**Neformalni uvod**

*1.10.2013*

***Problem*** Π smo neformalno definirali kot neskončno množico *računskih nalog*, ki imajo vse enako strukturo in jih želimo reševati po istem postopku, med seboj pa se razlikujejo po vrednostih parametrov (*podatkov*).

***Računski postopek*** za reševanje problema Π smo neformalno definirali kot končno zaporedje ukazov, katerega izvrševanje nas pri vsaki nalogi problema Π v končno mnogo korakih privede do rešitve. Za *velikost naloge* smo vzeli dolžino zapisa podatkov naloge in definirali ***časovno zahtevnost*** algoritma *A* kot funkcijo *TA*:*N*→*N*, kjer je *TA*(*n*) največje število korakov, ki jih algoritem *A* napravi pri reševanju nalog velikosti *n*.

Za funkciji *f*,*g*:*R*+→*R* smo definirali oznako *f*(*x*)=O(*g*(*x*)).

**Definicija**. Algoritem *A* je ***polinomski***, če obstaja polinom *p*(*n*), tako da za vse *n*∈*N* velja: *TA*(*n*)≤*p*(*n*).

Na nekaj zgledih smo primerjali hitrost rasti časovne zahtevnosti polinomskih in nepolinomskih algoritmov. Na podlagi te primerjave polinomske algoritme neformalno enačimo z *učinkovitimi algoritmi*. Seveda se je pri tem treba zavedati, da se prednosti algoritmov s počasneje rastočo časovno zahtevnostjo v splošnem pokažejo šele pri dovolj velikem *n*.

*8.10.2013*

Časovna zahtevnost algoritma je odvisna tudi od uporabljene ***kodirne sheme***, t.j. načina zapisovanja podatkov. Dogovorili smo se, da bomo naravna števila zapisovali v nekem mestnem številskem sistemu z osnovo *b*≥2, tako da je npr. velikost naloge s podatkom *n*∈*N* enaka ⌊log*bn*⌋+1 in ne *n*.

Omejili se bomo na analizo časovne zahtevnosti ***odločitvenih problemov*** in ***optimizacijskih problemov***. Pokazali smo, kako optimizacijskemu problemu priredimo njegovo odločitveno različico.

Definirali smo pojem ***formalnega jezika*** kot neke množice besed nad izbrano neprazno končno abecedo. Definirali smo *prazno besedo* *ε*, *dolžino* besede, *obratno besedo*, *stikanje besed*, *obratni jezik*, *stikanje jezikov*, *potence jezikov* ter *pozitivno* in *Kleenejevo zaprtje* formalnega jezika. Pokazali smo, kako lahko s pomočjo ustreznih kodirnih shem vsak odločitveni problem predstavimo kot problem pripadnosti dane besede nekemu formalnemu jeziku.