Garbage collection

Mange moderne programmeringsspråk har en *garbage collector*, som er en prosedyre som frigjør minne som garantert ikke vil brukes i programmet lenger. Du skal utlede en enkel algoritme for garbage collection.

Vi kan anta at alt som lagres er *objekter*, der et objekt kan referere til andre objekter. Vi lar en $G = (V, E)$ være en objektgraf, der V representerer alle objektene som er opprettet, og en (rettet) kant fra u til v betyr at objektet u har en referanse til objektet v. I tillegg har vi en mengde R med alle objektene som kan refereres til direkte (typisk objekter som refereres til av programvariabler). Alle objekter i R er også i V. Objekter det *ikke* finnes en referanse til via objekter i R er garantert å ikke bli brukt i programmet, og skal derfor frigjøres.

Det betyr at ingen av objektene i R skal frigjøres, og heller ingen objekter som kan nås gjennom referanser fra objekter i R skal frigjøres. Objektene som skal frigjøres er ikke i R, og kan heller ikke nås fra noe objekt i R.

Anta at du har en prosedyre `Free` som frigjør et objekt. Du skal gi en prosedyre `GarbageCollect` som tar en graf $G = (V, E)$ og en mengde R som input, og kaller `Free` på alle objekter det *ikke* kan nås fra R.

Input: En objektgraf $G = (V, E)$ og en mengde R med objekter
Output: Frigjør alle objekter det ikke finnes en referanse til

```
1 Procedure GarbageCollect(G, R)
    // ···
```