LATVIJAS 22. INFORMĀTIKAS OLIMPIĀDES I POSMA UZDEVUMU ĪSI ATRISINĀJUMI

1. "KĀRŠU SPĒLE"

Katru kāršu trijnieku apstrādājam atsevišķi. Apzīmēsim uz pirmā, otrā un trešā spēlētāja kārtīm uzrakstītos skaitļus ar attiecīgi ar *a*, *b*, *c*.

1) Ja visi trīs skaitļi ir vienādi, punktu neiegūst neviens;
2) Ja visi trīs skaitļi nav vienādi, tad
   a. ja $a = b$, tad punktu iegūst trešais spēlētājs;
   b. ja $a = c$, tad punktu iegūst otrais spēlētājs;
   c. ja $b = c$, tad punktu iegūst pirmais spēlētājs.
   d. ja visi skaitļi ir dažādi, tad punktu iegūst tas spēlētājs, uz kura kārts uzrakstīts lielākais no skaitļiem.

2. "SERIĀLS DISKOS"

Ieviesīsim dažus apzīmējumus:

- X – viena diska izmērs sekundēs;
- D – pašreiz izmantoto disku skaits, ieskaitot pēdējo disku, kurā kaut kas ir ierakstīts;
- V – sekunžu skaits, ko vēl var ierakstīt D-tajā diskā ($0 \le V < X$);
- G – pašreiz ierakstāmās sērijas garums sekundēs;

Mēģināsim ierakstīt kārtējo sēriju, kuras garums ir G sekundes. Pieņemsim, ka sērijai nepieciešami A pilni diski un vēl B sekundes. Ja G nedalās ar X, tad ir spēkā A = G div X un B = G mod X. Ja turpretī sērija ir precīzi ierakstāma G / X diskos, t.i., G dalās ar X bez atlikuma, tad uzskatīsim, ka tai nepieciešams (G / X) − 1 disks un vēl X sekundes. Tā mēs nodrošinām, ka B > 0.

Ja V < B, tad tas nozīmē, ka aizsāktajā diskā mēs nedrīkstam rakstīt aplūkojamo sēriju, citādi tā aizņems vairāk disku nekā tai nepieciešams. Tātad būs nepieciešams A + 1 jauns disks, un pēdējā no tiem paliks X − B brīvas sekundes.

Ja turpretī $V \ge B$, tad tas nozīmē, ka aizsākto disku varam pierakstīt līdz galam (pirmās V sērijas sekundes), atlikušo daļu ierakstot jaunos diskos (būs nepieciešami A jauni diski). Pēdējā diskā paliks V − B brīvas sekundes.

Sākumā inicializējam D = 1, V = X. Tas pārkāpj nosacījumu, ka V < X, taču tas ir tikai uz brīdi, jo pēc pirmās sērijas apstrādes un arī turpmāk būs spēkā V < X. D vienmēr būs vienāds ar mazāko disku skaitu, kādā var ierakstīt visas līdz šim apstrādātās sērijas, tāpēc beigās jāizvada D vērtība.

3. "BANKAS KONTI"

Dotam N jāatrod pēc iespējas tuvāks skaitlis $S \ge N$, ka $S = 2^x 3^y$ kādiem nenegatīviem veseliem *x*, *y* (t.i., vienīgie skaitļa S pirmreizinātāji ir 2 un 3). Skaidrs, ka $S \le 3 * N \le 3 * 10^{18}$, tātad, S ietilps 64 bitu mainīgajā.

Cik ir skaitļu, kuru vienīgie pirmreizinātāji ir 2 un 3, kas nepārsniedz $3 * 10^{18}$? Ievērosim, ka $2^{64} > 3 * 10^{18}$ un $3^{38} > 3 * 10^{18}$, tāpēc, ja $2^x 3^y \le 3 * 10^{18}$, tad $x < 64$ un $y < 38$. Tātad šādu skaitļu nav vairāk par 64 * 38 = 2432. (faktiski šo skaitļu ir vēl mazāk, jo pietiekoši lieliem *x* un *y* skaitlis $2^x 3^y$ pārsniegs $3 * 10^{18}$). Tāpēc risinājumā varam pārlasīt visus šos skaitļus un atrast vistuvāko derīgo.

4. "NODZĒSTAIS SKAITLIS"

Ievērojam, ka, saskaitot visus deviņus skaitļus stabiņā, neveicot pārnesi, iegūsim N-ciparu „skaitli" [45] [45] ... [45] [45], jo 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 = 45. Izpildot pārneses, iegūsim (N+1)-ciparu skaitli 499...995. Visu deviņu skaitļu summa, tātad, mums ir zināma. Zinot astoņus no šiem skaitļiem, varam atrast nodzēsto.

