# Laboratorio di Algoritmi e Strutture Dati

## Secondo esercizio, seconda parte: alberi red-black (punti: 1 o 2)

# Esperimenti con strutture dati

In questo punto, aggiungiamo una nuova struttura dati a quella costruita al punto precedente, implementando le stesse operazioni nello stesso modo, al fine di operare un confronto.

# Alberi red-black: una struttura ricorsiva

```
struct tree_node{
    int key
    int color
    *struct tree_node parent
    *struct tree_node left_child
    *struct tree_node right_child
}
```

```
struct tree{
    int cardinality
    *struct tree_node root
}
```

# Alberi red-black: gestione dell'esperimento singolo

Procediamo dunque come nel caso del precedente esperimento.

Al livello più basso, il singolo passo consiste nel generare un numero fissato di chiavi casuali, scegliere un'operazione tra inserimento e cancellazione, ed eseguirla.

Usiamo la stessa strategia dell'esperimento semi-randomico nel caso degli alberi binari di ricerca.

Attenzione: non abbiamo visto la cancellazione in questi alberi. Se questa viene fatta in maniera autonoma, allora l'esercizio verrá valutato 2 punti. Altrimenti il confronto sará fatto sull'inserimento unicamente, e verrá valutato un punto.

La cancellazione la guardiamo brevemente in fondo alla lezione.

# Alberi red-black: gestione dell'esperimento

```
proc SingleExperimentRBT (max_keys, max_search, max_delete, max_instances)
  t_tot = 0
  for (instance = 1 to max_instances)
    Initialize(T)
    t_start = clock()
    for (key = 1 to max_keys)
      {RBTTreeInsert(T, Random())
    for (search = 1 to max_search)
      {RBTTreeSearch(T, Random())
    for (delete = 1 to max_delete)
      {RBTTreeDelete(T, Random())
    t_end = clock()
    t_elapsed = t_end − t_start
    t_tot = t_tot + t_elapsed
    Empty(T)
  return t_tot/max_keys
```

# Alberi red-black: esperimento

```
proc Experiment (min_keys, max_keys)
    step = 10
    max_instances = 5
    percentage_search = 60
    for (keys = min_keys to max_keys step step)
        srand(SEED)
        max_search = keys * percentage_search/100
        max_delete = keys − max_search
        timeBST = SingleExperimentBST(keys, max_search, max_delete, max_instances)
        srand(SEED)
        timeRBT = SingleExperimentRBT(keys, max_search, max_delete, max_instances)
        print(timeBST, timeRBT)
        SEED = SEED + 1
```

# Alberi rbt: esperimento

Vogliamo quindi realizzare un esperimento che consiste nel misurare il tempo necessario ad effettuare un certo numero, crescente, di operazioni standard di inserimento (e cancellazione) in un albero binario di ricerca e in un albero red-black inizialmente vuoti.

Il risultato richesto prevede una rappresentazione grafica delle curve di tempo e una dimostrazione sperimentale di correttezza attraverso test randomizzati e funzioni antagoniste.

# Conclusione

Sperimentare di prima mano con le strutture dati ci dà l'occasione di capire davvero come queste sono fatte. Questa è condizione necessaria per un uso corretto delle strutture già implementate in librerie comuni.