Cau, napadlo me, ze bude mozna fajn, resit ty BigOnes priklady dohromady a ke kazdemu sem neco strucne napsat. Jak by se to asi dalo resit, v jakem jazyce atp.

Bude asi fajn, kdyz se kazdy pod svuj napad podepise (ale nemusi), abychom se pak mohli treba ptat nebo tak…

uzávorkování výrazu - na vstupu zadány jen konstanty true/false a operátory and, not, or - jak, to si zvolíme sami. našim prvním úkolem bylo zjistit, jestli je možné toto nějak uzávorkovat, aby to byl korektní výraz (tedy např "not true and" nepůjde nikdy, ale "true and not false" už ano). priorita ani asociativita operátorů nebyla zadaná. jestliže to lze uzávorkovat, určit, jestli to lze uzávorkovat tak, že celková hodnota toho výrazu je true, pokud ano, pak nějaké (jedno) uzávorkování vydat. samozřejmě byla žádoucí i aplikace nějaké heuristiky.

Reseni (Prolog - Lukáš)
Pouziji podobny trik jako u male ulohy s vyrazem (http://pastebin.com/Wv98rg79). Prevedu si vyraz na postfix. Zavedu si predikat proved, ktery jako argument vezme dva zasobniky (pocitaci a vstup) a bud prevede neco ze vstupu na “pocitaci” zasobnik nebo provede operaci na “pocitacim” zasobniku. Tudiz musim mit zadefinovane i vsechny operace, ktere z “pocitaciho” zasobniku neco seberou a pridaji vysledek. Pokud je vstup prazdny a na pocitacim zasobnik je jedna polozka, tak mame vysledek (ten muzeme “vratit zpet nahoru”) Stejne tak si v prubehu muzeme pamatovat uzavorkovani.
Pokud chceme zjistit, koreknost uzavorkovani zavolame proved([], Vstup, _, _)
Pokud chceme zjistit, jestli vysledek je true zavolame proved([], Vstup, true, Vyraz), kde Vyraz bude obsahovat spravne uzavorkovani, tak abychom dostali opet true.
Poznamka: nejvetsi peklo bude asi se stringy, ale to snad nebudou na ustnim tak hrotit. Heuristika, budu se snazil vyhodnocovat “pocitaci” zasobnik drive nez ho plnit. Tak se mi muze stat, ze uzavorkovani nebude validni drive nez dojdu na konec.

Ohodnocení vrcholů a hran orientovaného grafu -- Máte orientovaný graf a ke každé hraně máte seznam dvojic reálných čísel. Cílem je nějak ohodnotit vrcholy libovolnými čísly a hrany jednou dvojící ze seznamu tak, aby odchylka byla co nejmenší. Odchylka je součet odchylek všech hran. Odchylka hrany s hodnotou počátečního vrcholu v_1, hodnotou koncového vrcholu v_2 a ohodnocením hrany (e_1, e_2) je |v_1 - e_1| + |v_2 - e_2|. Máte zvolit vhodnou heuristiku.

Řešení (Štěpán)
Pozorování: Když určíme ohodnocení hran, a chceme určit ohodnocní vrcholu, vyplatí se vrchol v ohodnotit jako aritmetický průměr těch e_vi. To dá nejmenší odchylku a je to snadné.
Zároveň taky nemá cenu řešit vrcholy dříve, než se vyberou ohodnocení hran IMHO, takže je důležité akorát vybrat správné páry na hranách, a pak doprůměrovat vrcholy.
Řešil bych vybráním náhodné, řekněme M/5 velké množiny hran, pro které bych vyzkoušel všechna ohodonocení, aby to někdy skončilo. Když máme určené nějaké hrany, můžeme určit hladovým způsobem zbytek. Něco jako BFS: máme seznam určených hran, vezmeme

jeden konec hrany, podíváme se na všechny ostatní hrany, které zatím nejsou určené a vybereme ty dvojice, které mají na tomto konci co nejbližší číslo jako to, co už bylo zvolené. Nové hrany přihodíme do fronty. Tohle opakujeme dokud neexistuje neohodnocená hrana. Potom přiřadíme artim. průměr vrcholům a spočítáme odchylku tohohle grafu, s tím, že si zapamtujeme ten nejlepší graf, a zkusíme náhodně vybranou konfiguraci na začátku změnit. Asi v Haskellu, vygenerovat všecha odhodnocení není tak těžké pomocí list comprehension. BFS s frontou je taky lehké, na hledání neohodnocených hran bych si udržoval list ohodnocených a list neohodnocených a lookup by měl vyřešit problém.

Slovnik a edit distance (Petr H ?- kdyz to delal jako zapoctak)
měli jsme najít ve slovníku se vzorky slov slovo, které nejvíce odpovídá zadanému vzorku. Vzdálenost slov se určovala pomocí metriky, kterou nám zadali, je to tu již někde popsáno v jiném vlákně.
Reseni (Petr):
Editační vzdálenost se dělá klasicky dynamikou. Uděláme si tedy fci, která spočítá vzdálenost nějakých dvou stirngů (viz: https://wiki.haskell.org/Edit_distance). Pro nalezení nejbližšího slova projdeme ten slovník lineárně na vrátíme minimum.