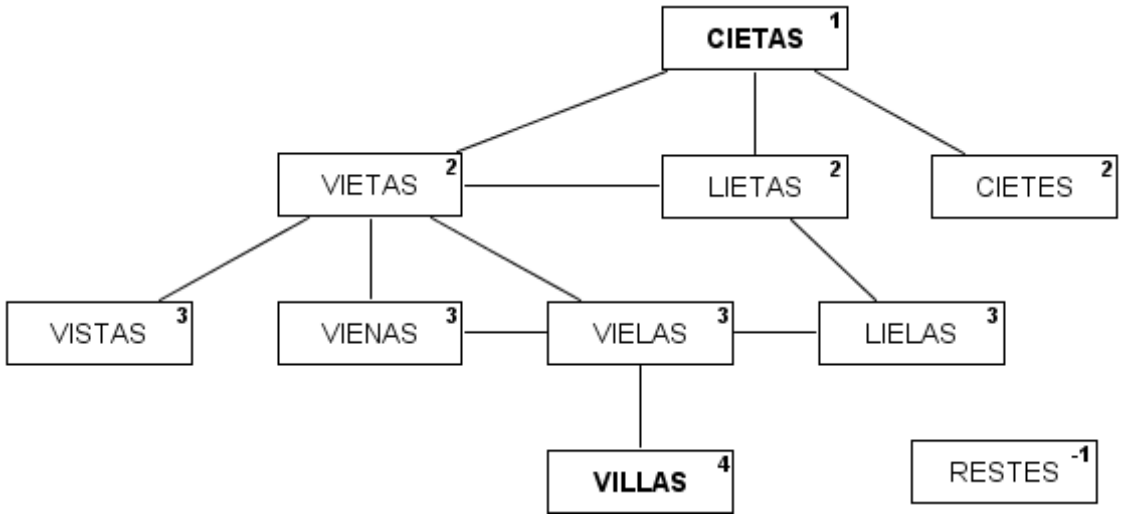
Piezīme: (N+1)-ciparu skaitlis $499...995 = 5 * (10^N - 1)$.

5. “VĀRDU SARAKSTS”

Teiksim, ka vārdu A var pārveidot par vārdu B, ja vārdiem A un B ir tieši viens atšķirīgs burts. Ievērosim, ka ja vārdu A var pārveidot par vārdu B, tad arī vārdu B var pārveidot par vārdu A.

Sākumā katriem diviem vārdiem noskaidro, vai ir iespējams vienu vārdu pārveidot par otru un izveido atbilstošu grafu (katrs dotais vārds atbilst vienai grafa virsotnei). Turpmāk vārdu saraksts vairs nav nepieciešams. Pēc tam izveidotajā grafā veic meklēšanu plašumā (*Breadth First Search* jeb *BFS*) no virsotnes, kas atbilst pirmajam vārdam, uz virsotni, kas atbilst beidzamajam vārdam. Ar šī algoritma palīdzību tiks atrasts īsākais ceļš starp norādīto starta un beigu virsotni. Tā kā īsākais ceļš grafā atbilst īsākajai virknei no pirmā vārda līdz norādītajam virknes beidzamajam vārdam, tad īsākā ceļa garums grafā sakrīt ar īsākās virknes garumu vārdu sarakstā.

Aplūkosim piemēru, kurā no vārda CIETAS jāiegūst vārds VILLAS. Attēlā katram vārdam pierakstīts īsākā ceļa garums. Faktiski tas sakrīt ar īsākās virknes, kas sākas ar vārdu CIETAS, un beidzas ar attiecīgo vārdu, garumu.



6. “ŠAUTRIŅAS”

Ja mēs izvēlamies izdarīt mazāk par 4 metieniem, tad varam uzskatīt, ka pārējos metienus tomēr esam izdarījuši, iegūstot tajos 0 punktus. Tāpēc varam pievienot skaitli 0 kā vienu no trāpījuma iespējām un vienmēr izdarīt tieši 4 metienus. Tad vienkāršs, bet lēns risinājums ir pārbaudīt visus $(N+1)^4$ variantus un atrast tādu, ka iegūtā punktu summa ir pēc iespējas tuvāka vajadzīgajai.

Aplūkosim efektīvāku risinājumu. Aprēķinām visas iespējamās summas, ko var iegūt, izdarot tieši divus metienus (saglabājot iespēju iegūt 0 punktus par metienu), un sakārtojam tās augošā secībā (šo summu skaits nepārsniedz $(N+1)^2$), izmetot visas summas, kas pārsniedz M. Saglabājam tās masīvā P[1..S], kur S – dažādo summu skaits. Ievērojam, ka P[1] = 0 un P[S] ≤ M. Katram indeksam i varam atrast „labāko” vietu masīvā j – tādu, ka P[i] + P[j] ≤ M, bet P[i] + P[j+1] > M (vai arī j = S). Tieši P[i] + P[j] vērtība ir tuvākās summas kandidāts. Ievērojam, ka, palielinot i vērtību, j vērtība nevar palielināties. Tāpēc varam inicializēt i := 1, j := S un ciklā palielināt i vērtību, attiecīgi samazinot j vērtību, līdz tās „satiekas”, t.i., līdz i ≥ j. Pseidokods:

```
i := 1; j := S; tuvākā_summa := P[i] + P[j];
while (i < j) do
   i := i + 1;
   while (P[i] + P[j] > M) do j := j − 1;
   tuvākā_summa := max(tuvākā_summa, P[i] + P[j]);
output tuvākā_summa;
```