Martin Vaško
úloha 3

Je zjavné že bude existovať také ideálne riešenie kde bude výška domov najskôr neklesať až príde do maxima (takúto výšku môže mať aj viac domov) a potom bude len ostro klesať. Je to preto lebo ak existuje nejaké iné ideálne riešenie ktoré nespĺňa tieto vlastnosti tak pre usporiadaním domov z neho vieme spraviť usporiadanie ktoré spĺňa tieto vlastnosti. To spravíme tak že usporiadame domy podľa veľkosti, z každej výšky zoberieme jeden pre klesanie a ostatné necháme usporiadané od začiatku. Aby sme toto vedeli spraviť potrebujeme aby z každej výšky boli aspoň dva domy (okrem najvyššieho domu). Toto ale zjavne musí platiť ak aj pôvodné usporiadanie bolo podľa zadania úlohy. To preto lebo ak sa výška niekedy dosiahne X tak musia existovať aspoň dva domy s výškou X-1, jeden pred a jeden za.

Preto je zjavné že najvyšší dom nemôže byť vyšší ako n (dokonca ako n/2 ale to pre odhad časovej zložitosti nie je podstatné).

Takže algoritmus bude fungovať nasledovne:

Najskôr spočíta koľko domov danej výšky zákazníci chcú (domy vyššie ako n nepočítame). Potom postupne pre každú maximálnu výšku od 1 (alebo od 0 ak v úlohe môžu byť aj zákazníci ktorý chcú 0 poschodový dom) do n pričom si budeme udržiavať dve premenné a to počet miest kde môžu ísť vyššie domy ako aktuálny najvyšší a počet domov ktoré vyhovujú zákazníkom. Pri každom kroku aktualizujeme tieto premenné nasledovne:

Počet domov ktoré vyhovujú zákazníkom zvýšime o minimum z počtu voľných miest a počtu domov danej výšky. Počet voľných miest znížime o maximum z 2 a počtu domov ktoré sme práve pripočítali k počtu domov ktoré vyhovujú zákazníkom. Vždy musíme odpočítať aspoň 2 voľné miesta lebo ich potrebujeme na stavanie vyšších domov. Tento algoritmus vždy postaví najviac vyhovujúcich domov ako môže. To môžeme spraviť lebo ak by sme ich nepostavili tak jediné čo by sme s tými ďalšími voľnými miestami mohli robiť je postaviť prinajlepšom rovnako veľa domov ako sme predtým nepostavili. Okrem toho si zapisujeme koľko domov sme v každom kroku postavili. Nakoniec vypíšeme postavené domy v poradí aké bolo opísané na začiatku riešenia.

Oba kroky algoritmu trvajú O(N) a to je teda aj celková časová zložitosť. Pamäťová zložitosť bude tiež O(N).

```
count = [0]*n

for i in domy:
   if domy <= n:
       count[i] += 1

volne = n
postavene = 0
postavene_s_vyskou = [0]*n
max_vyska = 0

for i in range(n):
   nove_domy = min((count[i], volne))
   postavene += nove_domy
   volne -= max((2, nove_domy))
   postavene_s_vyskou[i] = nove_domy
   if nove_domy > 0:
       max_vyska = i

for i in range(len(postavene_s_vyskou)):
   print(f"{i} "*(postavene_s_vyskou[i]-1))

for i in range(max_vyska, 0, -1):
   print(i)
```