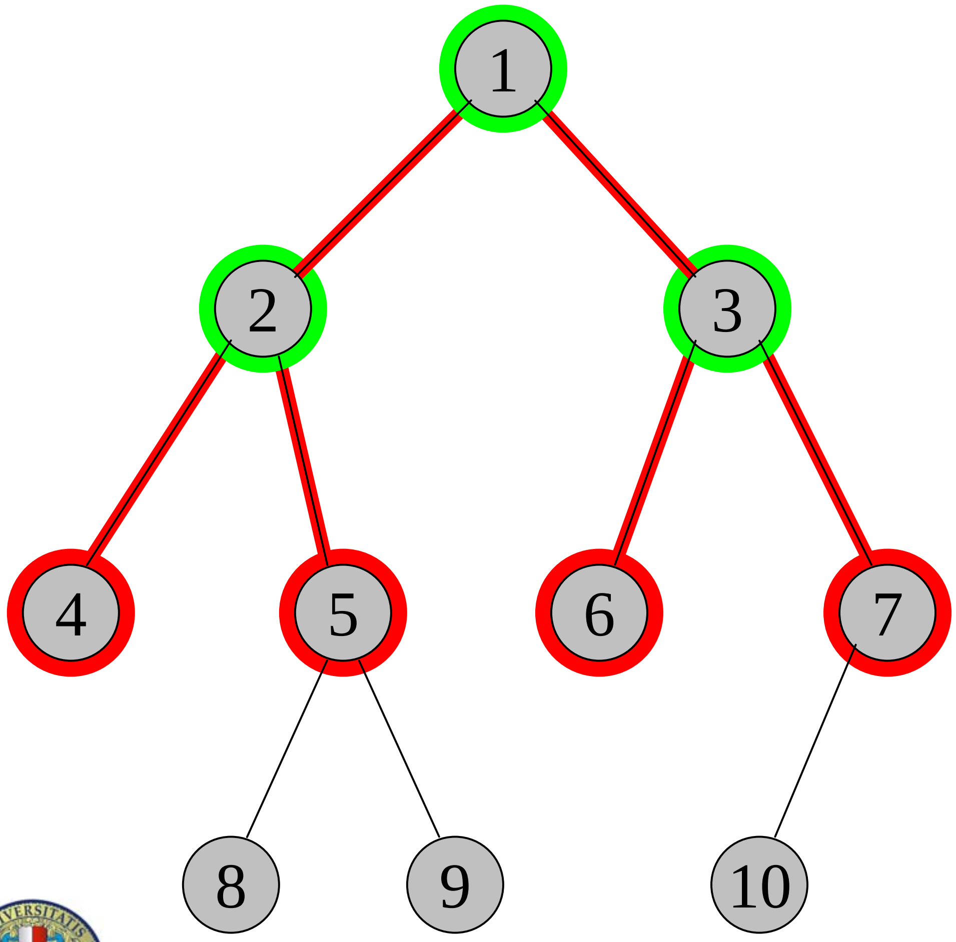


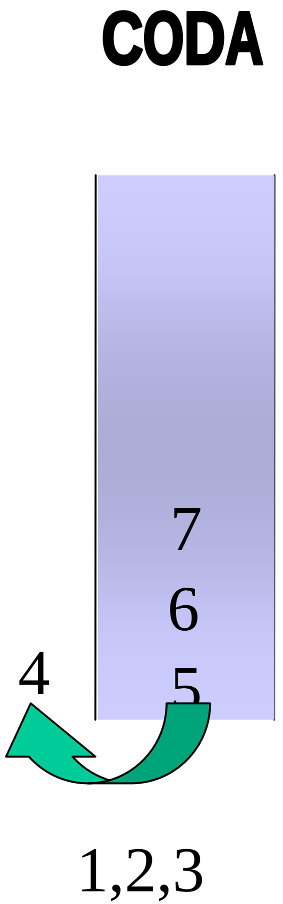
1,2,3

PROCEDURA DI VISITA IN AMPIEZZA

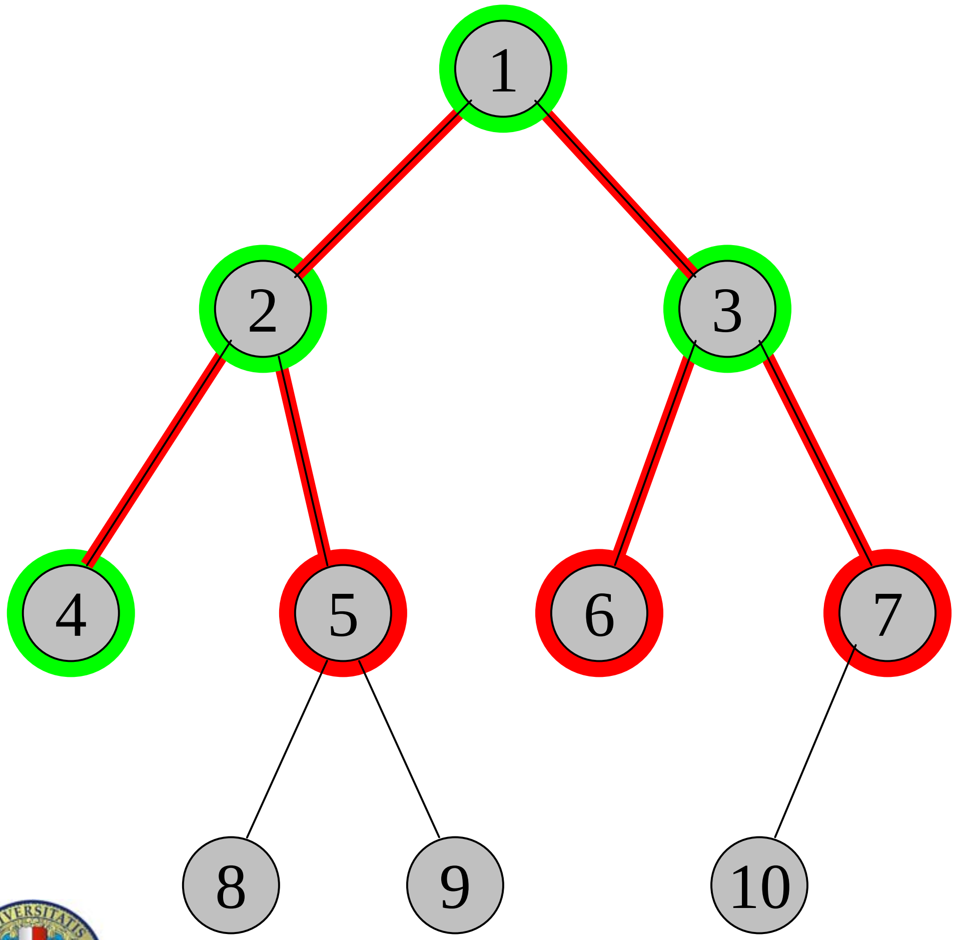


PROCEDURA DI VISITA IN AMPIEZZA

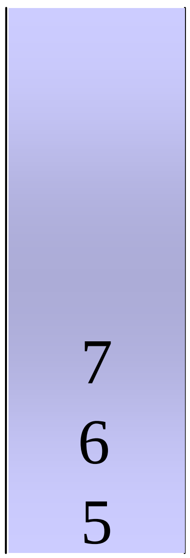




PROCEDURA DI VISITA IN AMPIEZZA

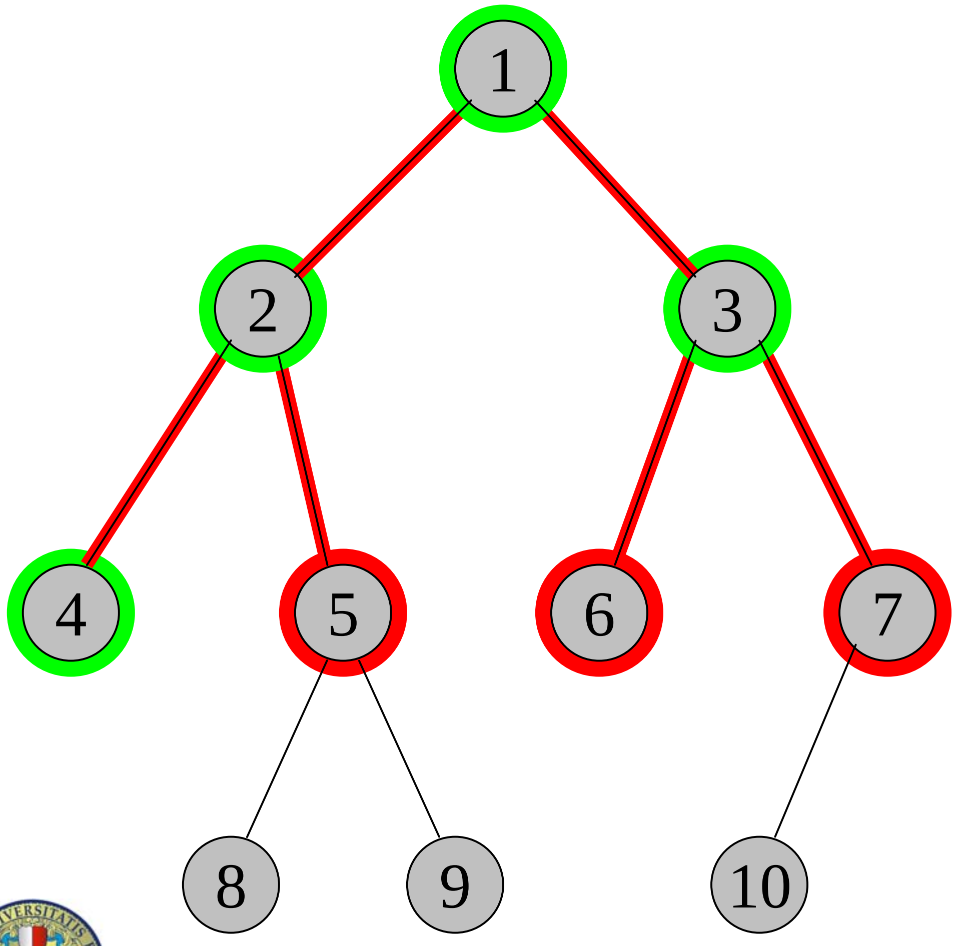


