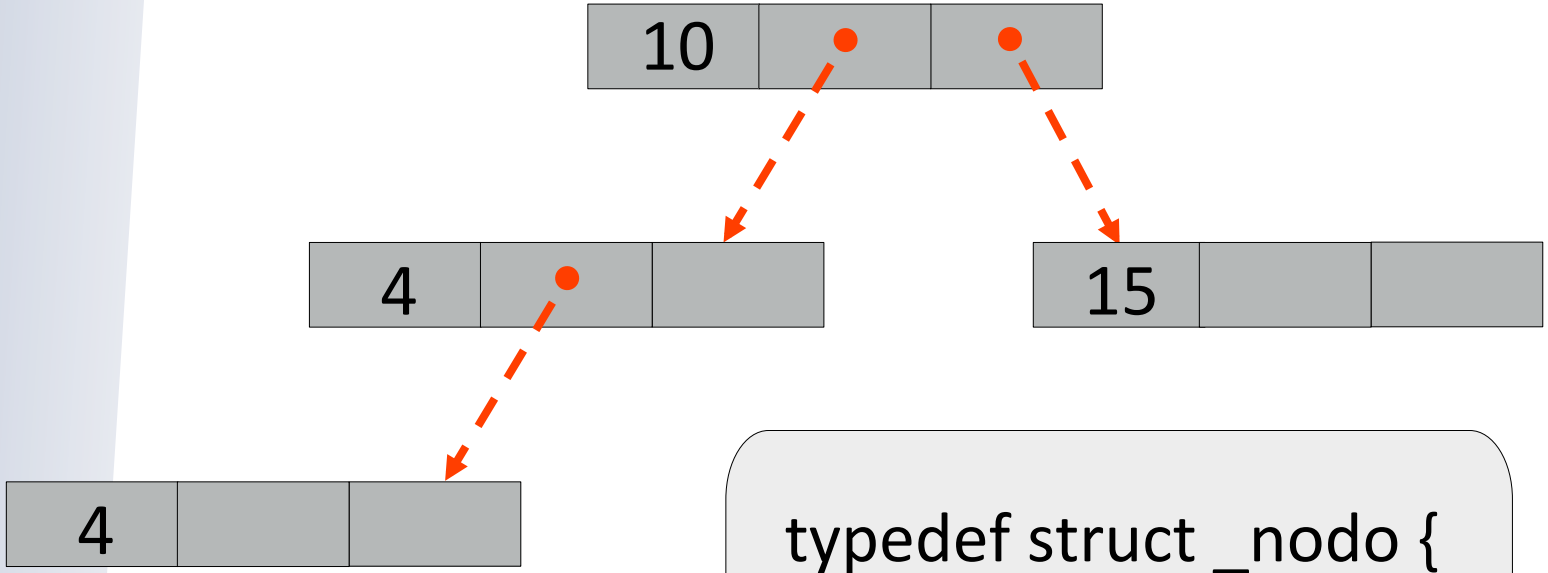
AA 2014-2015

PROGRAMMAZIONE 2

3a. Verso Java

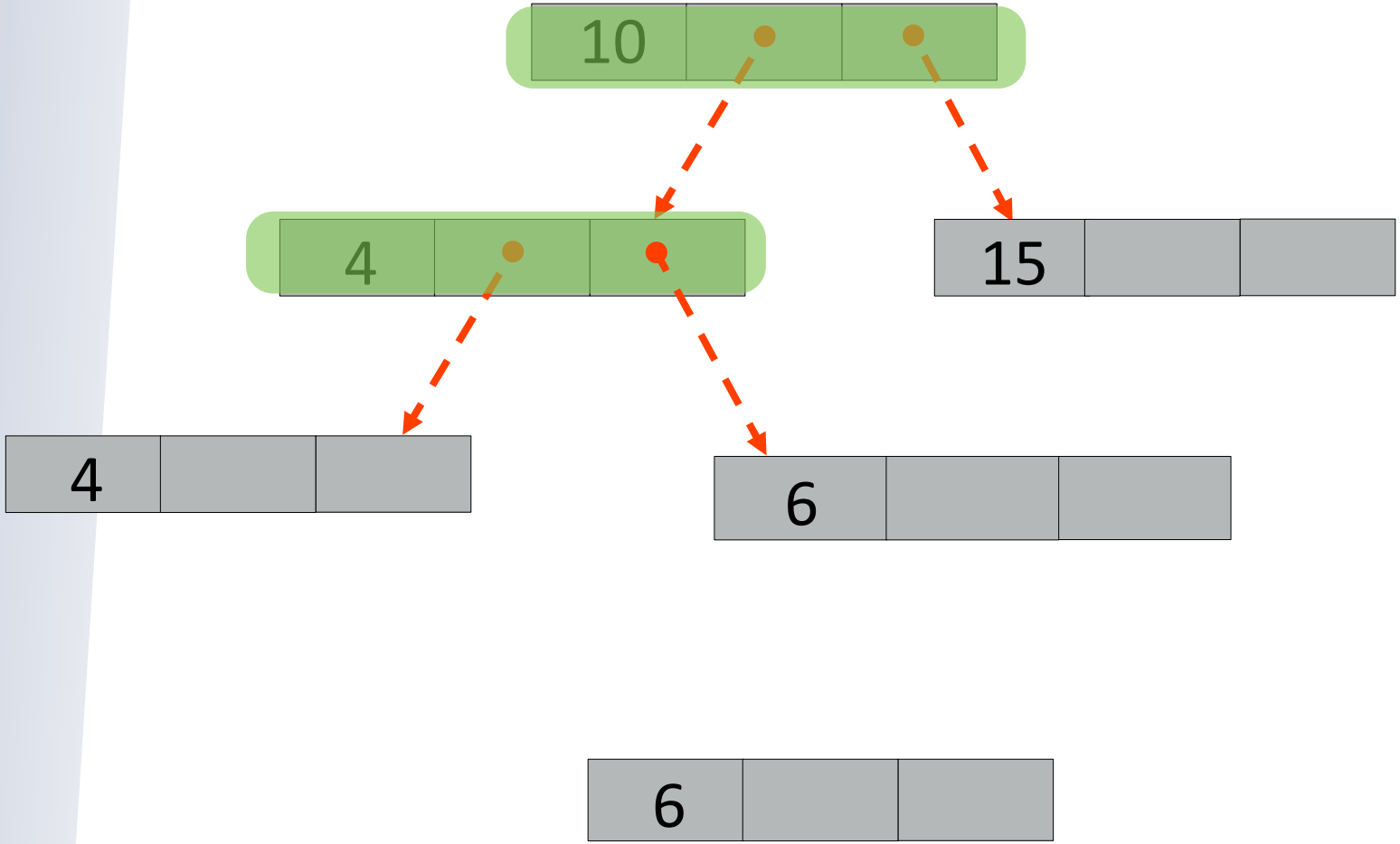
Una implementazione in C degli alberi binari di ricerca

