Algoritmické riešenia ťažkých problémov
Domáca úloha 3

Autor: Marián Kravec

Úloha 1 - Random permutations

a)

Ukážeme to induktívne.

Najprv ako bázu indukcie budeme mať zoznam dĺžky 1. Pre takýto zoznam existuje iba jedna permutácia čo je samotný pôvodný zoznam a keďže náš algoritmus začína prehadzovať od druhej pozície na tomto zozname nikdy nič nezmení, takže spĺňa, že vytvorí všetky permutácie (jednu) s rovnakou pravdepodobnosťou (keďže obe pozície s ktorými vymenil).
Pre istotu ešte ako bázu použime zoznam dĺžky 2. V tomto zozname spraví náš algoritmus iba jednu zmenu a to, že druhý člen zoznamu vymení s buď sám sebou alebo prvým (obe s rovnakou pravdepodobnosťou) čím vytvorí 2 permutácie čo je 2! čím spĺňa, že vytvorí všetky permutácie s rovnakou pravdepodobnosťou.

Teraz uvažujme, že náš vlastnosti nášho algoritmu platia pre všetky zoznamy dĺžky $k < n$.

Chceme ukázať, že platia aj pre zoznam dĺžka n. Vieme, že po $n-1$ krokoch náš algoritmus vytvorí jednu z $(n-1)!$ permutácii s rovnakou pravdepodobnosťou (indukčný predpoklad) ktorú označíme $p_{n-1} = \frac{1}{(n-1)!}$. V poslednom kroku vymení posledný člen s niektorým z n členov (všetky výmeny majú rovnakú pravdepodobnosťou). Čiže nakoniec dostaneme jeden z $n!$ zoznamov kde každý vznikol s pravdepodobnosťou $p_{n-1}\frac{1}{n} = \frac{1}{(n-1)!}\frac{1}{n} = \frac{1}{n!}$ (keďže doterajšie pravdepodobnosti boli všetky rovnaké a pravdepodobnosť poslednej výmeny je rovnomerná). Teraz už iba potrebujeme ukázať, že tento posledný krok nemôže vytvoriť dva rovnaké zoznamy. To dokážeme sporom.

Uvažujem, že po poslednom kroku máme 2 zoznamy $A = [a_1, ...a_n]$ a $B = [b_1, ...b_n]$ ktoré sú rovnaké, čiže platí $\forall 1 \leq i \leq n, a_i = b_i$, ale pred týmto krokom to boli dve rôzne permutácie $(n-1)$ prvkov.
Bez ujmy na všeobecnosti uvažujme, že v poslednom kroku vymenil algoritmus a_i s a_n a b_j s b_n, keďže na konci bola na n-tej pozícii tá istá hodnota muselo pred výmenou platiť $a_i = b_j$ a keďže na konci bola hodnota n na rovnakej pozícii (vždy je inicializovaná na n-tej pozícii a predchádzajúce kroky ju nevedia posunúť) tak musí platiť $i = j$, takže pred poslednou výmenou platilo $a_i = b_i$ a keďže okrem tejto pozície posledný krok nezmenil žiadnu inú a na konci platilo $\forall 1 \leq k \leq n, a_k = b_k$. Tak pred týmto krokom muselo platiť $\forall 1 \leq k \leq n-1, k \neq i, a_k = b_k$ a keďže platí aj $a_i = b_i$ tak pred posledným krokom boli zoznamy A a B rovnaké čo je v spore s predpokladom, že ide o dve rôzne permutácie.

Čiže sme ukázali, že po poslednom kroku budeme mať jednu z $n!$ rôznych permutácii každú s rovnakou pravdepodobnosťou, čo je to čo sme chceli ukázať. □

b)

Keďže vieme generovať iba po jednom bite, vygenerujeme naše číslo binárne. Algoritmus na začiatok vypočíta rozdiel $b-a$ aby zistil koľko čísel existuje medzi dvomi vstupnými číslami. Následne vygeneruje $\lceil log(b-a) \rceil$ bitov, čím s rovnakou pravdepodobnosťou vygeneruje číslo z intervalu $[0, ..2^{\lceil log(b-a) \rceil}]$ (keďže každé číslo má unikátny binárny zápis a každý postupnosť bitov má rovnakú pravdepodobnosť vygeneruje všetky s rovnakou pravdepodobnosťou). A výsledné číslo pripočíta k a. Nakoniec skontroluje či je vygenerované číslo menšie nanajvýš rovné b, ak nie tak proces generovania zopakuje inak výsledné číslo vráti.

