Trik s konveksno ovojnico

Jakob Žorž

2. december 2023

1 Časovna kompleksnost

Preden se lotimo reševanja problema, si oglejmo, kaj je časovna kompleksnost in kako je uporabna:
Velik problem pride pri tem, da ni čas izvedbe vedno popolnoma enak, včasih je hitrejši, včasih počasnejši. Zelo je odvisno od drugih procesov in preostalih okoliščin, zato opišemo čas izvajanja bolj približno. Namesto, da konkretno povemo, koliko časa potrebuje algoritem, povemo, kako se čas izvajanja spreminja, ko se velikost vhoda spreminja. Večina časa je ta ocena dovolj dobra, saj hitra rešitev porabi desetkrat ali pa celo stokrat manj časa, kot je na voljo. Počasna resitev pa porabi lahko potencialno tudi tisočkrat več časa, kot je na voljo, zato je dovolj groba ocena.
Primer:

- Algoritem, ki gre skozi seznam in izpiše vse elemente, ima časovno kompleksnost $O(n)$, kjer je n dolžina seznama.
- Algoritem, ki gre skozi seznam in izpiše vse pare elementov, ima časovno kompleksnost $O(n^2)$, kjer je n dolžina seznama.
- Algoritem, ki uredi seznam po vrsti, ima časovno kompleksnost $O(n \log n)$, kjer je n dolžina seznama. Dokaz časovne kompleksnosti urejanja bomo tukaj opustili, saj ni pomemben del dokumenta.

Velikokrat je časovna kompleksnost le zgornja meja in se včasih izkaže, da je algoritem hitrejši, kot je bila ocena. To novo oceno se da matematično dokazati a je velikokrat precej zapleteno.
Časovno kompleksnost ima tudi matematično definicijo:

Definicija Naj bo f funkcija, ki sprejme naravno število n in vrne realno število. Časovna kompleksnost algoritma je $O(f(n))$, če obstajata pozitivni konstanti c in n_0, da velja:

$$\forall n \geq n_0 \, : \, \text{čas izvajanja algoritma} \leq c \cdot f(n)$$

Ta definicija izgleda precej zapletena, a je v resnici precej preprosta. Razmislek o tej definiciji je prepuščen bralcu.

2 Problem

Oglejmo si naslednji problem:

Definicija Igraš igro, kjer imaš n različnih stopenj. Vsaka stopnja ima neko pošast, ki ima določeno moč in se lahko odločimo, da jo premagamo ali pa preskočimo. Premagovanje pošasti nam vzame $s_i \cdot f$ časa, kjer je s_i moč pošasti in f naša spretnost (pozor: nižja kot je spretnost, manj časa potrebujemo). Če pošast preskočimo, nam to ne vzame časa. Ko premagamo pošast, se nam spretnost nastavi na f_i. Cilj igre je, da premagamo zadnjo pošast v čim manj časa. Napiši program, ki dobi n - število stopenj, s_i - moč pošasti na i-ti stopnji in f_i - spretnost, ki jo dobimo, ko premagamo i-to pošast in izpiše najmanjši čas, ki ga potrebujemo, da premagamo zadnjo pošast.

Ena izmed možnih rešitev je, da gremo skozi vse možnosti in izberemo najboljšo. Čeprav je ta rešitev pravilna, je časovno precej neučinkovita in nam ne bo prinesla veliko točk na tekmovanju. Časovna kompleksnost te rešitve je $O(2^n)$, ker je na vsaki stopnji dve možnosti: premagamo ali preskočimo pošast.

3 Optimizacija z dinamičnim programiranjem

4 Optimizacija s konveksno ovojnico