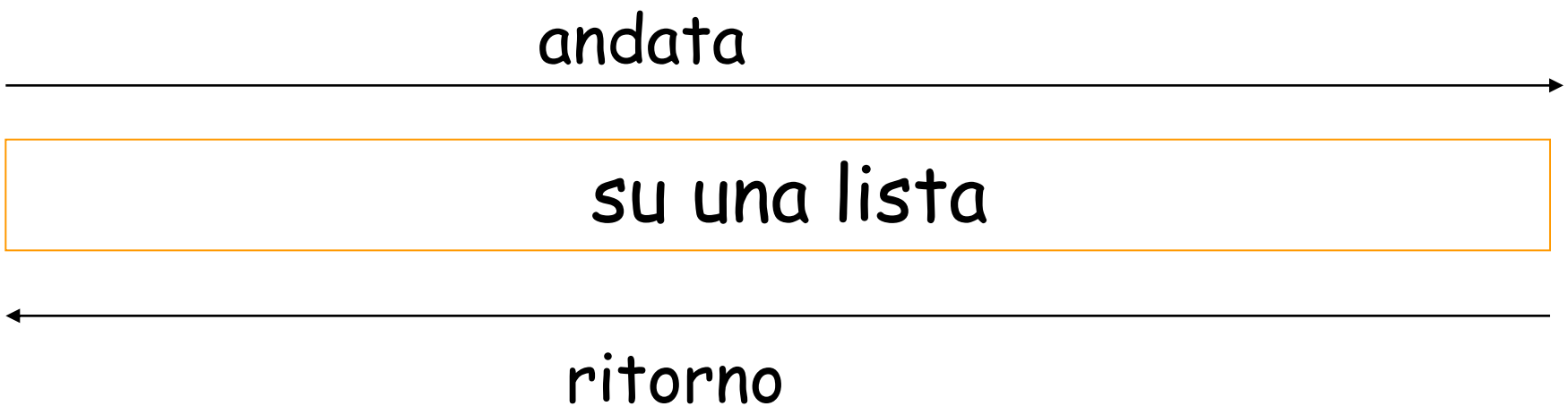
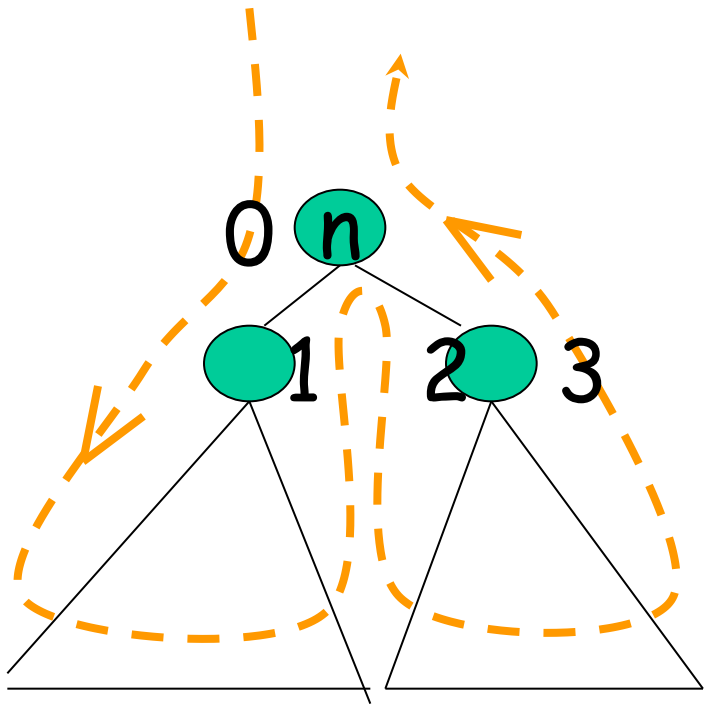
ricorsione
e
iterazione

lavorare sugli alberi è più difficile da simulare con iterazione rispetto a lavorare sulle liste



passiamo su un nodo solo 2 volte

la ricorsione è terminale se al ritorno non si fa nulla



questo schema rappresenta una ricorsione non terminale perché al ritorno dalla prima invocazione si fa la seconda

ricorsione terminale sulle liste: stampa

```
void stampa_ric(nodo * L)
{ if(L)
 {
  cout<<L→info<<' ';
  stampa(L→next);
  }
}
```

è facile

stampa iterativa di una lista:

```
void stampa(nodo *L)
{while(L)
{cout<<L→info<<' '; L=L→next; }

cout<<endl;
}
```

