Lezione 11

📚 Appunti di Davide Scarlata 2024/2025

👨‍🏫 **Prof:** *Michele Garetto*
✉️ **Mail:** 📩 michele.garetto@unito.it
📌 **Corso:** 🎓 **C**
🔗 **Moodle corso C**
🔗 **Moodle Lab matricole dispari**
🗓️ **Data:**

---

modifiche sugli alberi

aggiunta di un nodo (add)

2 casi:

- albero ordinato
  esiste un criterio di ordinamento per i nodi

esempio implementazione

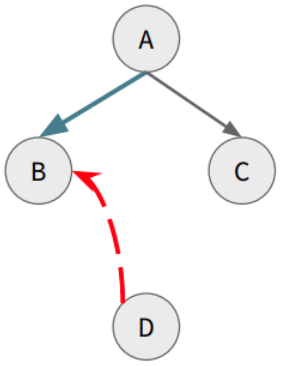
```
void add_ord(tree *T,el){
    if(T){// è l'indirizzo di un albero?
    }
    if(*T){
        if(el<(*T)->dato){
            add_ord(&((*T)->left),el);
        }
        else if(el>(*T)->dato){
            add_ord(&((*T)->right),el);
        }
        else{
            printf("nodo già esistente");
        }
    else *T = crea_nodo(el);
}
```

- albero non ordinato
  il problema è che non sappiamo dove aggiungere il nodo, quindi dobbiamo fare una ricerca per trovare la posizione giusta.

soluzione: passare alla funzione un percorso che dice come muoversi nell'albero per raggiungere il nodo genitore

"sd"

s: spostati a sinistra (da A raggiungo B)
d: spostati a destra (B non ha un figlio destro, il campo vale NULL)



ho raggiunto un puntatore NULL, è il posto in cui inserire il nodo

esempio di implementazione

```
void add(int el, tree *T, char* path) {
        if(T){// se t è un inidirizzo di un albero
        }
        if(*T){// se l'albero non è vuoto
            if(*path == '\0'){ //errore
                printf("c'è già un nodo in questa posizione %d",(*T)->dato)
            }
            else if(*path == 's'){ //spostati a sinistra
                add(el,&(*T)->left,path+1);
            }
            else if(*path == 'd'){ //spostati a destra
                add(el,&(*T)->right,path+1)
            }
            else{ //errore nel path
            }

        }
        else if(path == '\0'){//caso in cui inseriamo il nuovo nodo
            *T=crea_nodo(el);
        }
        else{//errore
            printf("il percorso non esiste")
        }
}
```

rimozione di nodi negli alberi

possiamo fare 3 tipi di rimozioni:

1. rimozione totale
   si implementa con una funzione ricorsiva in postorder
   implementazione:

```
void rimuovi(tree *T){
    if(t)
    if(*t){
        rimuovi(&((*t)->left));
        rimuovi(&((*t)->right));
        free (*T);
    }
}
```

2. restituzione all'ambiente chiamante
   la funzione restituisce l'albero senza il nodo
   implementazione:

```
tree* rimuovi(tree *T){
    if(t)
    if(*t){
        rimuovi(&((*t)->left));
        rimuovi(&((*t)->right));
        free (*T);
        return NULL;
    }
}
```

3. cancello solo il nodo e riposiziono i suoi nodi figli
   implementazione:

```
tree* rimuovi(tree *T){
    if(t)
    if(*t){
        rimuovi(&((*t)->left));
        rimuovi(&((*t)->right));
        free (*T);
        return NULL;
    }
    else{
```

```
        tree *tmp = *T;
        if((*T)->left == NULL){
            *T = (*T)->right;
            free(tmp);
            return *T;
        }
        else if((*T)->right == NULL){
            *T = (*T)->left;
            free(tmp);
            return *T;
        }
        else{
            tree *tmp2 = (*T)->left;
            while(tmp2->right != NULL){
                tmp2 = tmp2->right;
            }
            tmp2->right = (*T)->right;
            free(*T);
            return tmp2;
        }
    }
}
```

nodo generico

contiene:

- un campo per il dato di tipo void * (puntatore generico)
- uno o più riferimenti a nodi collegati (successore, figli, …)