Andrej Skok, 7h => napriek tomu že mi to dlho trvalo sa mi zdá obtiažnosť v poriadku

1.
Zovšeobecniť prípad pre k=1? Už tam veľa vecí na zovšeobecnenie nezostalo.
Po veľa neúspešných pokusoch sa ukázalo ako celkom efektívne zamyslieť sa nad tým čo vlastne robím pre k=1 => upravujem kôpky tak aby pri binárnom zápise a číslach napísaných pod sebou zostal v každom stĺpci súčet mod 2 = 0, súper potom nevie dosiahnuť túto vec, ale nech spraví hocičo tak ja mu to znova zrovnám. Tak keď pre k=1 modulujem 2 tak co keď by som pre všeobecné k moduloval k+1. Prehrávajúce pozície boli tie ==0 v súčte po stĺpcoch modulo k+1.
Vyzerá že to sedí, ktoré kôpky však vybrať? (stĺpce = bity na rovnakých miestach pri binárnom zápise, riadky = kôpky)
Beriem zľava stĺpce ktoré nemajú súčet 0 mod k+1 a beriem z nich vždy toľko riadkov aby, keď v nich flipnem bity tak dostanem v stĺpci súčet 0 mod k+1 ciže beriem ich toľko koľko je súčest stĺpca mod k+1. Takto prechádzam stĺpce (vždy selektujem také riadky ktoré som už neselektol predtým) až dokým buď niesom na konci a tým pádom zoberiem toľko kôpok, koľko som nasčitaval zo všetkých zvyškov dokopy, alebo som nasčítaval v niektorom okamihu >= k. Ak presne k tak to nieje problém a viem poflipovať bity vo vybraných čislach tak, aby sa nasčitavali stĺpce na 0 mod k+1. Ak je to >k tak z posledného stĺpca ktorý som pridával vyberiem len riadky tak, aby vybraných kôpok bolo dokopy k. Teraz z každej selektnutej kôpky nastavím most significant bit na 0, to mi zaruči že vsetky vybrané kôpky zmenším a zároveň v každom stĺpci ktorý som týmto nevynuloval mám dostatočne veľa selektnutých riadkov ktoré môžem ľubovolne flipovať tak aby mi súčty v stĺpcoch sadli na 0. T.j. ukázal som že z vyhrávajúcej pozície viem isť do prehrávajúcej.
Z prehrávajúcej pozície sa súper nevie dostať lebo musí upraviť aspoň jednu kôpku a najviac k čiže nemôže spôsobiť že všetky stĺpce zostanú ==0 mod k+1.
Posledná vlastnosť P a V pozícii, t.j. jediná terminálna pozícia (0,0,0,0…) je prehrávajúca je triviálne po stĺpcoch 0 mod k+1.

=> pozícia je vyhrávajúca ak súčet po stĺpcoch mod k + 1 nie je rovný 0.

3.
Kombinatorická úloha. Riešime cez FFT.
Máme 3 polynómy ktoré treba vynásobiť:
P1 = polynóm ktorý má ako koeficienty 1 pri x^(prvočíslo < 1.000.000), 0 inak
P2 = detto
Vynásobíme cez FFT, koeficient pri x^k je počet spôsobov ako vyskladať k ako súčet 2 prvočísel.
Teraz nám už stačí iba pre každé k = 0...1.000.000 zistiť S(k) a zosumovať ich štvorce.
S(k) získam tak že preiterujem všetky primes so substringom 47 menšími ako 1.000.000 a pripočítavam počet možností ako vyskladať číslo zmenšené o dané prime (to zistím z koeficientu môjho vynásobeného polynómu) + násobím 3 lebo viem toto prime vsunúť na prvé, druhé alebo tretie miesto v tom finálnom súčte 3 prvočisel.
Kód v prílohe. Použil som classu na násobenie polynómov cez FFT odtiaľto: http://www.davideisenstat.com/simplertimes/Polynomial.java
Výsledok: 1257047085573463080141