un albero binario di ricerca

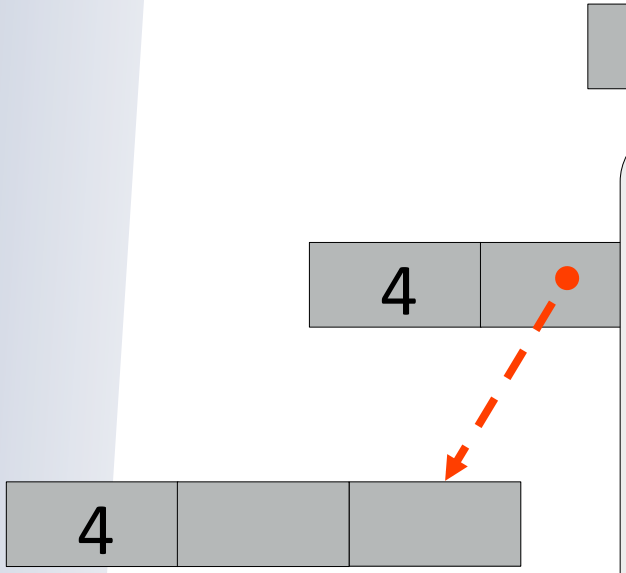



```
typedef struct _nodo {
    int key;
    struct _nodo *left;
    struct _nodo *right;
} nodo;
```

