## 1. Tá o tuneloch (25 bodov)

(a) Riešenie je veľmi podobné riešeniu úlohy 5 z cvičení 1. Graf zo vstupu si zduplikujeme, jedna kópia vrcholu $(v, 0)$ slúži pre sledy, ktoré ešte neprešli cez tunel, druhá kópia $(v, 1)$ pre sledy, ktoré už cez tunel prešli. Hrany na úrovniach ostanú rovnaké, navyše však každý tunel z $v$ do $u$ vytvorí hranu $(v, 0) \to (u, 1)$. Následne už len hľadáme najkratšiu cestu z $(a, 0)$ do $(b, 1)$ pomocou Dijkstrovho algoritmu.

(b) K predchádzajúcemu riešeniu pripojíme trik z domácej úlohy, kde vzdialenosť vrcholu bude dvojica vzdialenosti a počtu prejdených tunelov.

(c) Jeden tunel v grafe, ktorého jeden koniec má stupeň 1. Po prechode týmto tunelom sa môžeme iba vrátiť späť.

(d) Najjednoduchšie je využiť redukciu Najdlhšej cesty na náš problém. Stačí, aby každá hrana bol tunel.

## 2. Tá označujúca cesty (25 bodov)

Odstráňme zo stromu koreň, čím sa nám strom rozpadne na niekoľko podstromov. Uvedomme si, že pre každý podstrom je riešenie buď 0 alebo 1 podľa toho, či sa v danom podstrome nachádza označený list. Vždy totiž vieme označiť hranu vedúcu do pôvodného koreňa, čím minimalizujeme počet označených hrán.

Každý podstrom riešime samostatne.

Ak sú v podstrome dva označené listy, chceme vybrať jednu hranu. Musí platiť, že vrchol, z ktorého táto hrana vychádza dohora je spoločný predok oboch označených listov. A aby sme minimalizovali počet neoznačených vrcholov, ktoré sú pokryté touto hranou, chceme vybrať čo najnižší takýto vrchol. Odpoveďou je teda zjavne najnižší spoločný predok (LCA) týchto dvoch vrcholov, čo vieme riešiť efektívne.

Čo ak je však označených listov v podstrome viac. Hľadáme teda jedno spoločné LCA pre všetky tieto listy. To vieme robiť neefektívne po dvojiciach, alebo si uvedomíme, že toto spoločné LCA všetkých listov je v skutočnosti LCA najľavejšieho a najpravejšieho označeného listu v strome (očíslované podľa DFS).

Teraz je už riešenie jednoduché. V dynamickej množine (napr. set) si pamätáme zoznam všetkých označených listov v podstrome, indexovaných podľa DFS poradia. Pri pridaní alebo odstránení označenia iba upravíme túto množinu a vyberieme z nej najľavejší a najpravejší prvok a vypočítame LCA pre tieto dva listy. Celková zložitosť je preto $O(\log n)$ na každú otázku.

## 3. Tá o chobotnici (25 bodov)

Existuje pomerne priamočiare dynamické programovanie so zložitosťou $O(rs)$, ktoré sa dá popísať:

$$dp(x, y) = \max(dp(x + 1, y), dp(x, y + 1)) + r_{(x,y)}$$

kde $r_{(x,y)}$ je počet rýb na políčku $(x, y)$. Jednoducho sa pozrieme, či sa viac oplatí prísť na toto políčko zhora alebo zprava, iné možnosti totiž nie sú.

Toto riešenie má ale problém, že zbytočne prechádzame celú mriežku, aj keď je v nej iba malý počet rýb. Políčka s rybami si teda usporiadame podľa $y$-ovej (a sekundárne $x$-ovej) súradnice v poradí, v akom by ich spracovávalo naše dynamické programovanie.

Teraz nám stačí spracovávať políčka s rybami (keďže na ostatných sa nič nezmení). A pre políčko $(x, y)$ nás zaujíma, z ktorého iného políčka s rybami sme sem mohli prísť. Pričom kandidáti sú iba políčka, ktoré boli spracované skôr, lebo majú väčšiu súradnicu $y$. Z týchto políčok však nemôžeme uvažovať všetky, iba tie, čo sú napravo od $x$.

No a to nás navádza na použitie intervalového stromu. Pre každý stĺpec si pamätáme koľko najviac rýb sme zatiaľ vedeli zjesť tak, aby sme skončili v tomto stĺpci. Pri spracovaní políčka $(x, y)$ sa následne spýtame na najväčšiu hodnotu na intervale $< x, s >$, z tohto stĺpca sa totiž určite vieme presunúť na aktuálnu pozíciu. Novovypočítanou hodnotou si upravíme intervalový strom na pozícii $x$.

Posledný krok je už len pridanie kompresie súradníc, keď si uvedomíme, že mať ryby v stĺpci 1 a $10^9$ je to isté ako ich mať v stĺpcoch 1 a 2.

## 4. Tá na doplnenie do 80 (5 bodov)

(a) Nie je to pravda. Stačí nakresliť nejaký malý protipríklad.

(b) Je to pravda. Problém $A$ je totiž sám osobe polynomiálny (pozriem každý možný začiatok a koniec podúseku). To znamená, že naša polynomiálna redukcia ho môže najprv vyriešiť v polynomiálnom čase a následne vrátiť jednoduchý vstup pre problém $SAT$ – riešiteľný ak podúsek s daným súčtom existuje, neriešiteľný ak taký podúsek neexistuje.