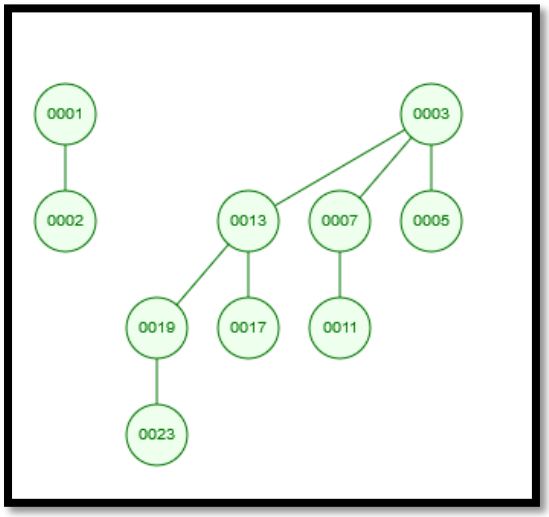
Naloga 3.1

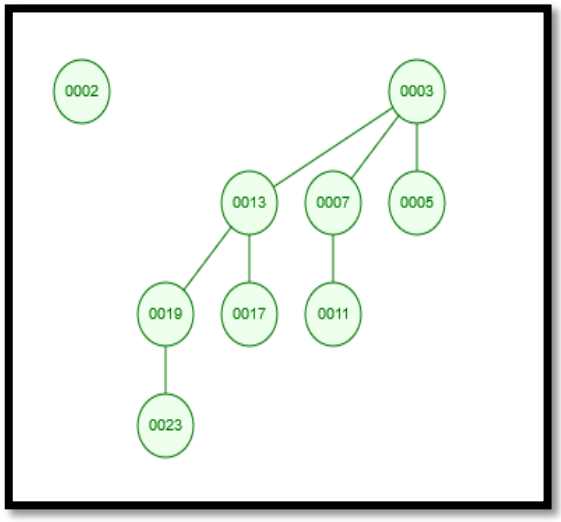
A)

a) Imejmo binomsko minimalno-urejeno kopico. Izvedite naslednje operacije Insert(23), Insert(19), Insert(17), Insert(13), Insert(11), Insert(7), Insert(5), Insert(3), Insert(2), ***Insert(1)****,* ***DeleteMin()****,* ***DecreaseKey(23, 1)****,* ***DeleteMin()****. Narišite stanje podatkovne strukture po koncu vsake poudarjene operacije.*

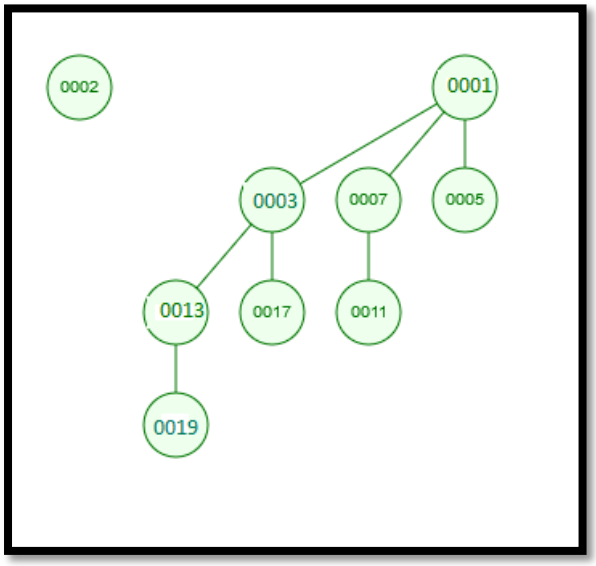
Insert(1)



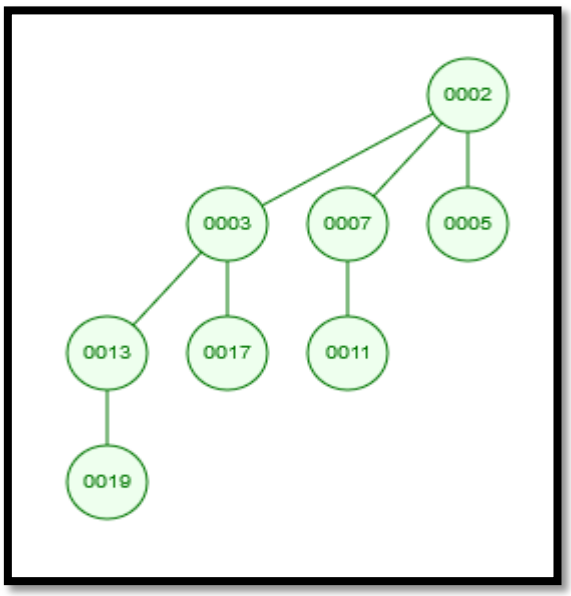
DeleteMin()