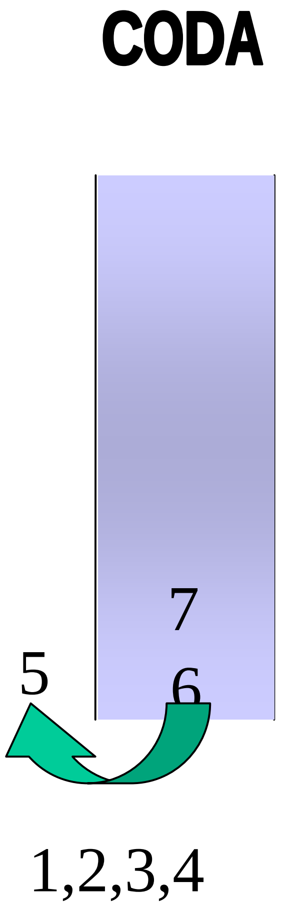
CODA



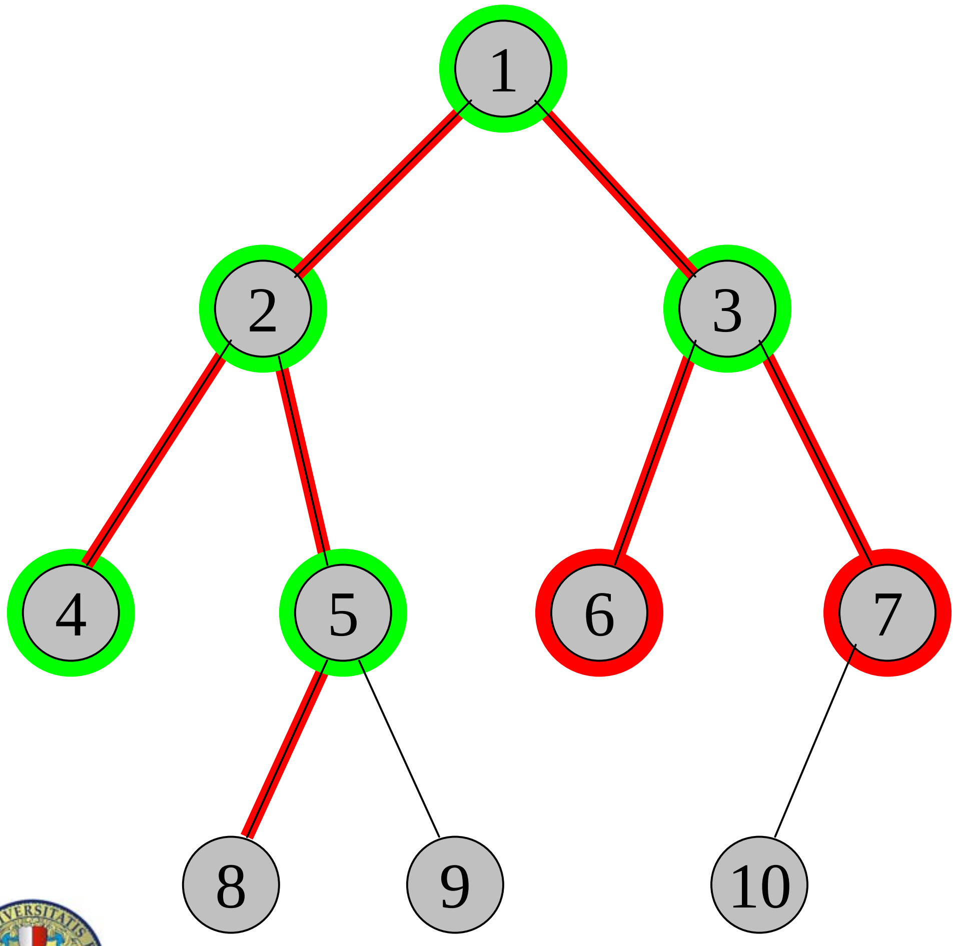
1,2,3,4

PROCEDURA DI VISITA IN AMPIEZZA

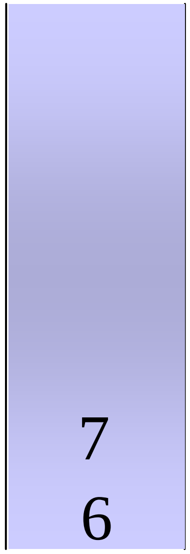




PROCEDURA DI VISITA IN AMPIEZZA

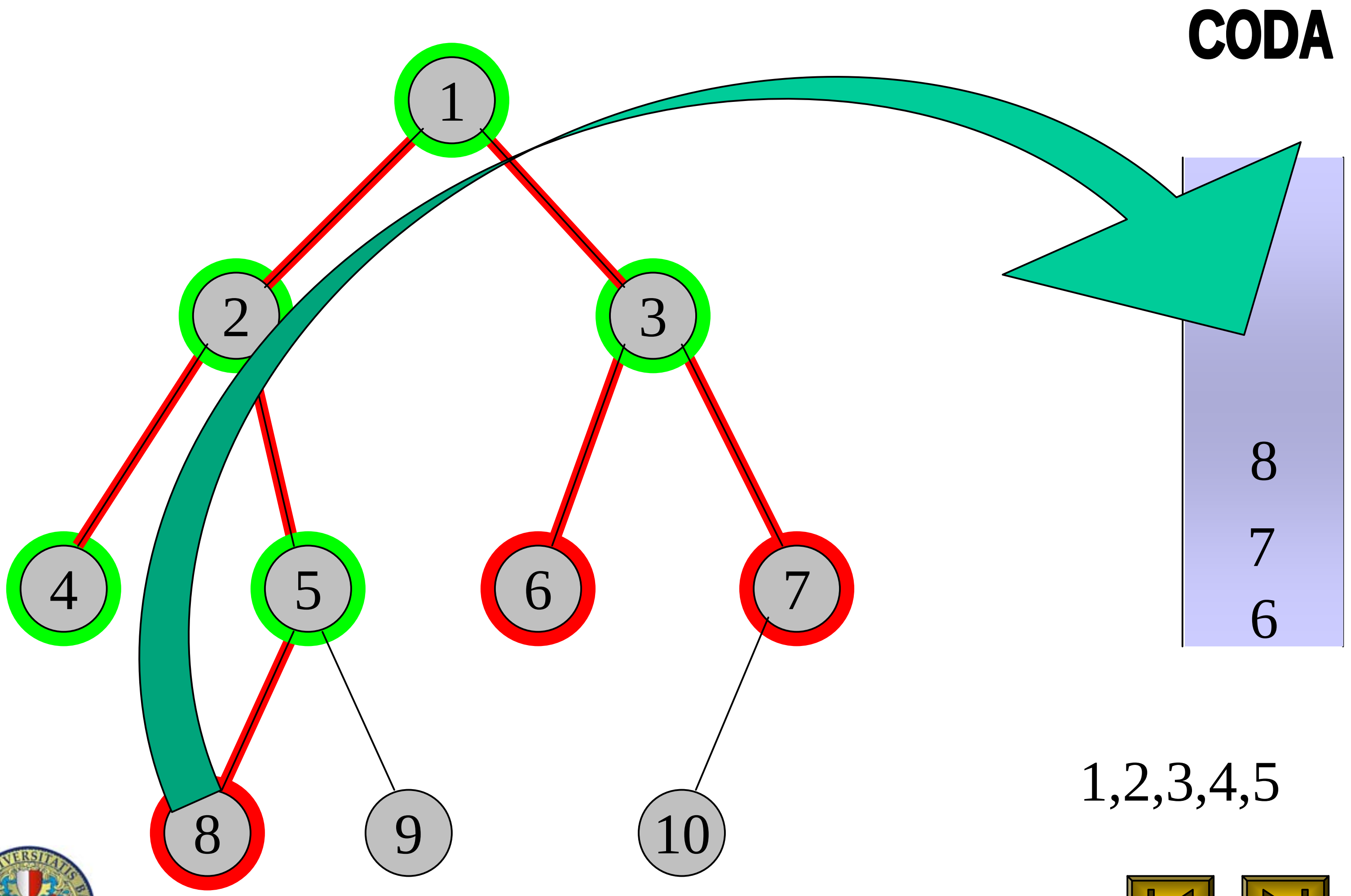


CODA



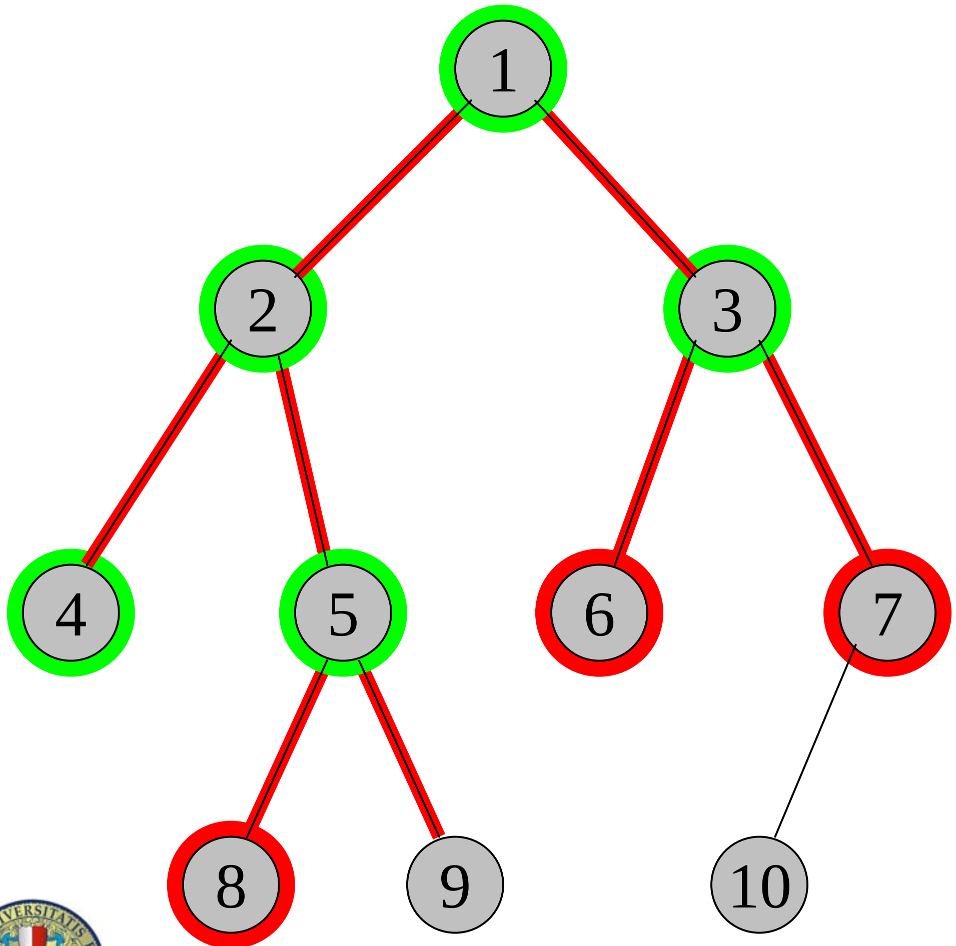
1,2,3,4,5

