Homework 2

Jakub Senko, Štefan Uherčík

13. apríla 2014

Príklad 1

Zaveďme všeobecnú reprezentáciu budov. Každá uvažovaná budova sa dá reprezentovať ako množina dvojíc (x_k, h_k), určujúcich výšku budovy h na súradnici x. Zápis sa dá zjednodušiť usporiadaním bodov vzostupne podľa x. Stačí uvažovať len tie dvojice, ktoré označujú miesto, v ktorom nastáva zmena výšky budovy. Tento zápis je ekvivalentný so zápisom použitým v zadaní

$$(1, \mathbf{5}, 5) \sim ((1, 5), (5, 0)) \tag{0.1}$$

ide len o vnútornú reprezentáciu za účelom zjednodušenia algoritmu.

Merge

Uvažujme algoritmus MERGE, ktorý z reprezentácie dvoch budov vypočíta reprezentáciu ich siluety.

Algoritmus využíva object BUILDING_ITERATOR pomocou ktorého je možné postupne prechádzat reprezentáciou danej budovy. Obsahuje tri metódy.
NEXT_COORDINATE_EXISTS a NEXT_COORDINATE_POSITION sú triviálne a neposúvajú pozíciu iterátora. Tretia metóda, GET_HEIGHT(x) vráti výšku budovy na zadanej súradnici. Táto metóda spôsobí dostatočný posun iterátora v prípade, že zadaná pozícia je väčšia alebo rovná ako NEXT_COORDINATE_POSITION. Kedže iterátor

je jednorázový, túto metódu je nie je možné zavolať s argumentom menším ako v predchádzajúcom volaní. Iterátor si jednoducho pamätá poslednú výšku.

Samotný MERGE pracuje s dvoma iterátormi, pre každú budovu jeden a výstup postupne ukladá do samostatného zoznamu. Základom je *while* smyčka, ktorá sa vykoná ak aspoň pre jeden s iterátorov platí NEXT_COORDINATE_EXISTS. Algoritmus potom vybere menšie x z NEXT_COORDINATE_POSITION a zavolá metódu GET_HEIGHT na oboch iterátoroch. Následne vybere väčšiu z výšok, h a zavolá funkciu TRY_ADD, ktorá jednoducho vloží novú súradnicu (x, h) do výsledného zoznamu v prípade, že sa výška siluety zmenila (čo nemusí nastať).

Tento algoritmus funguje pre ľubovolné reprezentácie s dĺžkou n_1, n_2 v čase $\mathcal{O}(n_1 + n_2)$, čo je $\mathcal{O}(n)$ pre budovy s rovnako veľkou reprezentáciou. Zdôvodnenie je jednoduché - využíva jednosmerný iterátor na jedno použitie pre každú reprezentáciu - a teda každú súradnicu spracuje práve raz. Algoritmus je konečný pretože pri každom priechode cyklom metóda GET_HEIGHT posunie aspoň jeden z iterátorov.

Rozdeľ a panuj

Výslednú siluetu dosiahneme aplikovaním funkcie MERGE na vhodné podproblémy. Toto delenie funguje rovnako ako pri algoritme *merge sort*. Funkcia COMPUTE_SILHOUETTE zoberie ako argument množinu reprezentácii budov. Ak táto množina obsahuje jednu budovu, tak ju vráti. Ak dve budovy, zavolá na nich MERGE a vráti výsledok. Ak viac, rozdelí množinu na dve rovnaké (s rozdielom jednej budovy v prípade nepárneho počtu) množiny, rekurzívne sa na oboch zavolá a výsledok znovu spojí pomocou MERGE a vráti. Týmto spôsobom funkcia COMPUTE_SILHOUETTE vždy vráti merge všetkých spojich argumentov (merge nezávisí na poradí).

Zložitosť závisí na počte MERGE operácii a veľkosti ich vstupu. Na každej úrovni rekurzie je suma veľkosti všetkých reprezentácií rovnaká (n dĺžky 2 na začiatku vs dve dlhé n na konci, kde n je počet budov) a počet úrovní je $\log_2 n$. Výsledná zložitosť je teda $\mathcal{O}(n \log n)$

Príklad 2

Príklad 3

Príklad 4

Príklad 5