un albero binario di ricerca

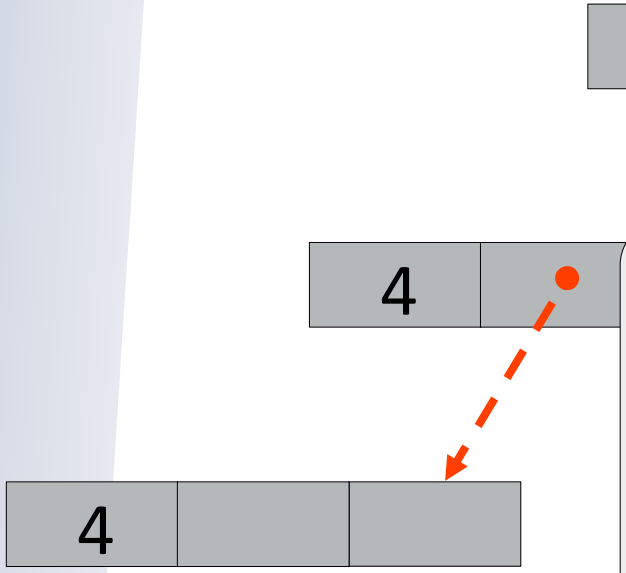


Inserimento iterativo



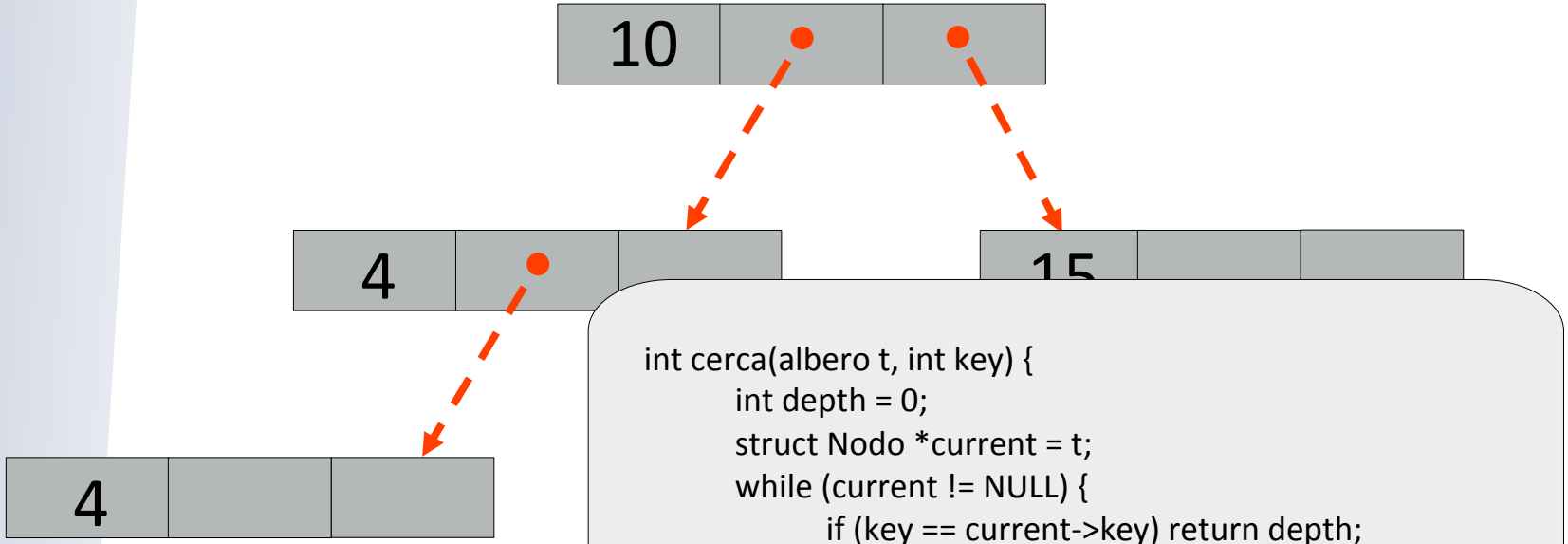

```
nodo* inserisci(nodo *t, int key) {
        nodo* new = (nodo *) malloc(sizeof(nodo));
        new->key = key;
        new->right = new->left = NULL;
        if (t == NULL) { return new; }

        nodo* parent;
        nodo* current = t;
        while (current != NULL) {
                parent = current;
                if (current->key < key) current = current->right;
                else current = current->left;
        }
        if (parent->key < key) parent->right = new;
        else parent->left = new;
        return t;
}
```