DecreaseKey(23,1),

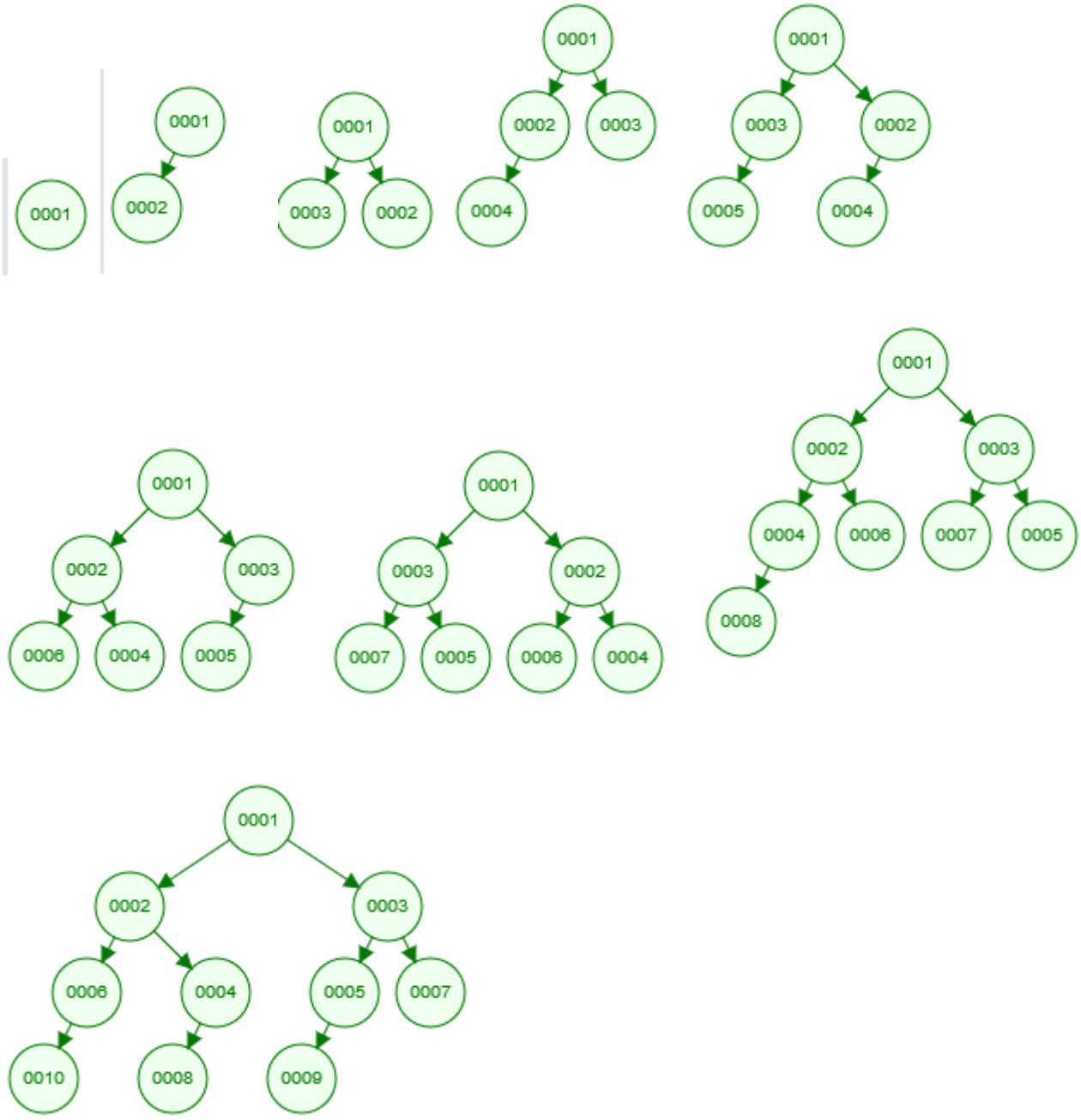


DeleteMin()



B)

V literaturi poiščite opis skrivljene kopice (angl. skew heap). Opišite, kako izgleda operacija zlivanja dveh skrivljenih kopic. V prazno omenjeno kopico vstavite elemente {1...8}. Ob koncu vsakega vstavljanja narišite sliko kopice. Za dodatne točke, narišite kopico po tem, ko vanjo vstavite po vrsti elemente {1, 2, 3, ..., n}.



Najprej primerjamo korene obeh kopic.

Koncni koren je koren kopice ki ima manjsi koren in koncno desno poddrevo je levo poddrevo kopice z manjsim korenom.

Za konec pa še primerjamo kočno levo poddrevo tako da rekurzivno združujemo desno poddrevo kopice z večjim korenom.

c)

V literaturi poiščite opis kopice min-maks (angl. min-max heap). V čem se razlikuje od običajne dvojiške kopice, katere operacije omogoča in kakšna je njihova časovna zahtevnost? V poljubnem programskem jeziku napišite funkcijo `boolean getLayer(int i)`*, ki z uporabo osnovnih aritmetičnih operacij (+, -, *, /, %, <<, >>), primerjavami (<, >, ==) in kontrolnimi stavki (if, while) za podan indeks vozlišča izračuna, ali je vozlišče na min (0) ali maks (1) nivoju. Koliko je časovna zahtevnost vaše funkcije v odvisnosti od podanega števila?*

Min-max kopica je popolna dvojiška struktura ki pa ubistvu vzame prednosti min kopice ina pa tudi max kopice Dobimo konstanni čas vračanja in logaritmični čas Brisanja. Vsak node in sodo stopnji je manjši kot pa vsi njegovi potomci dokler vsak node na lihi stopnji je pa večji od vseh svojih potomcev

```
V kopico vstavljamo kot pri običajni kopici,
Ko pa vstavimo x pogledamo:
*if  na max nivoju: check x < parent;
     Če je ga zamenja
*if na min nivoju: x > parent;
     Če je ga zamenja
To pa se nadaljuje rekurzivno do roota.
```