Progetto “albero di intervalli”

Appello del 22 luglio 2021

Descrizione del progetto

Lo scopo del progetto è quello di realizzare un *interval tree*, o albero di intervalli, che permette di memorizzare in modo efficiente dati che rappresentano intervalli nella forma $[a, b)$ per $a, b \in \mathbb{R}$ e $a \leq b$ (i.e., l'insieme $\{x \in \mathbb{R} \colon a \leq x < b\}$). L'implementazione consigliata consiste di una variazione dell'usuale albero binario di ricerca con l'aggiunta del supporto di due funzionalità:

1. Dato un punto (i.e., un valore $x \in \mathbb{R}$), restituire tutti gli intervalli in cui quel punto è contenuto (*appartenenza a un intervallo*).
2. Dato un intervallo $[c, d)$, restituire tutti gli intervalli che hanno intersezione non vuota con l'intervallo dato (*intersezione con un intervallo*).

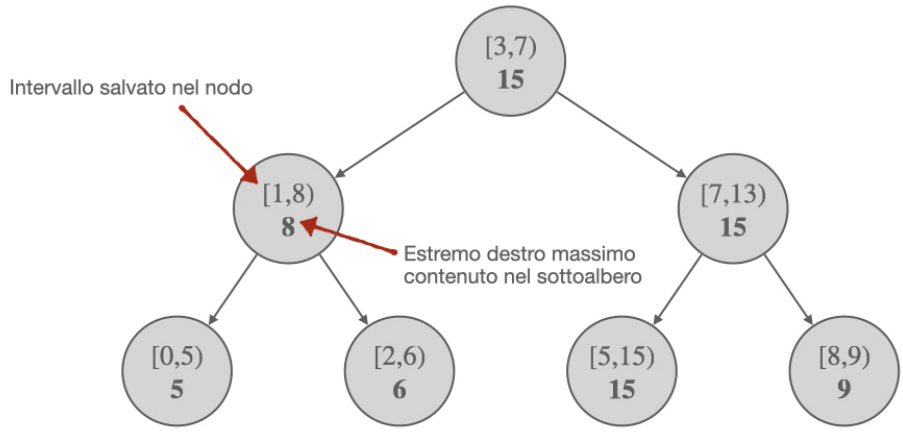
Per il progetto, un *interval tree* è un albero binario che salva in ogni nodo due informazioni:

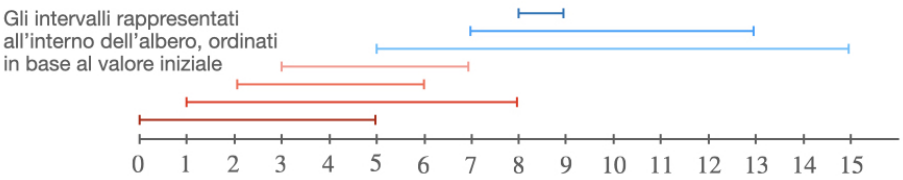
1. L'intervallo $[a, b)$ contenuto nel nodo.
2. L'estremo destro *massimo* contenuto in tutto il sottoalbero avente radice quel nodo.

I nodi nell'albero sono ordinati in base all'estremo sinistro dell'intervallo (il valore minore) ma, per consentire di svolgere in modo efficiente le operazioni di appartenenza e di intersezione viene mantenuto anche il massimo estremo destro contenuto nel sottoalbero.

Possiamo sfruttare le informazioni contenute in un nodo nel seguente modo: si supponga il nodo contenga l'intervallo $[a, b)$ e il valore $b_{\max}$; se si ricerca gli intervalli che intersecano $[c, d)$ allora: - se $d \leq a$ allora possiamo ignorare l'intero sottoalbero *destro* del nodo corrente (i valori iniziali degli intervalli saranno tutti maggiori). - se $c \geq b_{\max}$ allora possiamo ignorare il nodo corrente e tutti i suoi sottoalberi, dato che gli intervalli termineranno tutti prima dell'inizio dell'intervallo $[c, d)$. Un simile ragionamento permette di migliorare anche la ricerca di tutti gli intervalli che contengono un valore specifico.