PROCEDURA DI VISITA IN AMPIEZZA

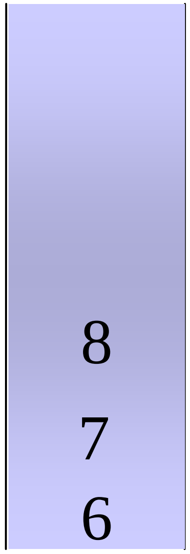


1,2,3,4,5

PROCEDURA DI VISITA IN AMPIEZZA

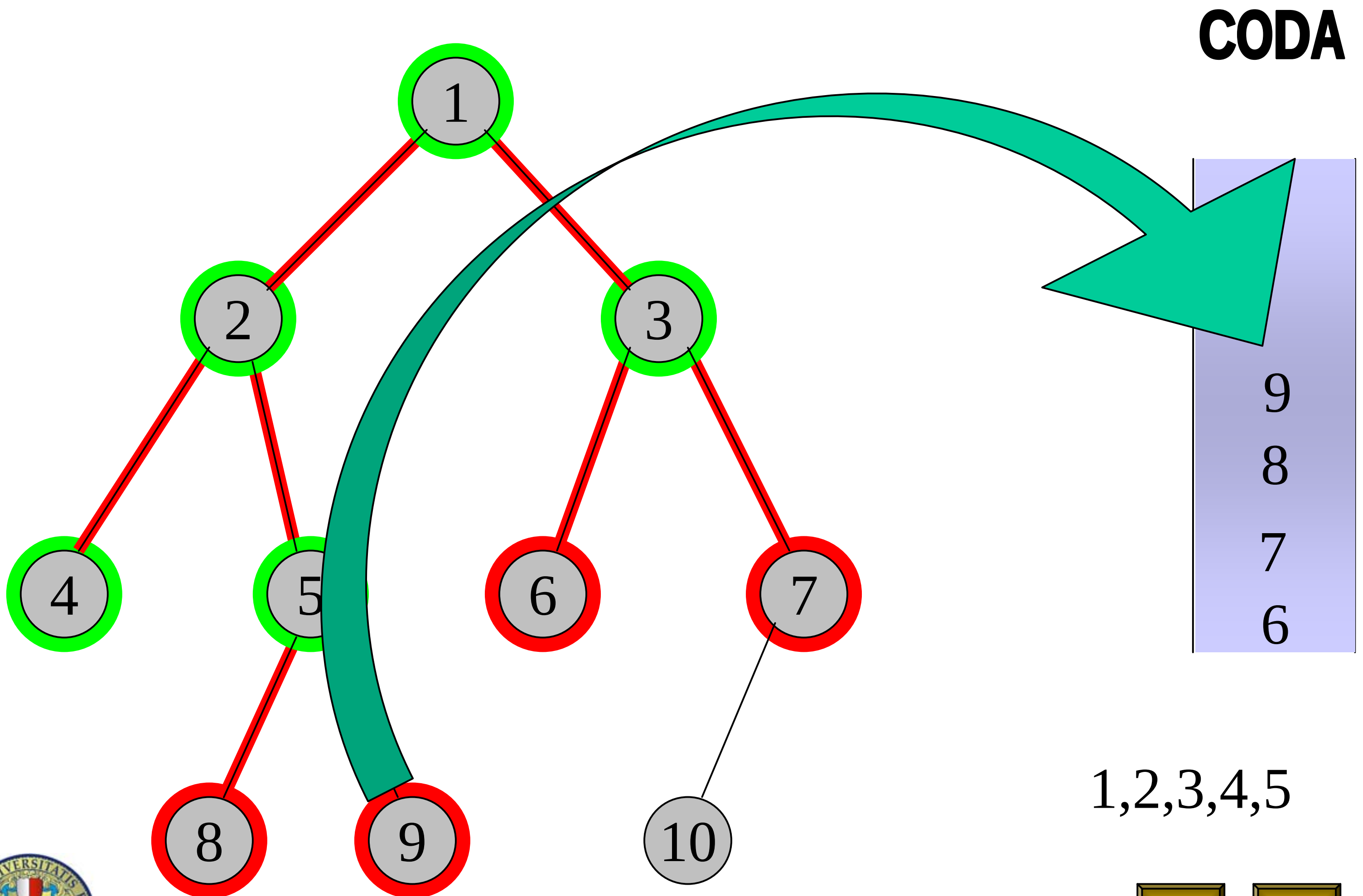


CODA



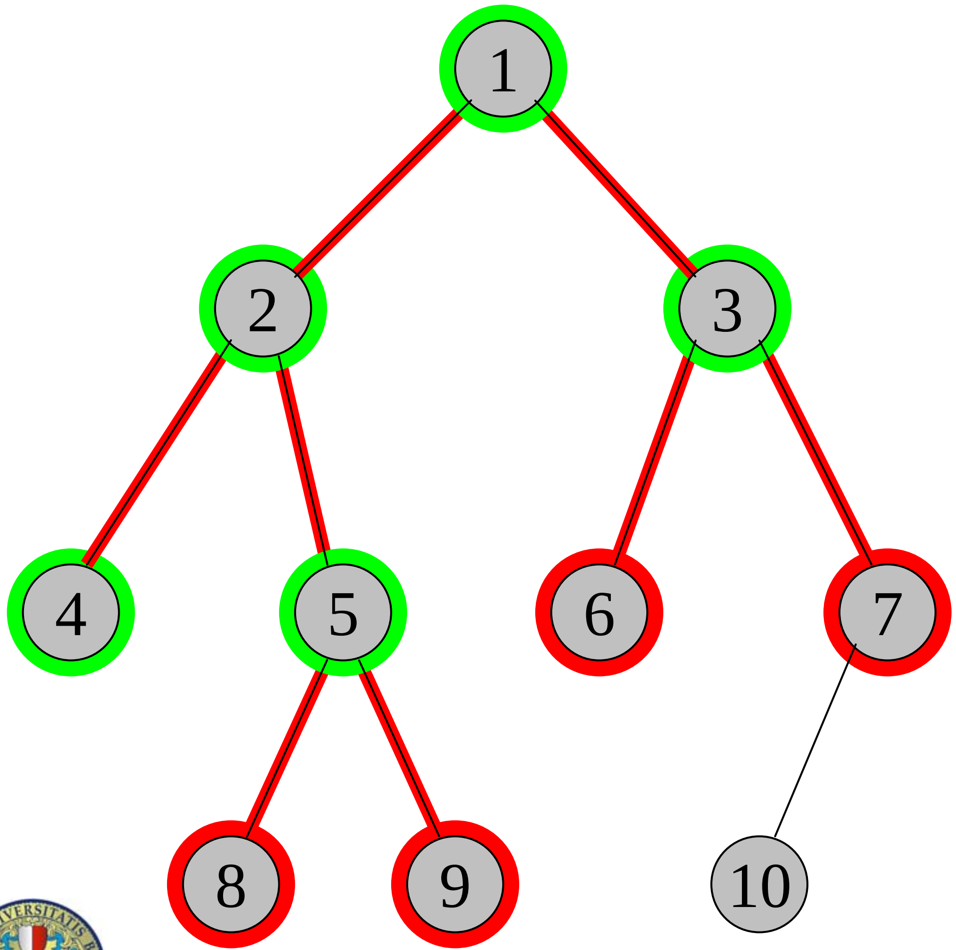
1,2,3,4,5

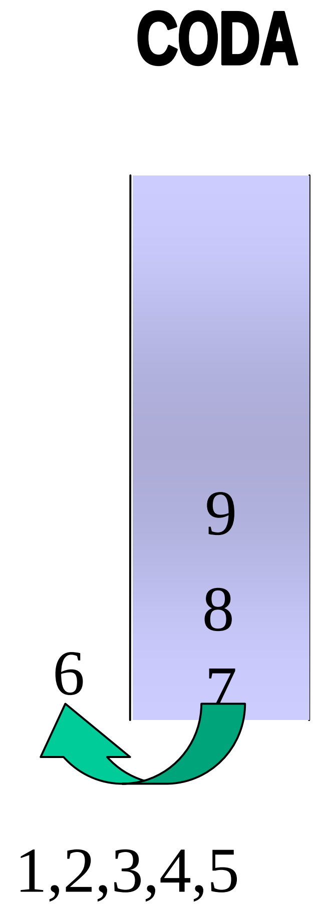
PROCEDURA DI VISITA IN AMPIEZZA



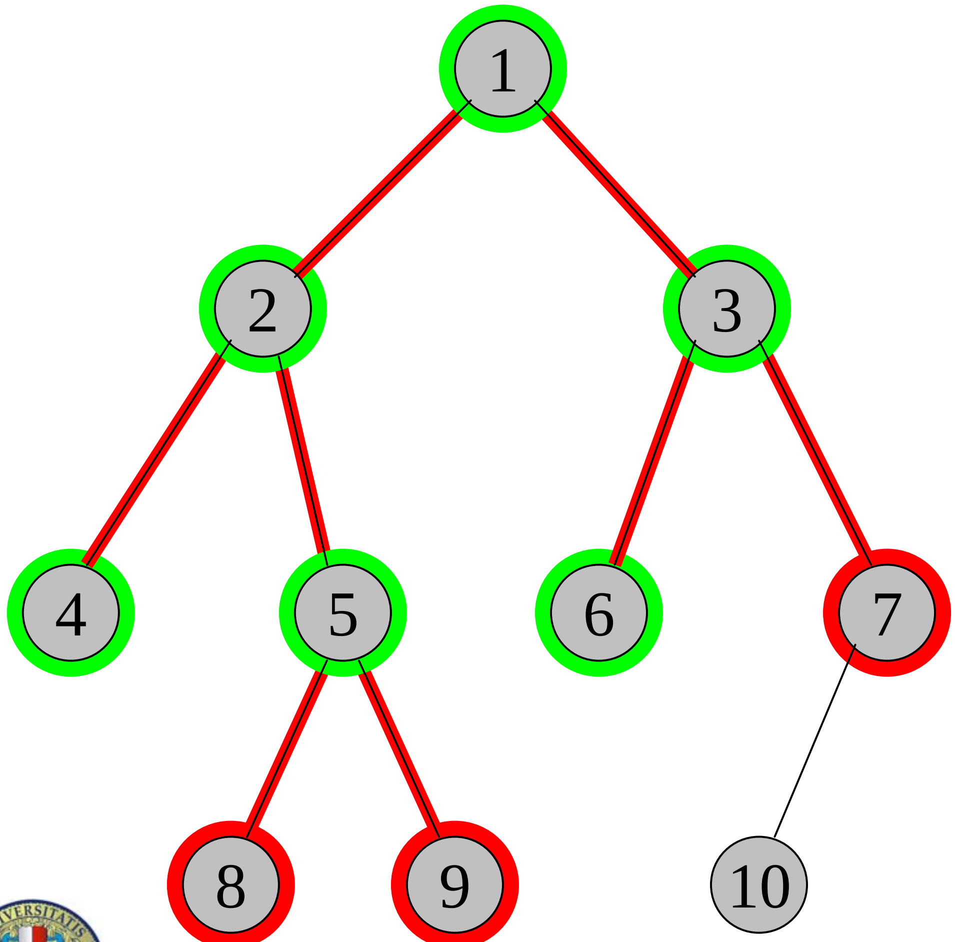