Inserimento ricorsivo



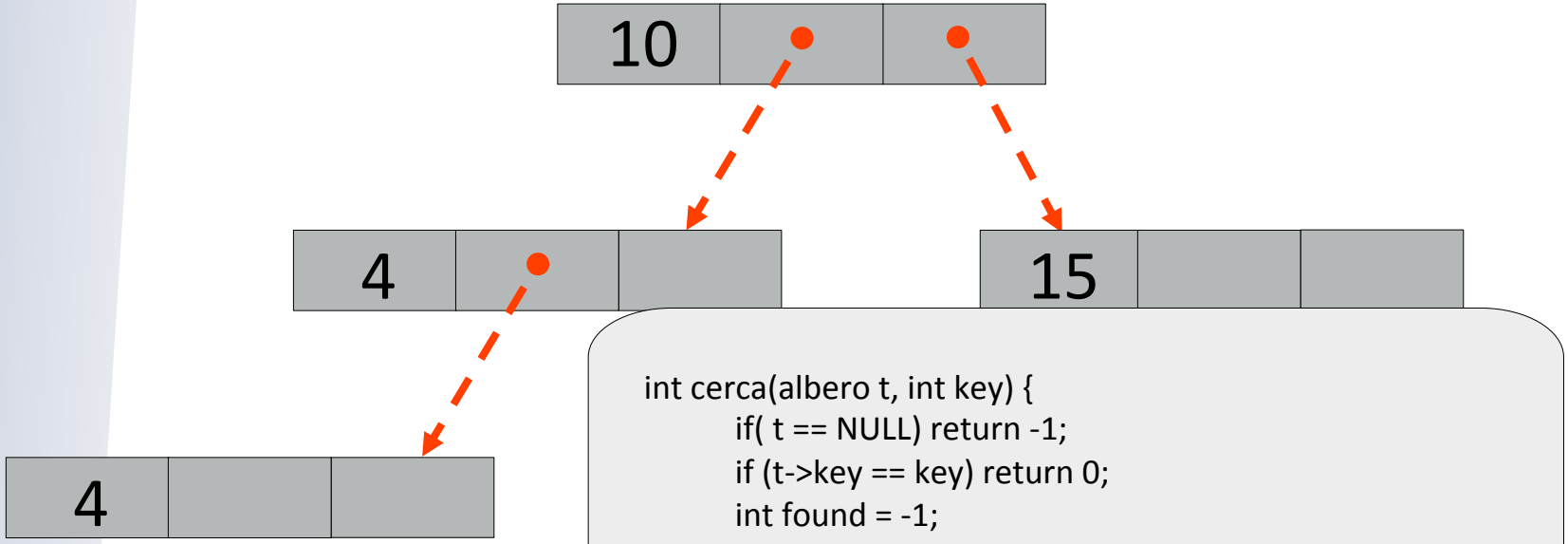

```
nodo* inserisci(nodo *t, int key) {
        if (t == NULL) {
          nodo* new = (nodo *) malloc(sizeof(nodo));
          new->key = key;
          new->left = NULL;
          new->right = NULL;
          return new;
        }

        if (t->key < key)
                        t->right = inserisci(t->right, key);
        else
                        t->left = inserisci(t->left, key);
        return t;
}
```

ricerca iterativa



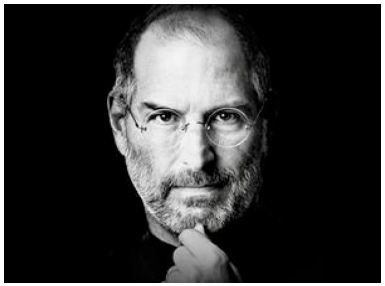

```
int cerca(albero t, int key) {
        int depth = 0;
        struct Nodo *current = t;
        while (current != NULL) {
                if (key == current->key) return depth;
                if (current->key < key)
                        current = current->right;
                else
                        current = current->left;
                depth++;
        }
 return -1;
}
```

