A hangya feladat megoldása:

Minden épülő városnál két dolgot csinálunk: „frissítjük” a megfelelő intervallumot és lekérjük a másik intervallum összegét. Ehhez szegmensfát használunk. A sima szegmensfa intervallum update függvénye az intervallum hosszától függ, ennél jobb komplexitást is el tudunk érni. A szegmensfa „lazy update” változatát használva, ahol egy adott intervallum frissítése log n idő marad, a lekérdezés és a frissítés működni fog log n időben. (Ez úgy működik igazából, hogy intervallum update-nél minden az intervallumban lévő legfelső node-nál nem megyünk lejjebb, azon kívül pedig az intervallum két végét visszük csak le a levelekig. A NEM levél belső node-oknál csak eltároljuk, hogy mennyivel kéne az őt tartalmazó elemeket frissíteni, egy lekérésnél pedig, ha az egész node-ot tartalmazza a lekérdezés, csak az intervallum hosszával megszorozzuk azt, hogy elemenként mennyi a növekedés. Ha egy lekérdezés nem tartalmazza az egész belső node-ot, akkor a node mindkét gyermekébe „lejjebb visszük” az összeget és így adogatjuk hozzá. Én innen ismerem: <https://www.geeksforgeeks.org/lazy-propagation-in-segment-tree/?ref=rp>)

Ahhoz viszont, hogy N (akár 100 000) új szegmensfát csináljunk egyenként M (akár 100 000) hosszú tömb elemeire, nem elég az O(M)-es szegmensfa készítési idő. Ezt a problémát úgy lehet kiküszöbölni, hogy egy „új városnál” a nem változó részeket nem duplikáljuk, tehát csak az L-R intervallumot kell „duplikálni”, amit a lazy update-et kihasználva szintén „könnyen” meg tudunk tenni. Ez azt jelenti, hogy a node-okból, amik benne vannak a frissítés intervallumában csinálunk egy duplikálást, de mindnek csak a legfelső szintjéről. Tehát ha az 1. város összes elemét növeljük pl 5-tel, akkor elég 1 új node-ot csinálni, hiszen a root node az összes elemet tartalmazni fogja. A legrosszabb eset, hogy egy intervallum „duplikálásakor” 2log M (mindkét oldalán teljesen a levélig vezető úton) új elem fog kelleni. Ez azt jelentené, hogy N \* 2log M tárhely kellene, és a lekérdezésekkel együtt a komplexitás O(N \* log M).