1,2,3,4,5

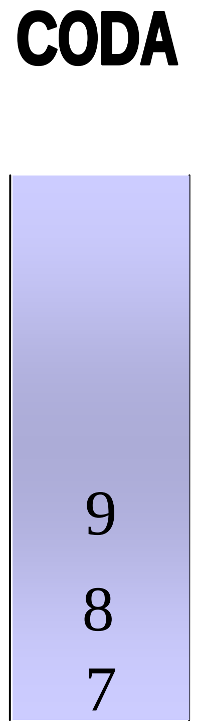
PROCEDURA DI VISITA IN AMPIEZZA





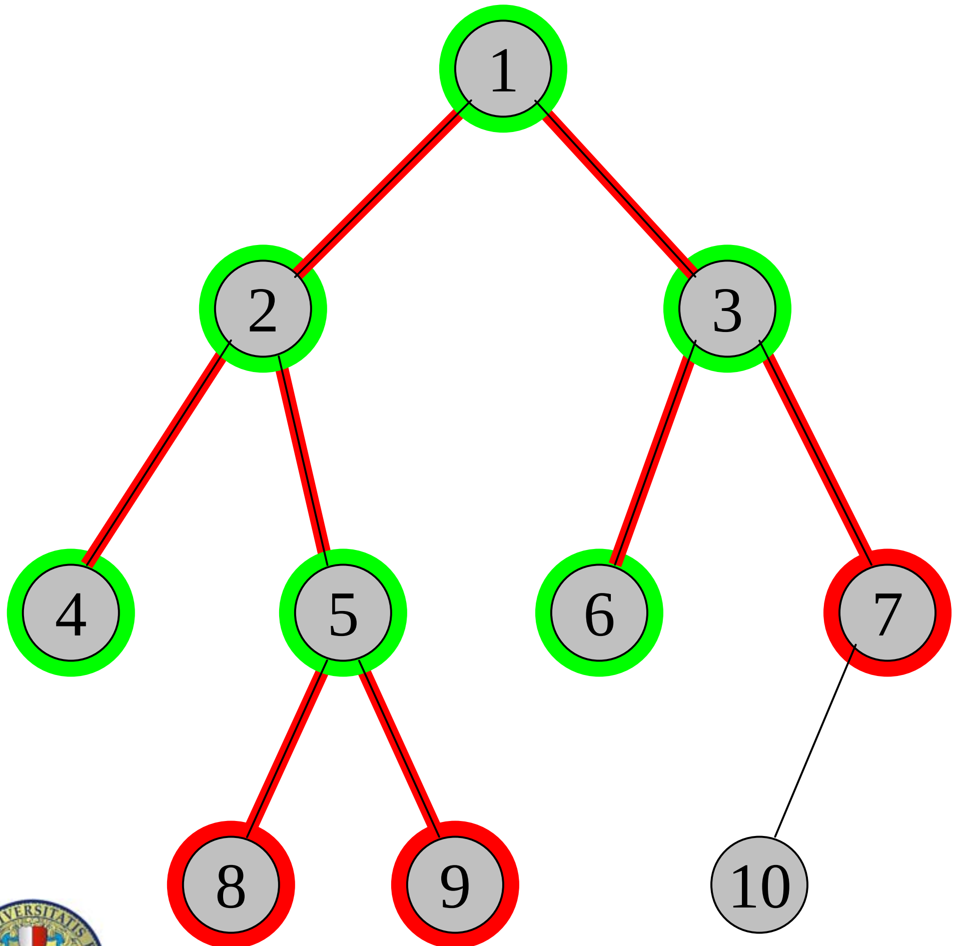
PROCEDURA DI VISITA IN AMPIEZZA



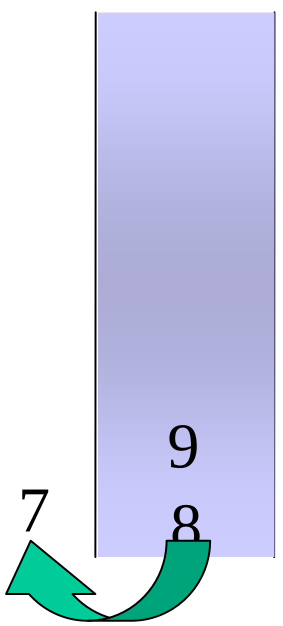


1,2,3,4,5,6

PROCEDURA DI VISITA IN AMPIEZZA

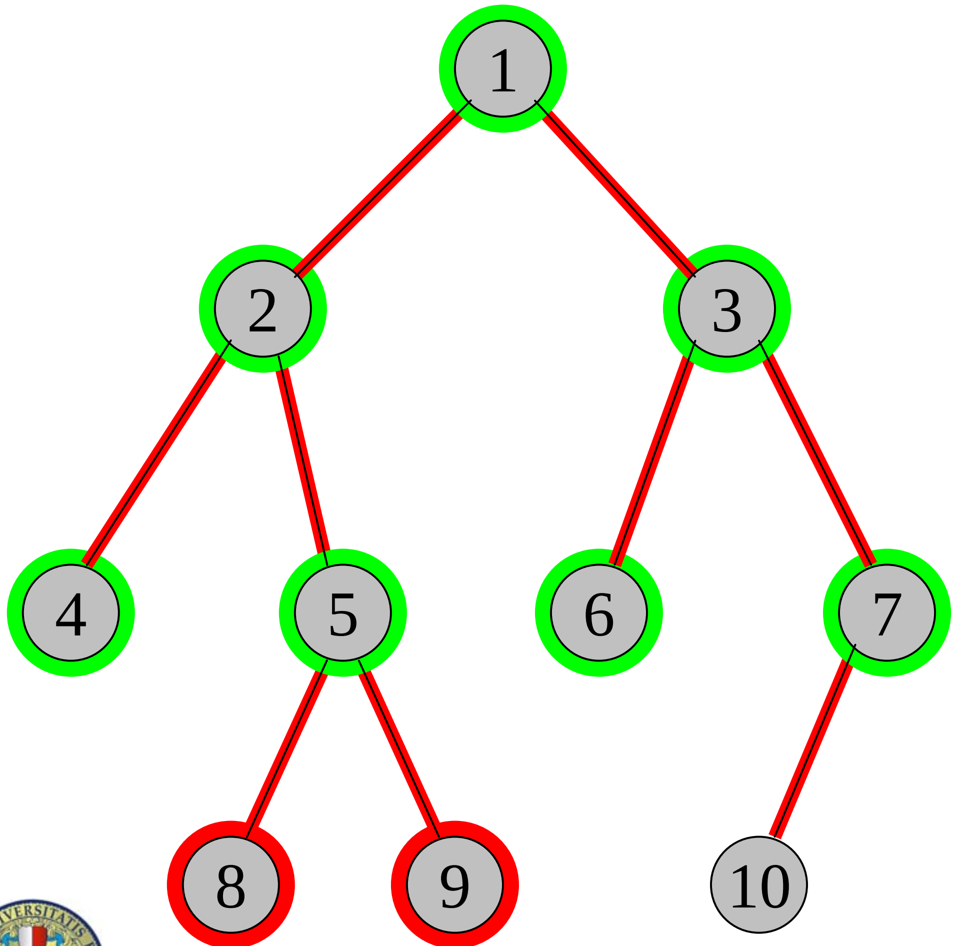


CODA