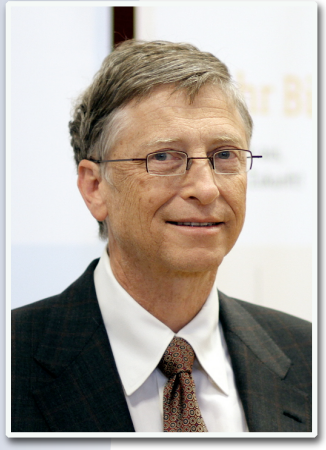
ricerca ricorsiva




```
int cerca(albero t, int key) {
        if( t == NULL) return -1;
        if (t->key == key) return 0;
        int found = -1;
        if( t->key < key)
                found = cerca(t->right,key);
        else
                found = cerca(t->left,key);

        if (found >= 0) return 1+found;
        else return -1;
}
```

scenario: ABR modulo condiviso



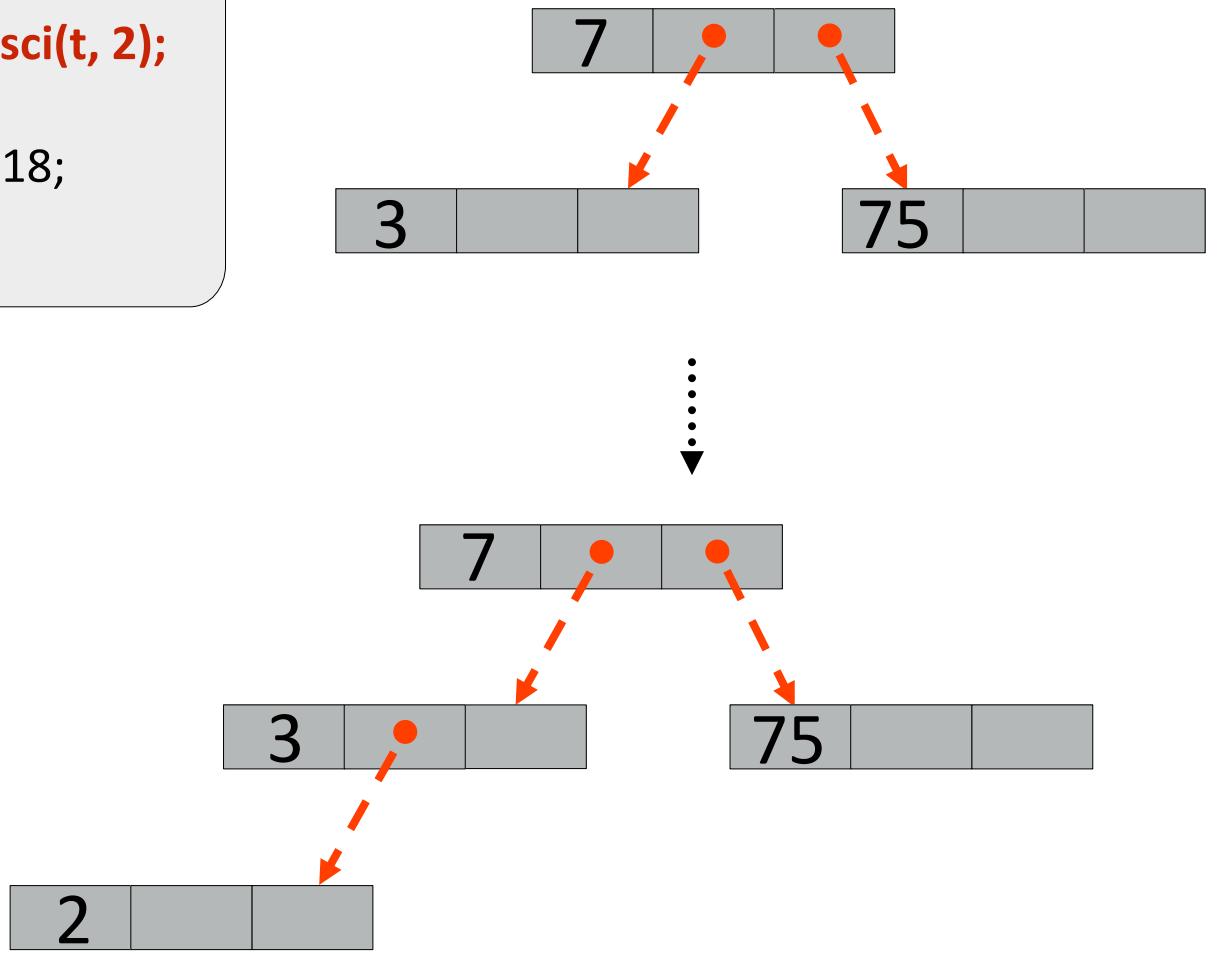


ABR

Programmatori
condividono
la struttura ABR

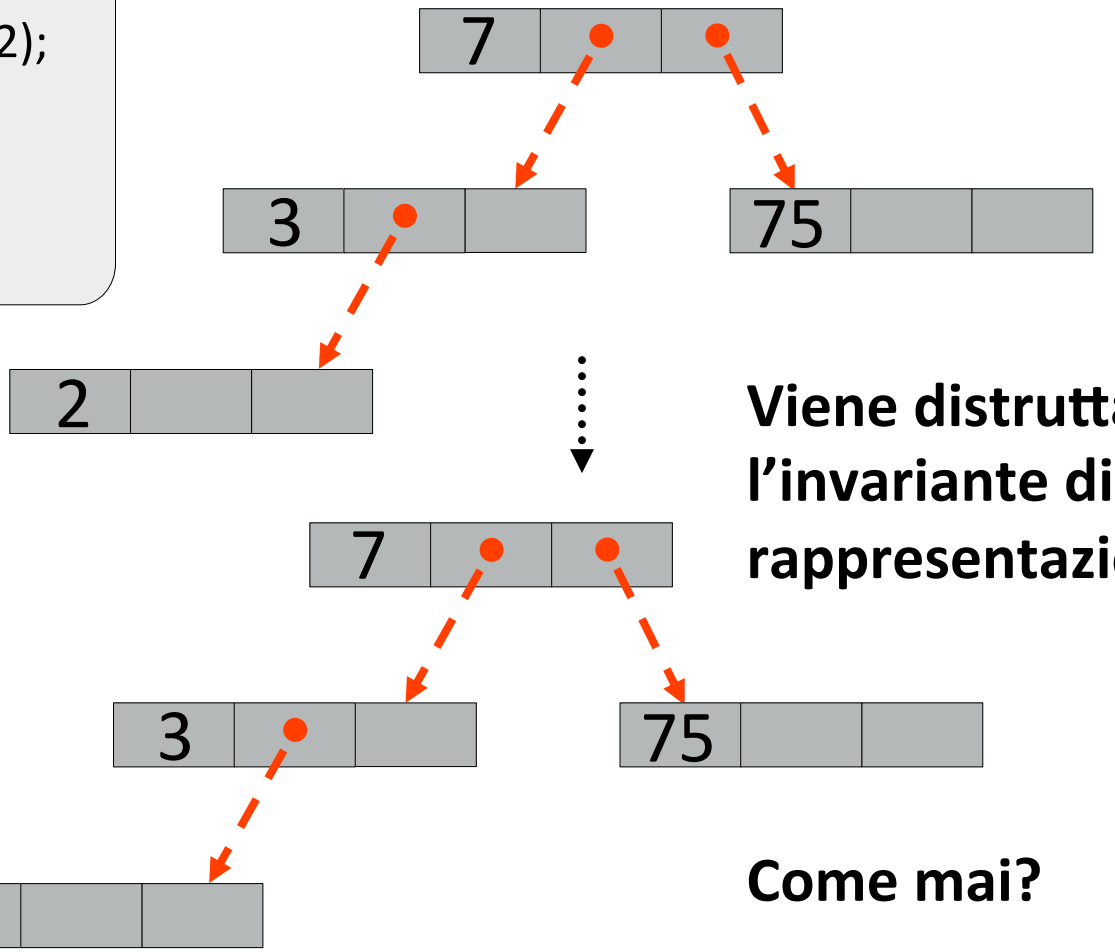
Goofy's code

```
:
G_node = inserisci(t, 2);
:
G_node >key  = 18;
:
```