Generovanie jedného čísla je generovanie $\lceil log(b-a+1) \rceil$ bitov, to vieme zhora ohraničiť pomocou $log(n)$ (n je najvyšší rozdiel ktorý dostaneme) bitov. Čiže zložitosť generovanie jedného čísla je $O(log(n))$. Avšak vždy môže nastať, že číslo je väčšie ako b a generovanie musíme zopakovať. Preto celková zložitosť môže byť až $O(k * log(n))$ kde k je počet opakovaní generovania čísla, čo môže byť teoreticky nekonečno.

Teraz odhadnúť strednú hodnotu tejto zložitosti.
Najprv zistíme, aká je pravdepodobnosť, že číslo bude väčšie ako b, keďže vieme, že vygeneruje rovnomerne celé číslo medzi 0 a $2^{\lceil log(b-a) \rceil}$ stačí nám zistiť aký zlomok z týchto čísel je väčších ako $b-a$, pričom $b-a$ vieme zapísať aj ako $b-a = 2^{log(b-a)}$.

Keďže generujeme prirodzené čísla od nuly tak pravdepodobnosť, že vygenerované číslo je väčšie ako x je $1 - \frac{x}{\text{maximálne vygenerované číslo}}$. Čiže v našom prípade:

$$
\begin{aligned}
P(g > 2^{log(b-a)}) &= 1 - \frac{2^{log(b-a)}}{2^{\lceil log(b-a) \rceil} - 1} < \\
&< 1 - \frac{2^{log(b-a)}}{2^{\lceil log(b-a) \rceil}} = \\
&= 1 - 2^{log(b-a) - \lceil log(b-a+1) \rceil} \leq \\
&\leq 1 - 2^{log(b-a) - (log(b-a)+1)} = \\
&= 1 - 2^{log(b-a) - log(b-a) - 1} = \\
&= 1 - 2^{-1} \\
&= \frac{1}{2}
\end{aligned}
$$

Takže vidíme, že pravdepodobnosť, že budeme musieť generovať číslo znovu je menšia ako $\frac{1}{2}$, teraz môžeme počítať strednú hodnotu časovej zložitosti:

$$
\begin{aligned}
E(ALG(n)) = \sum_{k=1}^{\inf} log(n)\frac{k}{2^k} \\
& (log(n)\text{-čas generovania,} \\
& k\text{-počet opakovaní,} \\
& \frac{1}{2^k}\text{-pravdepodobnosť } k \text{ opakovaní)} \\
& = log(n)\sum_{k=1}^{\inf}\frac{k}{2^k} \\
& \text{(sumu si prepíšeme, ZDROJ)} \\
& = log(n)\lim_{k->\inf}\frac{2^{k+1}-n-2}{2^k} \\
& = 2log(n)
\end{aligned}
$$

Čiže vidíme, že v strednej hodnote algoritmus potrebuje $2log(n)$ krokov, čiže logaritmický čas. Avšak úplne najhoršom prípade algoritmus nikdy neskončí.

Algoritmus by mohol vyzerať asi takto:

```
# vypocet poctu generovanych bitov
n = b-a
k = ceil(log(n))

# inicialny vystup abi zacal cyklus
# (da sa to riesit aj pouzitim "do while" namiesto takejto inicializacie)
vystup = b+1

# opakovanie kym vystupne cislo nesplna, ze patri do pozadovaneho intervalu
while vystup >= b:
        # anulovanie generovaneho cisla
        generovane = 0

        #generovanie cisla
        for i in range(k):

                # vygenerovanie nahodneho bitu
                bit = gen_bit()

                #pridanie bitu na koniec cisla
                generovane = generovane*2+bit

        # vypocet vystupneho cisla
        vystup = a + generovane

# vratenie vystupu
return vystup
```