Co se týče fce na vzdálenost dvou stirngů, tak klidně popíšu blíž, ale nějak pochybuju, že bych to popsal líp než odkaz nahoře.

Planovani vyroby pro stroje
Je dany pocet strojov n, a zoznam vyrobkov (kazdy vyrobok ma dane cislo, dobu kolko trva jeho vyrobenie, cas odkedy je mozne ho vyrobit, a cas dokedy musi jeho vyroba skoncit). Mozeme predpokladat, ze doba vyrobenia sa vojde do tohoto intervalu od-do (teda kazdy vyrobok je vyrobitelny). Stroje su navzajom zamenitelne, vyrobky tiez. Kazdy stroj moze samozrejme naraz vyrabat len jeden vyrobok.

Ulohou bolo vratit rozvrh vyroby (teda zoznam zaznamov typu [cislo vyrobku, cas zacatia vyroby, cislo stroja]), tak aby sa maximalizoval pocet vyrobenych vyrobkov. Nemusia sa dat vyrobit vsetky. Malo sa to riesit s heuristikou. Dvorak nam poradil, aby sme to pisali zhora, teda pomocne funkcie az nakoniec, pre pripad, ze by sme to nestihli.

Reseni (Lukas):
Pozorovani: kazdy vyrobek ma stejnou cenu, tedy se vyplati zacit vyrabet v poradi od nejkratsi vyroby po nejdelsi.
Jednoducha heuristika: Udelam neco jako BFS, ktere naplanuje pro stroje co budou delat. Pro kazdy stroj naplanuji prvni vyrobne nejkratsi vyrobek, ktery se v tu dobu muze vyrabet. Nemusi nutne vratit optimalni reseni. Za tuto heuristiku nekdo na foru dostal za dve.

Truhly
Máte N truhel, N+1 klíčů, v každé truhle je právě jeden klíč, jeden máte na začátku.

Každý klíč má svou barvu a každá truhla má barvu, truhlu otevřete jen klíčem té samé barvy. Zároveň když použijete klíč, už ho nemůžete použít znovu (zůstane v zámku)? Existuje posloupnost otvírání truhel tak, že otevřete všechny truhly? Pozn. máte to udělat polynomiálně

Řešení (Lukáš, Štěpán - Prolog/Haskell ono je to vlastne jedno)

Situace je vlastně orientovaný graf, protože když ve žluté truhle je modrý klíč, tak po otevření žluté můžeme otevřít i modrou, atd. Nás tedy zajímá, zda existuje počáteční vrchol takový, že když ho naším univerzálním N+1-ním klíčem otevřeme, tak se dostaneme všude. To nemusí, protože pokud jako první otevřeme modrou, může se nám stát, že cestou potkáme modrý klíč dříve, než v poslední truhle a tak už nemůžeme dále pokračovat. Můžeme tedy napsat DFS takové, že se zastaví právě tehdy když budou navštívené všechny vrcholy a pustíme ho na všechny počáteční truhly (což jsou asi všechny). Pokud někdy doběhne správně, tak víme, že otevřením téhle truhly otevřeme všechny. *Podivně lehké*.

Hypergraf H = (V,E), kde V je množina vrcholů a E je libovolná podmnožina vrcholů (ne nutně dvouprvková). Lineární hypergraf je pak hypergraf, jehož žádné 2 hrany se neprotínají ve víc než jedno bodě. Erdösova hypotéza říká, že každý hypergraf se dá hranově dobře (stejně obarvené hrany mají prázdný průnik) obarvit pomocí nejvýše |V| barev. Napište predikát/funkci, který(á) generuje všechny lineární hypergrafy a otestujte pro každý, zda ho lze obarvit maximálně n barvami. U obecných hypergrafů je hledání takového obarvení NP-Úplný problém, ale pro lineární se předpokládá (nikdo Erdösovu hypotézu za 40 let nevyvrátil), že takové obarvení existuje. Použijte proto pro hledání toho obarvení nějakou heuristiku.

Řešení(Lukas+Štěpán - Haskell/Prolog):

Generováni: můžeme generovat například pro pevně zadané N. Tedy generování hypergrafu vlastně spočíná v generování *množiny* podmnožin množiny vrcholů. Takze budu mit funkci subsets, ktera bude generovat vsechny podmnožiny. Spustím na seznam [0..N] mam seznam podmnožin vrcholů P(V) a na tu to spustím znovu a dostanu seznam všech možných E *hypergrafu* na N vrcholech. (Možná by se už generování dalo omezit linearitou). Teď stačí seznam profiltrovat, tak aby platila podmínky pro lineární hypergraf. Kontrola podmínky pro linearitu třeba tak, že pro zadaný hypergraf vygeneruju všechny dvojice hran(list comprehension), a pro každou z nich musí platit, že pro všechny elementy první hrany je maximálně jeden elementem i té druhé.

Barvení: To už by mohlo být relativně jednoduché. Mohu backtrackingem zkoušet barvit hrany (v pořadí sestupně podle počtu vrcholů, které hrana obsahuje). Na foru psal, že barvy střídal, což dává smysl, protože když budu barvit a dojdu na barvení vrcholu, který má nějaké (již obarvené) sousedy, tak chci, abych ti sousedi měli pokud možno jinou barvu.