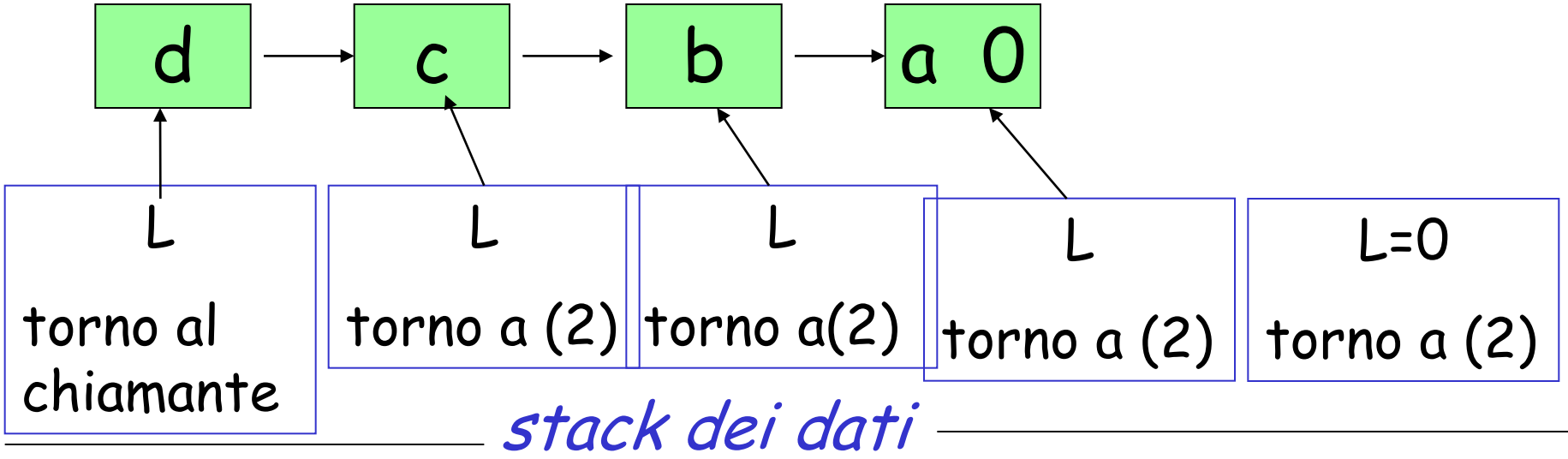
è facile simulare la ricorsione terminale,
ma se volessimo la stampa a rovescio?

la soluzione ricorsiva è facile !!

```
void stampa_ric(nodo * L)
{ if(L)
 {stampa(L→next);          ←(1)
  cout<<L→info<<' ';       ←(2)
  }
}
```

L viene visitato 2 volte

e quella iterativa ??



ogni invocazione ritorna al punto (2) della precedente, cioè alla stampa del campo info del nodo gestito da questa

l'operazione da fare al ritorno è sempre la stessa (stampa) quello che conta è cosa stampare (L->info)

il while simula l'attraversamento della lista, ma non ci da il ritorno indietro !!!

e in questo esempio ci serve,

dobbiamo essere capaci di considerare i nodi della lista 2 volte:

