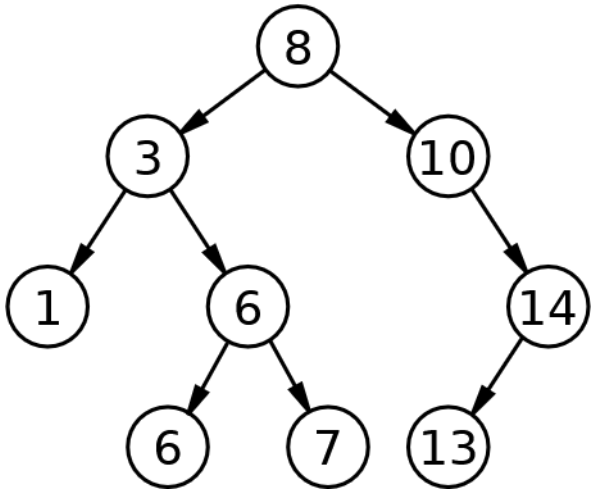
Alberi binari di ricerca: Bag

14 giugno 2023



L'esame di oggi è sugli **alberi binari di ricerca**. Li usiamo per rappresentare la struttura dati **bag** (borsa), una variante degli insiemi che ha la possibilità di avere **duplicati**. In particolare, nella struttura dati in figura ci sono due copie del numero 6. La definizione della struttura dati cambia rispetto a quanto avete visto a lezione ammettendo che il figlio **sinistro** di un nodo sia minore **o uguale** al padre, mentre, come abbiamo già visto, il figlio destro di ogni nodo deve sempre essere maggiore del figlio sinistro.

Seguendo l'organizzazione dei testi di laboratorio visti finora, avrete tre file sorgente: un main, un file header e un file **bag.cpp** che sarà l'unico che dovrete modificare. Per aiutarvi, vi forniamo anche alcuni test che funzionano come avete visto finora nei testi di laboratorio e un semplice file di input, **dati.txt**.

Naturalmente testeremo il vostro codice con file di input diversi e dati diversi.

La nostra bag contiene elementi di tipo *int*, e il nodo dell'albero è definito come segue:

```
struct bag::Node {
    Elem value;
    Node *left;
    Node *right;
};
```

Il tipo *Bag* è definito in bag.h, sempre seguendo quel che avete visto in laboratorio, come

```
typedef Node* Bag
```

Nel file **bag.cpp**, avete tre funzioni già implementate; per il codice necessario a voi sarà utile solo *isEmpty*, che testa se una bag è vuota (o, come abbiamo visto nei nostri esercizi con gli alberi, se lo è un sotto-albero).

Vi chiediamo di implementare tre funzioni:

1. *void bag::printInReverseOrder(const Bag b)*, che stampa gli elementi di *b* in ordine decrescente, su un'unica linea, separati da spazi (non è necessario andare a capo a fine linea);
2. *bool bag::isDuplicate(const Bag b, Elem v)*, che restituisce true se e solo se *b* contiene due o più copie di *v* (**indizio:** se un nodo ha duplicati nel suo sottoalbero, uno di essi sarà il suo figlio sinistro);
3. *Elem bag::sumLarger(const Bag b, Elem v)*, che restituisce la somma degli elementi di *b* più grandi di *v;*

Le funzioni da completare sono già presenti in **bag.cpp**. Le soluzioni valutate a punteggio pieno saranno quelle corrette, efficienti (visitano solo i nodi necessari) commentate laddove sia utile e ben scritte (ad esempio, non c'è codice ripetuto o ridondante).

Consegnate su AulaWeb SOLO il file bag.cpp la cui prima linea sarà un commento contenente nome, cognome e matricola.

Non fatevi prendere dal panico!

- Questa struttura dati è simile (anche se più semplice) al dizionario che avete visto nel laboratorio 6. In questo caso l'elemento del tipo di dato non è una coppia chiave-valore ma semplicemente un intero.
- Preoccupatevi prima di avere una soluzione corretta e poi, se avete tempo, di renderla più efficiente evitando (dove ha senso) di visitare tutti i nodi dell'albero.
- È possibile risolvere l'esercizio scrivendo delle semplici funzioni ricorsive. La nostra soluzione richiede un totale di 21 linee di codice per le tre funzioni.
- Commentate solo le parti non ovvie per chi conosce il C++.

Test

Di seguito alcuni test che potete eseguire a mano; sono un sottoinsieme di quelli automatici già inclusi, nel caso non vi troviate a vostro agio con essi.

Dopo aver letto il file dati.txt:

- La stampa in ordine inverso (comando "a") dovrebbe restituire "67 42 6 5 3 2 2 1 -9 -9".
- La verifica se elementi sono duplicati deve funzionare correttamente, ad esempio restituendo "vero" per 2 e "falso" per 4 e 42.
- La somma di tutti gli elementi (ad esempio, quelli maggiori di -20) è 110.