Úvod do myšlenky

Podle zadání, nemůže být řešení závislý na počet mravenců, což může být až N x N. Tohle chápu tak, že bychom neměli procházet všechny mravenci a zkontrolovat, jestli na ně spadne šiška nebo ne. Ale řešení vyžaduje záchranu co nejvíce mravenců, tudíž v případě, kdy máme tak málo krabiček, že nemůžeme zachránit každého mravence, který je v ohrožení. V takovém případě musíme umět rozeznat, který mravenec je v ohrožení, a který ne, abychom mohli chránit jenom ty ohrožené.

Protože nemůžeme procházet každého mravence, zbývá nám možnost projít všechny oblasti, na které padají šišky a ochránit všechny mravence v té oblasti, dokud ještě máme krabičky.

Základní princip algoritmu hladové zachránění každého mravence, kterého najdeme pod plochou dopadu šišky. Problém tohoto základu je, že v případě, kdy zachráníme mravence jednou, dojdou nám kravičky a na před-tím zachráněného mravence spadne další šiška. V takovém případě jsme se pokusili zachránit mravence, který ve finále nezachráníme.

Můžeme také poznamenat, že výstupem algoritmu má být pozice a čas pokládání každé krabičky, a protože na záchranu jednoho mravence bude občas potřeba více než jedna krabička, je validní si ukládat všechny mravence, který zachráníme, protože to zabere míň místa než samotný výstup programu.

Algoritmus na řešení

Začneme už průběhem algoritmu — vezmeme první šišku a projdeme celou jeho plochu dopadu. Jestli nenarazíme na mravence, jdeme dál, jestli narazíme na mravenec, uložíme si jeho pozice do stromu a zároveň do slovníku, kde klíč je jeho pozice a hodnota je počet šišek, které na něj spadnou. Doprohlédneme plochu dopadu první šišky a jdeme na druhou. Když narazíme na mravence, zkontrolujeme nejdřív slovník. Jestli už je v tabulce, v této době, hodnota v slovníku je 1, tudíž víme, že je v prvním stromě, a že dopadne na něm *ještě* jedna šiška, tedy budeme potřebovat 2 šišky zachránit tohoto mravence. Proto Zvýšíme hodnotu v slovníku, projdeme první strom, odstraníme jeho pozici ze stromu a vytvoříme druhý strom, který bude mít uložený pozice všech mravenců, které potřebují 2 krabičky na zachránění. Takhle pokračujeme dál, dokud neprojdeme všechny šišky. Algoritmus nemůžeme ukončit dřív, protože možná přehlédneme spoustu mravenců, které vyžadují jenom jednu krabičku na zachránění.

Tedy, budeme mít různé stromy pro všechny různé počty potřebných krabiček na záhchranu mravence na té pozici. Na konci můžeme projít všechny stromy, počínaje od prvního stromu, a za každého zachráněného mravence odečteme potřebných počet krabiček na jeho záchranu. Tímto vybíráme nejdřív z 'nelevnějších' mravenců na záchranu, což zřejmě maximalizuje počet zacháněných mravenců.

A i v případě, kdy bychom nemohli poskládat více krabiček na sebe, tak stačí druhý slovník, kde pro každou pozici si uložíme pole časů dopadů šišek, před kterých je zachraňujeme.

Stromy tedy celkově ukládají všechny mravence pod ploch dopadů šišek, což může být až N x N. Nabízí se nám ale možnost zmenšovaní celkového počtu uložených pozic mravenců a to je, že si zároveň budeme ukládat počet krabiček potřebných na záchranu všech uložených mravenců. Když toto číslo přesáhne počet krabiček co máme, můžeme bez pochybu odstranit libovolné pozici uložené v posledním stromě, tedy strom 'nejdražších' mravenců na záchranu. Tímto omezíme celkovou velikost stromů na počet zachraněných mravenců, což jsme už v úvodu určili, jako menší než výstup našeho programu.

První slovník, který si ukládá počet krabiček na záchranu mravence, je vlastně stejnou velikost jako celkovou velikost všech stromů. Druhý slovník, který si ukládá všechny časy dopadů šišek, které padají na mravence, což bude vlatně náš výstup, což je $\mathcal{O}(K)$.

A teď časovou komplexitu. Njedřív si řekneme, že stromy, do kterých si ukládáme všechny pozice, jsou uspořádané, vyvážené AVL stromy, kde zachováváme logaritmickou komplexitu přidání, vyhledávání a odebírání. Situaci procházení všech ploch dopadů šišek se vyhneme pouze v případě, kdy máme v prvním stromu stejný počet pozic jako počet krabiček. Ve všech ostatních případech se nevyhneme procházení všech ploch dopadů šišek, což může být až N x N. Nejhorší případ našeho algoritmu nastance, když máme v prvním stromě $K-1$ prvků, a všechny další šišky budou mít stejnou plochu dopadu jako do teď, což znamená, že přesuneme první strom do druhého stromu, ale zachováme jen polovinu pozic, a potom bude následovat další posun stromu, a tak dále. Toto celkově vyjde na $(K + \frac{K}{2} + \frac{K}{3} + \cdots + \frac{K}{K}) \cdot \log K < K\sqrt{K} \cdot \log K$, pro větší K. Zavedeme h jako kolikáty strom jsme přesunuli původních $K-1$ prvků a můžeme říct, že po každém posunu těch prvků jsme našli ještě právě 1 mravenec, který zachráním jednou krabičkou a přidáme k hornímu odhadu našeho algoritmu $h \cdot \log h$. Je otázkou, jestli toto není vlastně zanedbatelný, ale raději to zapíšu do komplexity.

Celková časová komplexita vychází jako $\mathcal{O}(N^2 + \min(N^2,\ K\sqrt{K}) \cdot \log K + h \cdot \log h)$ (procházení všech šišek + ukládání přesun starých mravenů + ukládání nových mravenců). Jediný způsob, co jsem vymyslel, jak by algoritmus nemusel být závislý na N^2, tedy celou plochu šišek, je kdyby bylo zaručeno, že najdeme alespoň K mravenců, které vyžadují jenom 1 krabičku na záchranu.