V této úloze se nabízí díky tomu, že pracujeme se stromem, napsat algoritmus rekurzivně.

Nejprve budeme uvažovat, jak pro jednu prodlužovačku zjistíme, jaké zařízení vypojit, když do této prodlužovačky jsou připojeny jen zařízení. Na to použijeme následující postup: nejprve zjistíme příkony zapojených zařízení a jejich součet, a pak odpojujeme zařízení s maximálním příkonem, dokud není součet menší než maximální příkon, protože tím jedině dosáhneme nejnižšímu počtu kroků.

Tím se dostáváme k případu, kdy do prodlužovačky bude zapojena další prodlužovačka. Protože když budeme znát všechna zařízení zapojené přímo či nepřímo do zapojené prodlužovačky, můžeme tento případ vyřešit stejně jako kdyby všechna zařízení byla zapojena rovnou do zpracovávané prodlužovačky. Tohle však můžeme udělat, jakmile všechny dceřiné prodlužovačky jsou zpracované, proto budeme zpracovávat od spoda nahoru, aby byl výsledek správný.

Implementaci tohoto algoritmu můžete vidět v pseudokódu níže. Pro efektivní získávání maxima využívám maximovou haldu. K nejhorší časové složitosti pak dojde, když se všechna zařízení odpojí v prvních prodlužkách, protože v každé vrstvě dojde ke slévání hald, což celkem potrvá $\mathcal{O}(NL)$ pro počet vrstev L a pro počet zařízení N, a odpojení všech zařízení $\mathcal{O}(N \log N)$.

```
Funkce prodluzovacky(vrchol v, int &pocetOdpojenych, nafPole<int> &kOdpojeni)
    -> (maxHalda<(int, int)>, int):
  // zarizeni budu zpracovavat jako pary dvou cisel, kde prvni
  // je pozadovany prikon a druhy ID zarizeni, abych mohl vypsat,
  // jake presne zarizeni vypojit
  maxHalda<(int, int)> h
  int s = 0 // suma
  (int,int) max // aktualni zarizeni s maximalnim prikonem
  Pokud v.deti.pocet = 0: // kdyz se jedna o zarizeni (list)
    h.pridat((v.prikon, v.cislo))
    s <- s + v.prikon
    Vratit (h, s)
  Jinak:
    Pro u ve v.deti:
      (maxHalda<(int, int)>, int) vysledek = prodluzovacky(
        u, pocetOdpojenych, kOdpojeni
      )
      h.spojitPole(vysledek[0]) // predpokladam, ze halda je ukladana v poli
      s <- s + vysledek[1]
    h.vyrovnat();
  Dokud v.P < s:
    max <- h.extrahovatMax()
    s <- s - max[0]
    kOdpojeni.pridat(max[1])
  Vratit (h, s)

int pocetOdpojenych
nafPole<int> kOdpojeni
prodluzovacky(koren, pocetOdpojenych, kOdpojeni)
Vratit (pocetOdpojenych, kOdpojeni)
```