(1) per andare avanti

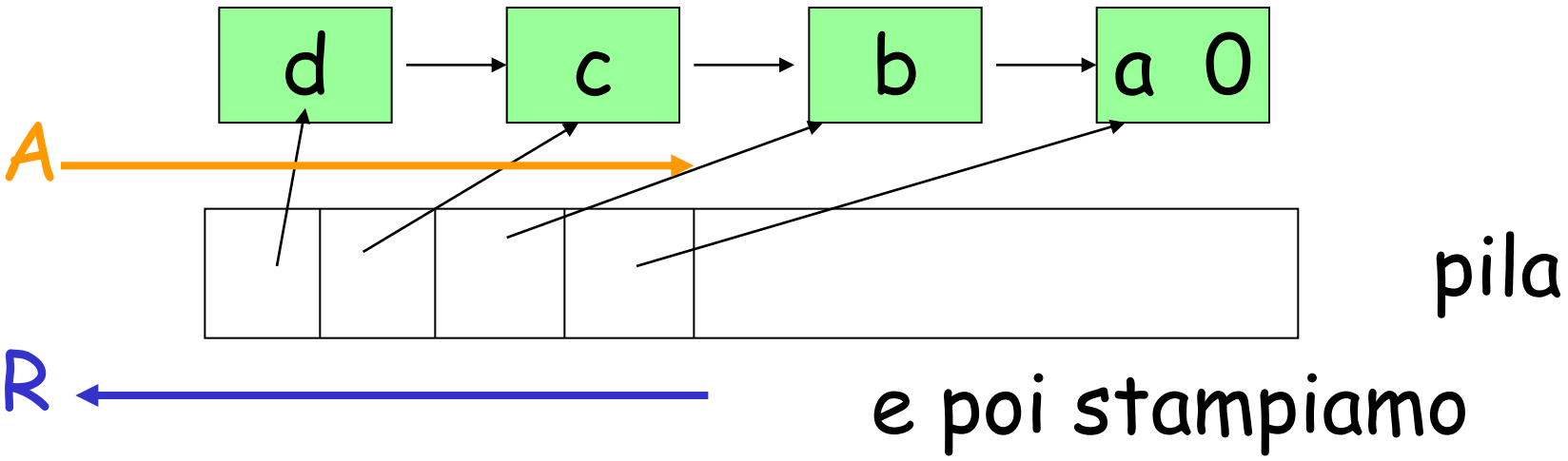
(2) per stampare il campo info

come fa la soluzione ricorsiva

facciamo così:

nel while costruiamo una pila con i puntatori ai nodi attraversati, dopo ci basterà fare pop e stampare i nodi ritrovati

fino a che la pila diventa vuota



```
void stampa_iter(nodo *L)
{ nodo * pila[100];   int top=0;
while(L)
{pila[top++] = L; L=L→next;}

top=top-1; while(top>=0)
{cout<<(pila[top])→info<< ' '; top--;}
//indietro                    cout<<endl;

}
```

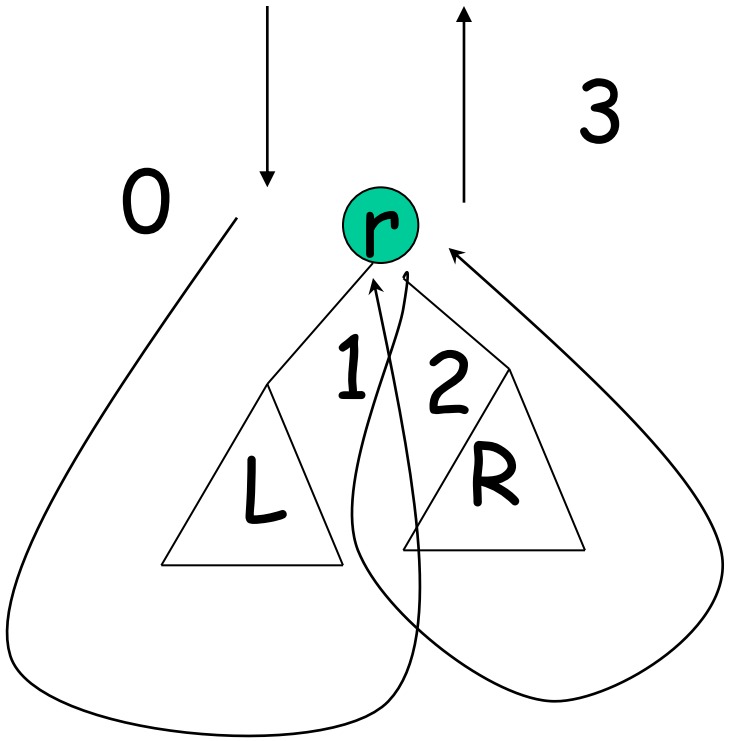
basta il puntatore al nodo senza altre informazioni perché al ritorno stampo sempre il campo info del nodo,
ma per gli alberi ?

stampa infissa di un albero:

```
void stampa(nodo *r)
{if(r)
{stampa(r→left);   ←(0)  left
cout<<r→info<<' ';   ←(1)  stampa
stampa(r→right);   ←(2)  right
}                          ←(3) return

}
```

per il nodo r, 4 operazioni diverse da fare



ogni nodo ha 4 "stati" diversi ed ogni stato richiede di fare un'operazione particolare

rappresentiamo i 4 stati:

0) [0, r]

stampa(x->left);

1) [1, r]

cout<<x->info;

2) [2, r]

stampa(x->right);

3) [3, r]

```
struct ele{int fase; nodo * N; };
```

usiamo anche il costruttore di ele

la pila diventa:

```
ele pila[100];
```

```
void stampa_iter(nodo *r)
{ele pila[100]; int top=0;
if(r) pila[top++]=ele(0,r);
while(top)
{switch(pila[top-1].fase)
case 0:
if(! pila[top-1].N)   top--;   //base
else
{pila[top-1].fase++; //prossima fase
pila[top++]=ele(0,pila[top-1].N→left);}
break;
case 1: ……???……                                  } }
```