Un esempio di interval tree è dato dalla seguente figura:





Si richiede che questa struttura dati supporti le seguenti operazioni:

- Creazione di un oggetto di tipo `IntervalTree` inizialmente vuoto.
- Inserimento di un intervallo $[a, b)$ all'interno dell'albero, questo potrebbe comportare l'aggiornamento dei valori degli estremi *massimi* contenuti nei nodi intermedi.
- Dato un valore x, ritornare tutti e soli gli intervalli $[a, b)$ contenuti nell'albero tali per cui $x \in [a, b)$.
- Dato un intervallo $[c, d)$, ritornare tutti gli intervalli $[a, b)$ contenuti nell'albero tali per cui $[a, b) \cap [c, d) \neq \varnothing$.
- Stampa in ordine (per estremo iniziale) di tutti gli intervalli contenuti nell'albero.

Si impongono inoltre i seguenti vincoli:

- La struttura usata deve essere basata su quella descritta (i.e., albero binario)
- L'informazione aggiuntiva contenuta nei nodi deve essere sfruttata nel processo di ricerca

Codice

Viene fornito uno scheletro del codice che deve essere implementato, con i metodi che devono essere scritti:

```python
class IntervalTree:

    def __init__(self):
        # Implementazione del costruttore.
        # L'albero è inizialmente vuoto

    def inserisci(self, a, b):
        # inserisce l'intervallo [a,b) nell'albero

    def intersecanti(self, c, d):
        # Ritorna una lista Python di coppie di numeri (tuple)
        # rappresentanti tutti gli intervalli che sono
        # intersecati da [c, d)

    def contenenti(self, x):
        # Ritorna una lista Python di coppie di numeri (tuple)
        # rappresentanti tutti gli intervalli che contengono
        # il valore x

    def stampa_in_ordine(self):
        # stampa tutti gli intervalli contenuti nell'albero
        # in ordine per valore dell'estremo iniziale
```

Nel progetto è consentito avere funzioni aggiuntive che testano il buon funzionamento delle funzionalità richieste. È anche consentito avere metodi aggiuntivi.

Si ricorda di commentare adeguatamente il codice, approfittandone per spiegare le scelte implementative effettuate.

Esempi d'uso

Il seguente frammento di codice chiama tutti i metodi la cui implementazione è richiesta dal progetto. Come commento è indicato il valore atteso in una implementazione funzionante:

```python
def test():
    intervalli = [(3, 7), (1, 8), (7, 13), (0, 5),
                  (2, 6), (5, 15), (8, 9)]
    t = IntervalTree()
    for a, b in intervalli:
        t.inserisci(a, b)
    t.stampa_in_ordine()
    # stampa [0, 5) [1, 8) [2, 6) [3, 7) [5, 15) [7, 13) [8, 9)
    print(t.contenenti(5))
    # stampa [(3, 7), (1, 8), (2, 6), (5, 15)]
    print(t.contenenti(2))
    # stampa [(1, 8), (0, 5), (2, 6)]
```

```
print(t.contenenti(15))
# stampa []
print(t.intersecanti(1, 9))
# stampa [(3, 7), (1, 8), (0, 5), (2, 6), (7, 13), (5, 15), (8, 9)]
print(t.intersecanti(5, 7))
# stampa [(3, 7), (1, 8), (2, 6), (5, 15)]
print(t.intersecanti(16, 18))
# stampa []
```

Indicazioni

Il progetto deve essere svolto **individualmente**. La consegna dovrà avvenire entro le ore 23:59 del giorno 16/07/2021 secondo le seguenti modalità:

- Invio di una email a `lmanzoni@units.it` dal vostro account email istituzionale con oggetto *[informatica] consegna progetto appello del 22/07/2021*.
- L'email deve avere come allegato il progetto in un singolo file in codice sorgente Python versione 3, dal nome `Nome_Cognome_matricola.py`, quindi, per esempio Mario Rossi di matricola 12345 consegnerà un file dal nome `Mario_Rossi_12345.py`.
- Il file deve contenere sotto forma di commento le seguenti linee indicanti nome, cognome e numero di matricola: `python # Nome: Mario # Cognome: Rossi # Matricola: 12345`