1. Tá o sumách podmnožín (25 bodov)

(a) Jazyk A je polynomiálne redukovateľný na jazyk B ak existuje funkcia f s polynomiálnou časovou zložitosťou, ktorá transformuje slová zo σ_A^* do σ_B^* tak, aby platilo, že $x \in A \Leftrightarrow f(x) \in B$.

(b) Množinu A necháme nezmenenú, množina B obsahuje iba číslo t. Treba popísať, prečo platí ekvivalencia uvedená vyššie.

(c) Nová množina A musí obsahovať nejakým spôsobom aj prvky z A aj z B. Zároveň musíme zaručiť, že súčet t bude existovať iba ak vyberieme práve jednu z pôvodných hodnôt v B a zodpovedajúcu sadu čísel z množiny A. Dobrý trik je zvoliť si nejaké pevné t a čísla z B "nafúknuť" tak, aby sa do t zmestilo najviac jedno – napríklad vynásobiť tie čísla nejakým 10^k. To zároveň pekne rozlíši hodnoty z A a B. Pridáme ešte hodnoty na nasčítanie do t, ku každému číslo z B práve jednu. Odporúčam pozrieť riešenia k cvičeniu 6, kde sa tieto triky aplikovali.

2. Tá o zvieratách (25 bodov)

Na zisk 10 bodov stačilo riešenie so zložitosťou $O(t)$. Takým je napríklad úplne priamočiare počítanie zadaných vzorcov v jednom cykle. Iba si treba dať pozor, aby sa hodnoty 3^t nepočítali vždy nanovo, ale využívala sa hodnota z predchádzajúcej iterácie vynásobená 3.

Optimálne riešenie využíva umocňovanie matíc. Tento postup nájdete vysvetlený v prednáške "Dynamické programovanie 2 (riešenie zložitejších úloh)". Spočítať hodnoty múch, žáb a bocianov ide úplne priamočiaro technikami spomenutými v prednáške. Líšky treba trochu vymyslieť ako započítať 2^t každú druhú iteráciu.

3. Tá s Kruskalom (25 bodov)

(a) Ak vieme, ktoré hrany sú obsadené, vieme kadiaľ sa dajú potenciálne presúvať figúrky. Stačí sa preto sústrediť na každý komponent súvislosti zvlášť. Pre každý komponent musíme spočítať, koľko figúrok treba kúpiť. Zjavne to musí byť maximum z najväčšej hodnoty a_v a b_e v tomto komponente. Treba dokázať, že ak nakúpime všetky figúrky v jednom vrchole, vieme ich postupne presúvať po tomto komponente a všetko obsadiť. A preto nakúpime v najlacnejšom vrchole.

(b) Dobrý pozor na to, čo je formálny dôkaz. Nestačí len napísať, že zjavne sa to tak oplatí. Treba ukázať, že ak máme riešenie, ktoré tú hranu obsadí iným spôsobom, vieme toto riešenie upraviť na také, ktoré ju obsadí z jednej strany.

(c) Keď sa rozhodujem, či hranu obsadiť alebo nie, viem jednoducho porovnať tieto dve možnosti využijúc (a). Ak hranu obsadím, dostanem väčší komponent, kde záleží len na maximálnom a_v a najmenšom c_v v tomto komponente (najdrahšia hrana musí byť kvôli poradiu pridávania tá aktuálna). A keď hranu neobsadím, je to len súčet toho, koľko ma stojí obsadiť jeden a druhý komponent.

To dôležité uvedomenie však je, že bez ohľadu na to, či túto hranu pridám alebo nie, tieto dva komponenty si môžem spojiť do jedného a pracovať ďalej len s ním. Ak sa totiž niekedy neskôr rozhodnem obsadiť hranu, ktorá vychádza z tohto komponentu, tá bude určite drahšia ako všetky hrany, ktoré potenciálne tvoria tento komponent (lebo idem v poradí podľa hodnoty b_v). Takže ak obsadím neskôr hranu vedúcu z tohto komponentu, nutne viem obsadiť aj vnútorné hrany tak, aby som vytvoril kostru komponentu.

4. Tá na doplnenie do 80 (5 bodov)

(a) Rozstrihnime kruh na pásik dĺžky n. Ak sa v tomto pásiku nachádza P, úlohu sme vyriešili. Problém ale nastane, ak rozstrihne kruh práve vnútri výskytu P, potom ho totiž algoritmus KMP neobjaví. Mohli by sme skúsiť každé možné rozstrihnutie, to je však pomalé. Elegantné riešenie je kruh rozstrihnúť, vzniknutý pás skopírovať, tieto dve kópie zlepiť za seba a na takomto reťazci dĺžky $2n$ pustiť hľadanie P.

(b) Ak sa každá hrana zdvojnásobí, zdvojnásobí sa aj dĺžka ľubovoľnej cesty. Najkratšia cesta teda ostane najkratšou cestou.