Dokumentacija

Osnove

Čvor je pomoćna generička struktura koju koristimo u implementaciji klase Stablo. Prvo pojasnimo implementaciju Čvora. Čvor ima 7 atributa. Posjeduje pokazivač na roditelja, pokazivač na lijevo i desno dijete. Svaki čvor sadrži broj elemenata što je u principu broj svih čvorova kojim je on tranzitivno roditelj + 1, jer se broji i on sam. Sadrži i generički tip element koji je zapravo vrijednost u tom čvoru. Posjeduje još dva atributa a to su statički bool napravljen_seed i prioritet. Prioritet je zapravo snaga tog čvora jer svi čvorovi koji su tranzitivno njegova djeca moraju imati manji prioritet od njega. Prioritet se nasumično generiše pri kreiranju čvora a napravljen_seed nam pomaže da pri svakom pokretanju programa nasumični brojevi budu drugačiji. Konstruktori i Destruktori su standardni, samo se treba pobrinuti da konstruktor dobro generiše nasumične brojeve za priroritet koristeći napravljen_seed.

Stablo ima jedan atribut i to pokazivač na čvor koji mu predstavlja korijen. Prije no što krenemo govoriti o konstruktorima i destruktorima govorit ćemo o nekim privatnim funkcijama koje pomažu. nadji_najmanji(Cvor*) i nadji_najveci(Cvor*) primaju čvor a nalaze najveći odnosno najmanji čvor koji je njegovo tranzitivno dijete. sljedeci(Cvor*) vraća sljedeći čvor u sortiranom redoslijedu a prethodni(Cvor*) analogno vraća prethodni. Konstruktor, destruktor, i operator dodjele koriste funkciju umetni() i izbrisi() kako bi napravili novo stablo i u njega umetnuli sve elemente prethodnog stabla i kada umetnemo, stavljamo čvorove takve da nam prioritet bude isti, pa nam stabla budu isto raspoređena. Konstruktor, destruktor i operator dodjele su svi $O(n \log n)$ jer su nam osnovne operacije $O(\log n)$. Svu asimptotsku analizu radimo pod pretpostavkom da je stablo dobro raspoređeno tj. da mu je visina $O(\log n)$ što je u našem slučaju vrlo vjerovatno.

Funkcionalnosti

nadji(Tip element) je generička funkcija koja traži da li se dati element nalazi u stablu. Ako da, vrati pokazivač na njega, a u suprotnom vraća nullptr. Nalaženje radi u $O(\log n)$ jer nam je visina stabla $O(\log n)$, a mi idemo na desno odnosno lijevo dijete roditelja u zavisnosti od veličine elementa.

umetni(Tip element) je generička funkcija umetne element u stablo. Stavimo ovaj element na dno stabla kao list nekog čvora i onda radimo nešto kao „upheap“ po prioritetu a pritom pazeći na to da broj elemenata ostaje isti kada vršimo rotacije. Radi u $O(\log n)$

izbrisi(Tip element) je funkcija slična kao prethodna, samo sada tražimo dati element, ako ga nema vratimo 0, a u suprotnom vratimo 1. Kada ga nađemo njegove roditelje i djecu zamjenimo sa djetetom koji ima veći prioritet. Neka desno dijete ima veći prioritet od lijevog sada će lijevo dijete elementa koji brišemo postati lijevo dijete desnog djeteta tog elementa. Pri tome dijete koje se nalazi skroz lijevo od desnog djeteta i najdesnije dijete lijevog djeteta u zavisnosti od prioriteta „zakačiti“ jedan za drugi, ovaj postupak opet ponovimo penjući se gore i na kraju cijeli blok zakačimo kao lijevo dijete desnog djeteta koji mora imati najveći prioritet. Takve djece ima maksimalno $O(\log n)$ jer se krećemo samo u jednom smjeru u stablu. Ova funkcija radi u $O(\log n)$.

Pomoćne funkcije

razdvajanje(Stablo<Tip>&, Tip) je generička funkcija koja razdvaja stablo na dva podstabla gdje jedan ima manje elemente od zadatog a drugi veće. Radi kako je i objašnjeno u zadatku.

spajanje(Stablo<Tip>&, Stablo<Tip>&) je funkcija koja spaja dva stabla koja su nastala razdvajanjem. Napravi se čvor čiji element je veći od svih elemenata prvog stabla a manji od svih elemenata drugog i također ima veći prioritet od svih elemenata. Sada napravimo stablo čiji je korijen ovaj čvor kojem je lijevo dijete prvo stablo, a desno dijete drugo stablo. Sada ovom stablu pomoću izbrisi() uklonimo korijen i dobijemo spojeno stablo, pošto je za element čvora dovoljno da je veći od elementa jednog a manji od elementa drugog korijena i da mu je prioritet veći od oba korijena. To nam je u konstantnom vremenu kao i kreiranje stabla jer su po referenci. Izbriši nam je u $O(\log n)$ pa nam je kompleksnost funkcije $O(\log n)$.

unija(Stablo<Tip>&, Stablo<Tip>&) je funkcija koja radi tačno kako je objašnjeno u zadatku. Njena kompleksnost je data sljedećom rekurzivnom relacijom $T(n) = 2T\left(\frac{n}{2}\right) + \lg n$ a ne možemo iskoristiti master theorem zato što ne postoji $k < 1$ takvo da $2\lg\left(\frac{n}{2}\right) \leq k\lg n \Rightarrow 2\lg n \geq k\lg n + 1 \Rightarrow k > 2$. Sada ako manuelno računamo $T(n) = \sum_{i=1}^{\lg n} 2^i \lg\left(\frac{n}{2^i}\right) = \sum_{i=1}^{\lg n} 2^i(\lg n - i) = lgn\sum_{i=1}^{\lg n} 2^i + \sum_{i=1}^{\lg n} i \cdot 2^i$ sada ove sume računajmo ovako:

$$\lg n \sum_{i=1}^{\lg n} 2^i = \lg n\left(2^{\lg n+1} - 1\right) = \lg n\,(2n-1) = 2n\lg n - \lg n = O(n\lg n)$$

$$\sum_{i=1}^{\lg n} i \cdot 2^i = \sum_{i=1}^{\lg n} 2^i + \sum_{i=1}^{\lg n} (i-1) \cdot 2^i = 2\lg n - 1 + 2 \cdot \sum_{i=1}^{\lg n} i \cdot 2^i - 2n\lg n$$

Sada neka je $\sum_{i=1}^{\lg n} i \cdot 2^i = S$ imamo $S = 2\lg n - 1 + 2S - 2n\lg n \Rightarrow S = 2n\lg n - 2\lg n + 1$.

Vidimo da je $S = O(n\lg n)$. $T(n) = O(n\lg n) + O(n\lg n) \Rightarrow T(n) = O(n\lg n)$.