4.
Postupoval by som takto:
Budem kódovať polynómy do veľkých čisiel. Zvolil by som nasledové kódovanie:
Napr.

37X^2 + 20x + 115 => 00037 00020 00115
('kódovanie', je to to isté ako keby som polynóm vyčíslil pre x=10^dostatočné čislo a potom umocnil, resp násobil a potom z toho znova spravil polynóm v premennej x čo je jednoduché keďže moje x je mocnina 10, sú to iba posuny)
Ako viem koľko znakov treba nechať navyše ako padding?
Ak mám polynómy p a q, tak najhoršie co sa mi môže stať je že sa najväčší koeficient vynásobí so všetkými koeficientami druhého polynómu a to sa nasčítava t.j. max koeficient polynómu ktorý by vznikol ich vynásobením by bol cca (max stupeň(p,q)) * max_koef(p) * max_koef(q) // slightly overestimated. => každé čislo teda 'nafúknem' na dĺžku maxima ktoré som práve odhadol aby sme mali po vynásobení dosť miesta na zápis a nepretekalo nám to do ďalších koeficientov.
Po vynásobení dostanem koeficienty polynómu tak že robím dané čislo mod 10^k*x kde x je počet cifier spomínaneho maxima a k je k-ty koeficient.

Keď už to mám použijem zvyšok predchádzajúceho programu. To sú už len for-cykly, zopár polí a scitavanie a násobenie.

5.
Jedná sa o orientovaný graf v ktorom elixíry sú vrcholy a suroviny sú orientované hrany s cenami.
Cieľom je predspracovať graf tak, aby sme vedeli efektívne zostrojiť cestu od jedného elixíru k druhému.
Pokiaľ by nám stačilo poznať iba ceny, tak by bolo treba iba umocniť maticu (i,j pozícia je cena hrany z i do j, alebo inf ak neexistuje) na #vrcholov (=maximálna dĺžka cesty). Toto sa dá robiť v čase násobenia matíc => tranzitívny uzáver s použitím + namiesto konjunkcie a min namiesto disjunkcie. Po tejto operácií máme na súradnici i,j cenu najlacnejšej cesty, prípadne inf ak neexistuje.
Riešením by mohlo byť udržiavať si druhú maticu kde by bolo na (i,j) poznačené ktorá hrana vychádzajúca z i je prvá na najlacnejšej ceste z i do j. Tým pádom by sme na každú query vedeli odpovedať v lineárnom čase od počtu elixírov. Keďže by stačilo nasledovať hrany v našej pomocnej matici a cesta môže mať maximálne #elixírov vrcholov.
Zostrojenie tejto pomocnej matice:
Na začiatku má matica poznačenú hranu z i do j na pozícií (i,j) ak sú i a j spojené hranou, inak null.
Potom stačí že vždy keď robíme operáciu min(X,Y) tak vyberáme vlastne z cien dvoch ciest, nejakej cesty X a cesty Y, aktualizujeme aj hranu na danom indexe na takú ktorou začína buď cesta X alebo Y.

Predspracovanie čas násobenia matíc, potom pri každej query zostrojenie cesty v čase O(#elixírov).

7.
Zostrojíme si maticu domčekového grafu.
Tá má tvar A = {
    {0,1,1,0,0,0}
    {1,0,1,1,1,0}
    {1,1,0,1,0,1}
    {0,1,1,0,1,1}
    {0,1,0,1,0,1}
    {0,0,1,1,1,0}
}

Umocnením tejto matice na ^k dostaneme na súradnici (i, j) počet sledov z vrchola i do vrchola j dĺžky k. Tým pádom všetky sledy dĺžky n dostaneme ako prvú súradnicu vektora v = {1,0,0,0,0,0} * A^n (resp. súradnica 0,0 v matici A^n, to sú sledy dĺžky n z vrchola 0 do vrchola 0)

Toto vieme spraviť v čase logn*M(6) kde M(k) je čas násobenia matíc KxK.
Rýchlosť rastu odhadneme pomocou vlastných čísel matice A (pomôže wolfram). Najväčšie vlastné čislo je rovné 3.4679… Vlastný vektor prisluchajúci tomuto vlastnému číslu má s vektorom {1,0…} nenulový skalárny súčin (niesú na seba kolmé). Tým pádom počet sledov dĺžky n na danom grafe je rádovo 3.46..^n.
(rýchla skúška správnosti, umocníme maticu na n = 100 a n = 101 spravíme podiel čísel na súradnici (0,0))