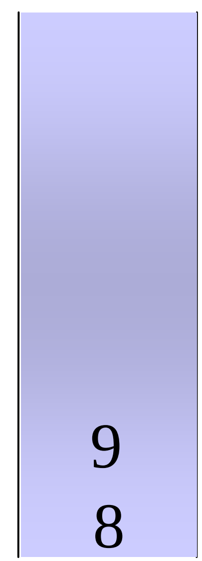


1,2,3,4,5,6

PROCEDURA DI VISITA IN AMPIEZZA

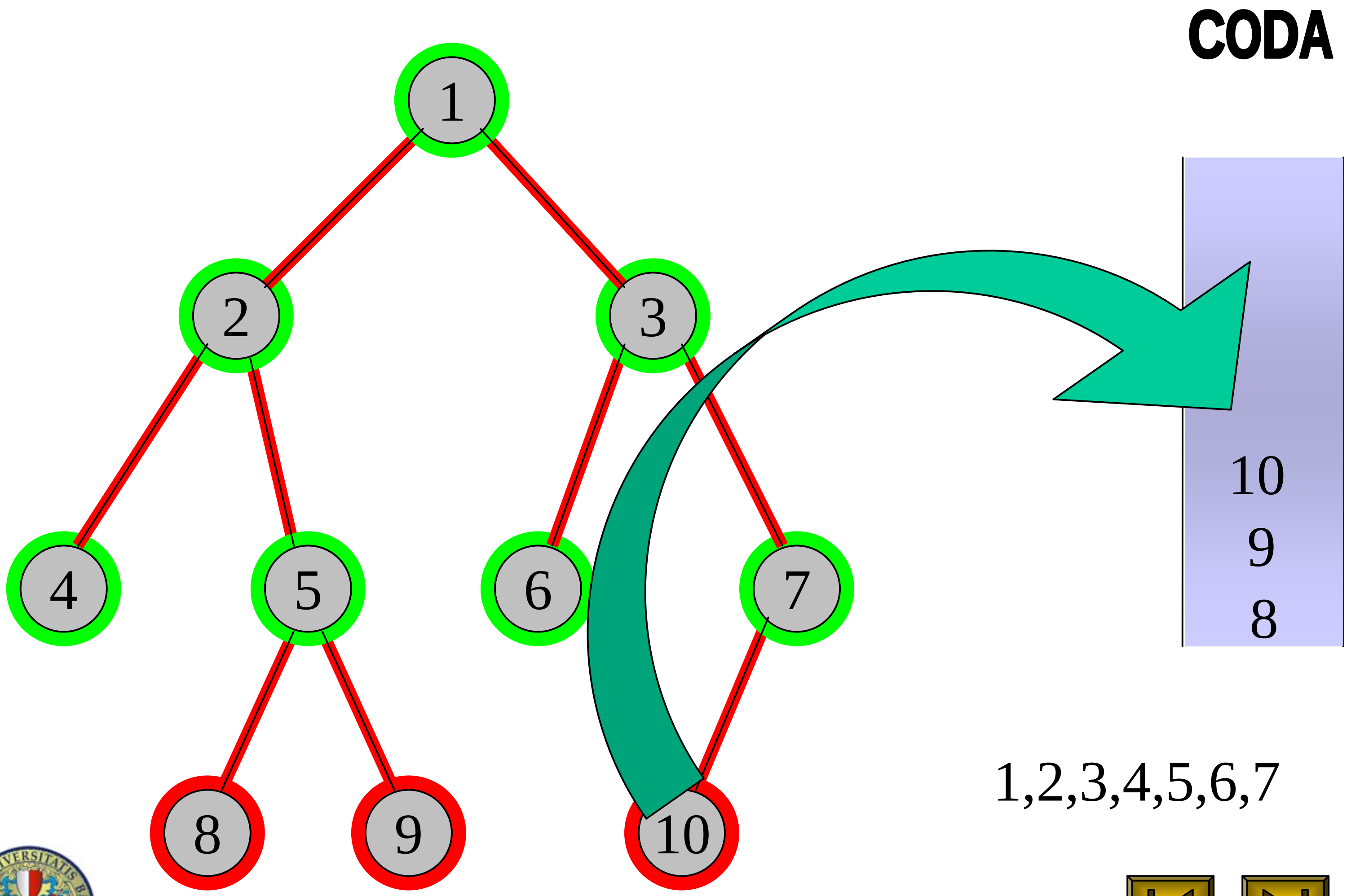


CODA



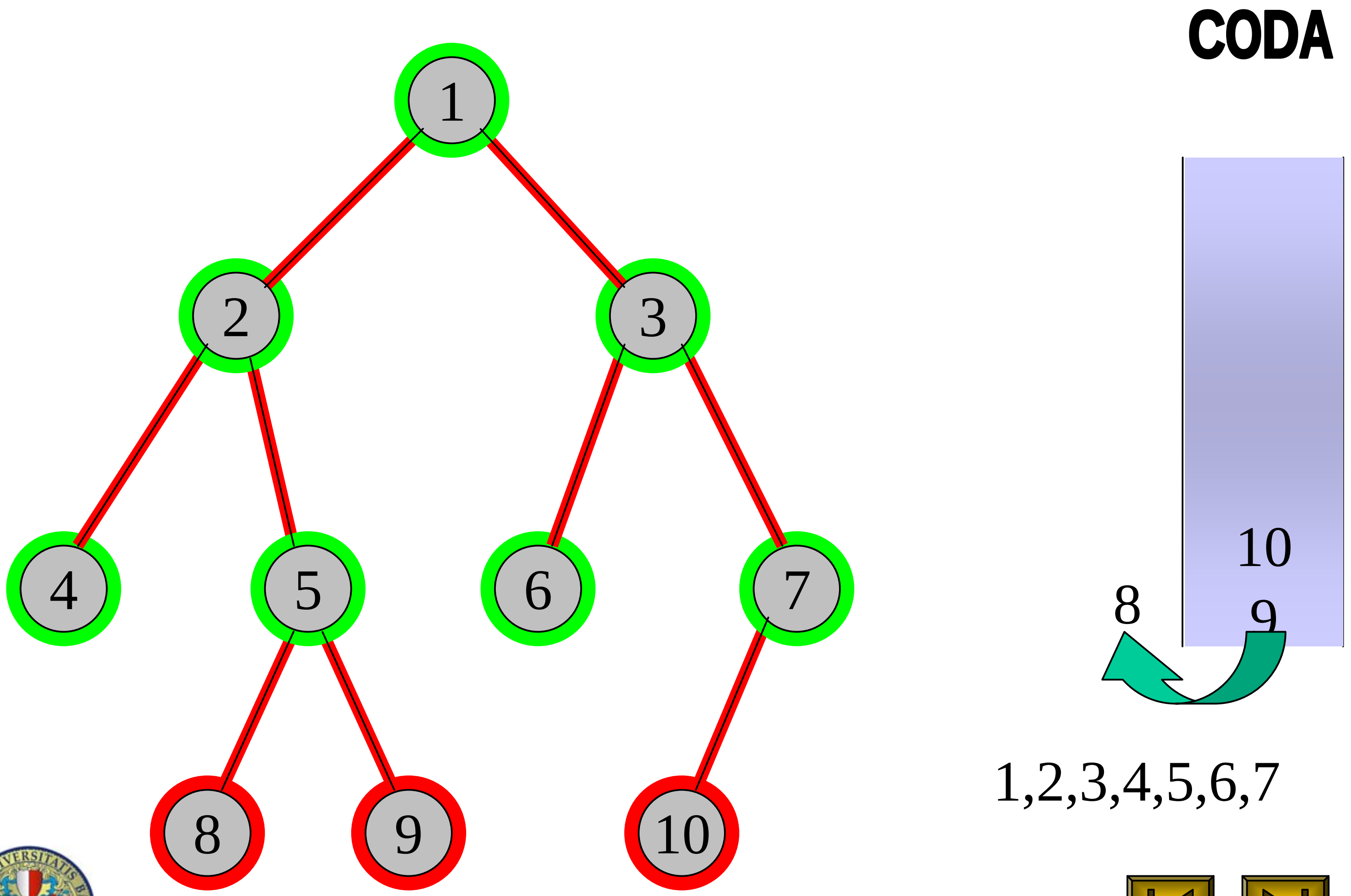
1,2,3,4,5,6,7

PROCEDURA DI VISITA IN AMPIEZZA

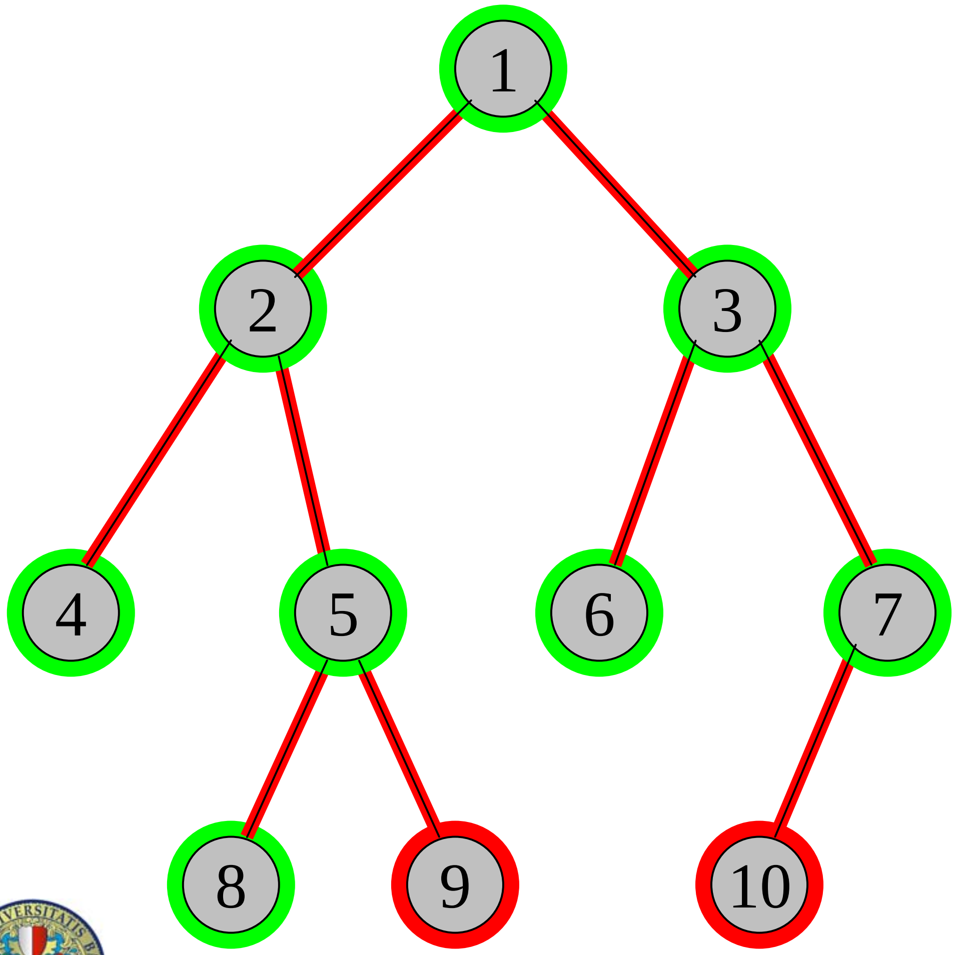


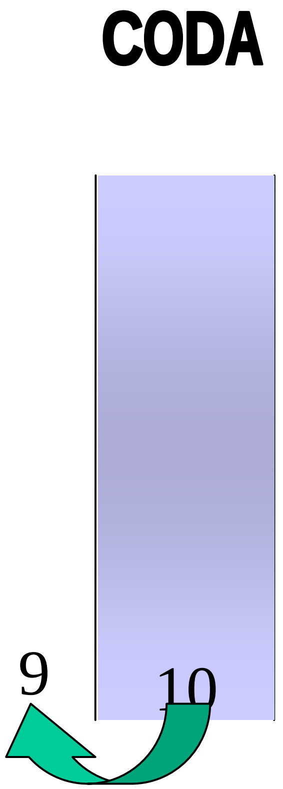