Goofy's code

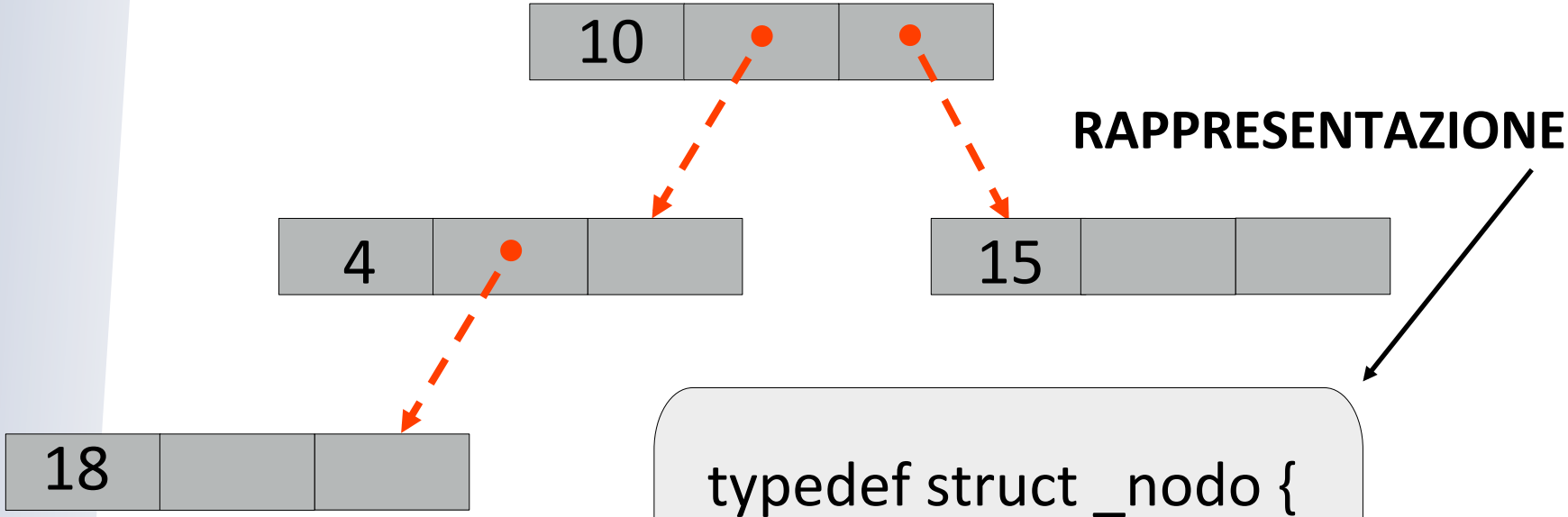
```
:
G_node = inserisci(t, 2);
:
G_node >key  = 18;
:
```



Viene distrutta l'invariante di rappresentazione

Come mai?

invarianti e rappresentazione



RAPPRESENTAZIONE

(nodo ->left)->key < nodo->key &&
nodo->key < (nodo ->right)->key

PUBBLICA: visibile a tutti

```
typedef struct _nodo {
    int key;
    struct _nodo *left;
    struct _nodo *right;
} nodo;
```

Scenario 2: ABR e dizionario

- ABR per implementare un dizionario
  - l'estensione richiede di avere una chiave per effettuare la ricerca e una stringa per codificare l'informazione
- Riuso del codice: utilizziamo il vecchio modulo aggiungendo le opportune modifiche per realizzare il dizionario

Scenario 2: ABR e dizionario


```
typedef struct _nodo {
    int key;
    string info;
    struct _nodo *left;
    struct _nodo *right;
} nodo;
```

L'invariante è ora una proprietà delle chiavi

ricerca…


```
nodo* inserisci(nodo *t, int key, string info) {
    if (t == NULL) {
      nodo* new = (nodo *) malloc(sizeof(nodo));
      new->key = key;
      new->info = info;
      new->left = NULL;
      new->right = NULL;
      return new;
    }

    if (t->key < key)
            t->right = inserisci(t->right, key, info);
    else
            t->left = inserisci(t->left, key, info);
    return t;
}
```

valutazione della soluzione

- Non abbiamo strumenti linguistici (ovvero previsti nel linguaggio) per estendere il codice alle nuove esigenze
- Cut&Paste Reuse
  - codice debole
  - difficile da mantenere
  - difficile evitare errori

sull'astrazione

- "Abstraction arises from a recognition of similarities between certain objects, situations, or processes in the real world, and the decision to concentrate upon those similarities and to ignore for the time being the differences."

  [Tony Hoare]

- Astrazione: separare le funzionalità offerte dalla loro implementazione

funzionalità vs. implementazione

encapsulation

- "Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior; encapsulation serves to separate the contractual interface of an abstraction and its implementation."

[Grady Booch]

sull'erediterietà

- Passare dal *Cut&Paste reuse* a un metodo supportato da strumenti linguistici nel quale una nuova funzionalità è ottenuta estendendo esplicitamente del codice già implementato

- La nuova implementazione estende la vecchia con funzionalità aggiuntive ma conservando quelle esistenti