1,2,3,4,5,6,7

PROCEDURA DI VISITA IN AMPIEZZA



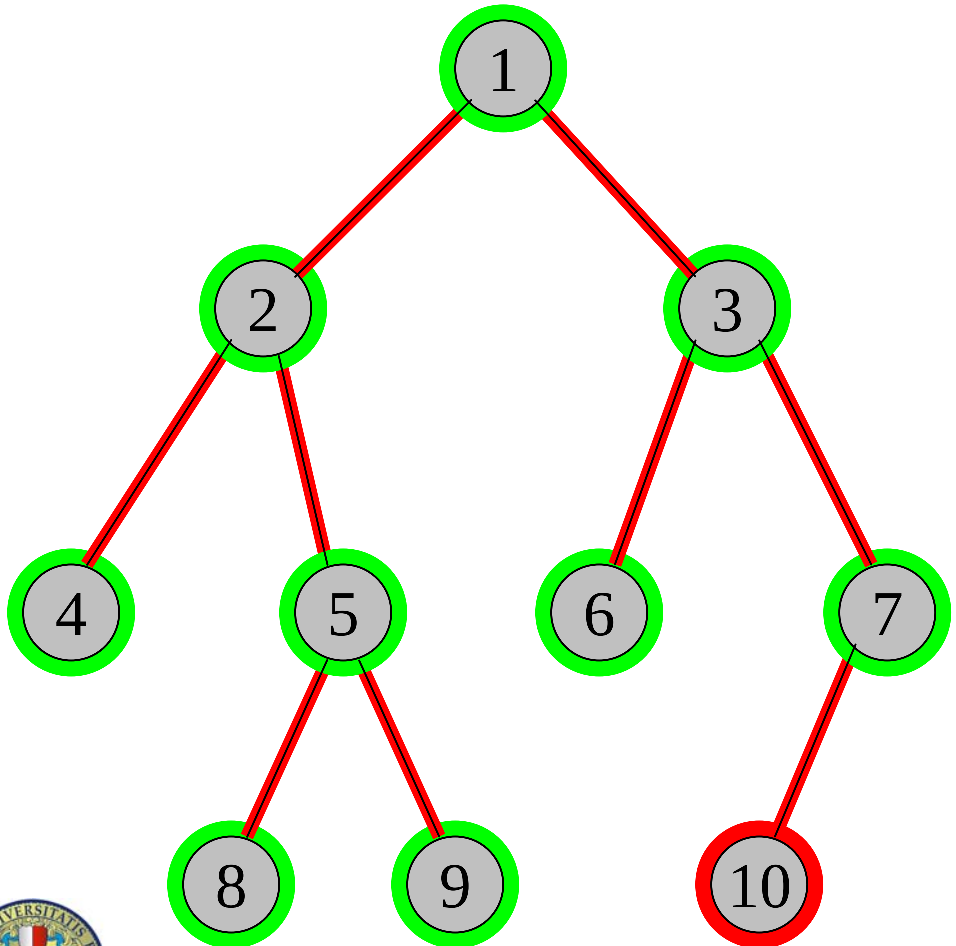
PROCEDURA DI VISITA IN AMPIEZZA

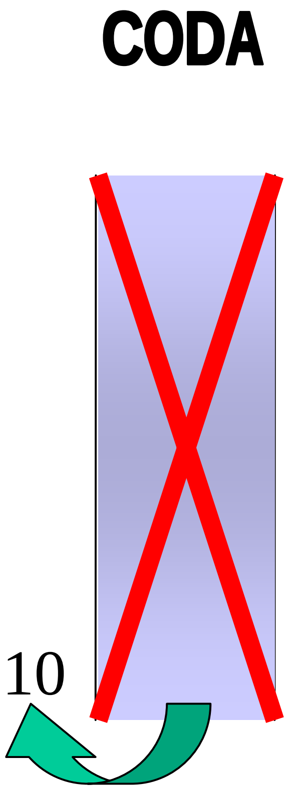




1,2,3,4,5,6,7,8

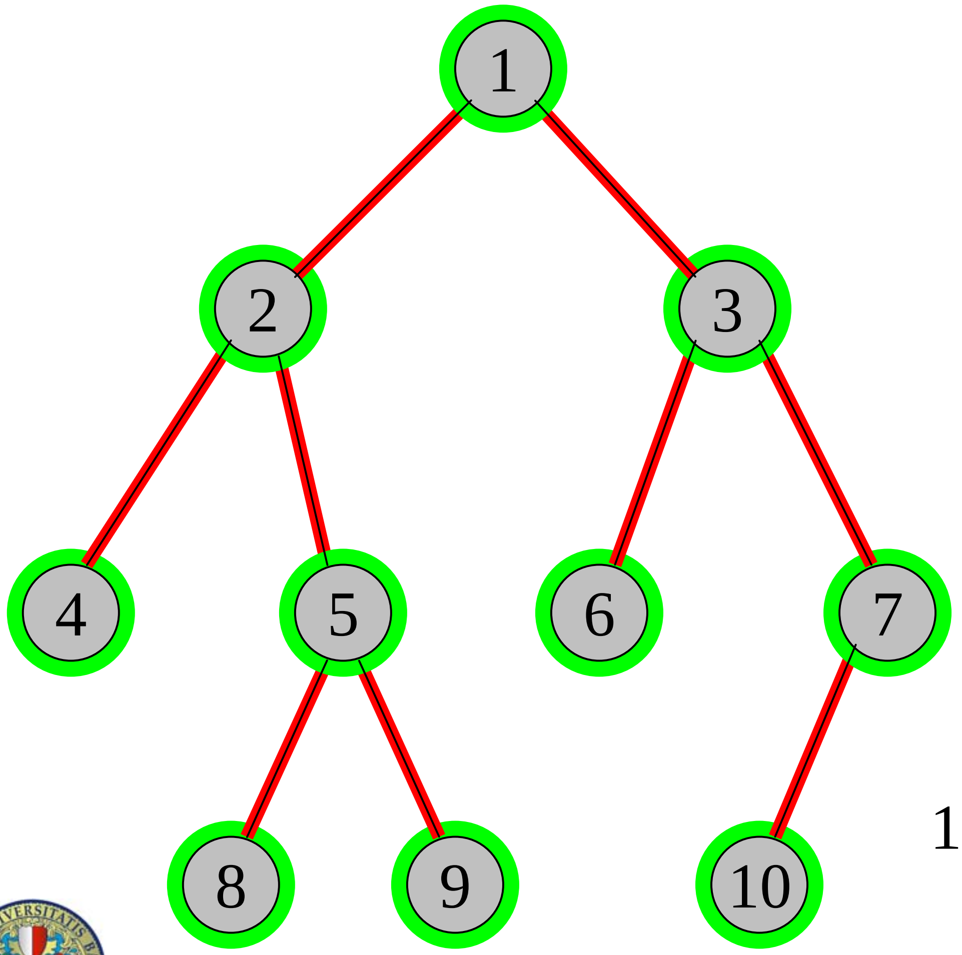
PROCEDURA DI VISITA IN AMPIEZZA



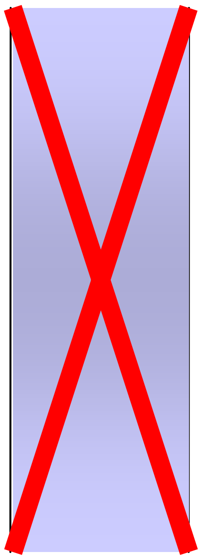


1,2,3,4,5,6,7,8,9

PROCEDURA DI VISITA IN AMPIEZZA



CODA



1,2,3,4,5,6,7,8,9,10

PSEUDOCODICE:

```
PROCEDURE visita_livello (T albero)
coda c,  tipolelem a ,  nodo u
      u ← binradice (T)
      creacoda (C)
      incoda (u,C)
      WHILE (codavuota(C) = FALSE) DO
                  u ← leggicoda (C)
                  fuoricoda (C)
                  a ← legginodo (u,T)
                   SCRIVI (a)
                  IF (sinistrovuoto (u,T) = FALSE)
                  THEN
                        incoda (figliosinistro (u,T))
                  IF (destrovuoto (u,T) = FALSE)
                  THEN
                        incoda (figliodestro (u,T))
```