OLIMPIADA NAȚIONALĂ DE INFORMATICĂ, ETAPA NAȚIONALĂ, CLASA A V-A, DESCRIEREA SOLUȚIILOR

COMISIA ȘTIINȚIFICĂ

Problema 1: Robinhood

Propusă de: prof. Șerban Marinel, Colegiul Național "Emil Racoviță", Iași

Descrierea unor soluţii posibile

Soluția 1.

Propusă de: prof. Șerban Marinel, Colegiul Național "Emil Racoviță", Iași

După citirea datelor se inițializează un vector *tinta*[] cu valoarea 0, indicând faptul că nici o țintă nu a fost atinsă. Se inițializează cu 0 alte două variabile *atinse* și *secunde*, care reține câte ținte au fost atinse și, respectiv cât timp a trecut.

Două variabile $tinta\_robin = 1$ (Robin Hood pleacă de la ținta 1) și $tinta\_john = n$ (Little John de la ținta n) reține numărul țintei în dreptul căreia se află la un moment dat fiecare dintre cei doi arcași. Alte două variabile $dir\_robin = true$ și $dir\_john = false$ indică direcțiile celor doi arcași.

Într-o structură repetitivă se simulează apoi concursul până când toate țintele au fost atinse (variabila *atinse* devine egală cu n).

Cerința 1: După terminarea simulării concursului, dacă cerința este 1 se afișează valoarea variabilei *secunde*.

Cerința 2: Dacă cerința este 2, se parcurge vectorul *tinta*[] contorizându-se valorile 1, apoi se afișează contorul.

Cerința 3: Dacă cerința este 3 se determină maximul elementelor vectorului *tinta*[], care se afișează, apoi, parcurgând din nou vectorul se afișează indicii elementelor egale cu maximul determinat.

Soluția 2.

Propusă de: prof. Rotar Dorin-Mircea, Colegiul Național "Samuil Vulcan", Beiuș

Cei doi jucători au un traseu de urmat bine stabilit. Robin Hood parcurge pe rând de la prima țintă până la ultima și înapoi până la prima țintă un număr de 2*n-1 ținte. Observăm că traseul astfel parcurs se repetă. Am reținut într-un vector **A** pozițiile celor 2*n-1 ținte descrise anterior, iar într-un vector **B** se reține similar pozițiile celor 2*n-1 ținte parcurse de Little John. Se simulează jocul pas cu pas și se reține într-o mulțime toate țintele atinse de cei doi. Jocul se termină în momentul în care mulțimea conține n elemente. Totodată se folosește un vector de frecvență pentru a reține de câte ori a fost atinsă o țintă.

Cerința 1: Pentru prima cerință se afișează numărul de parcurgeri rezultat în urma simulării jocului.

Cerința 2: La cerința a doua putem afla din vectorul de frecvență țintele ce au fost atinse o singură dată.

Cerința 3: Cerința a treia se rezolvă aflând din vectorul de apariții frecvența maximă. Cunoscând această valoarea putem afișa țintele cu numărul maxim de loviri printr-o simplă parcurgere a vectorului de frecvență.

Soluția 3.

Propusă de: prof. Panete Adrian, Colegiul Național "A. T. Laurian", Botoșani

Să urmărim mișcarea lui Robin Hood fără să ținem cont de momentele în care trage.
Pentru asta vom numerota țintele 1, 2, ..., n cu 0, 1, 2, ..., n-1.
Robin Hood se găsește la secunda 0 în dreptul țintei 0.
În primele $n-1$ secunde Robin Hood se deplasează spre dreapta și ajunge la ținta $n-1$.
În următoarele $n-1$ secunde el se deplasează spre stânga și ajunge înapoi la ținta 0.
Se poate spune că după exact m = 2*n-2 secunde Robin Hood se află în situația inițială și că din acel moment el va relua același traseu la fiecare m secunde.
Folosind acest raționament putem să spunem exact în ce poziție se găsește Robin Hood la orice moment de timp t >= 0 astfel:

- fie r = t % m;
- dacă r < n-1 atunci el se află în dreptul țintei r și se deplasează spre dreapta;
- dacă r >= n-1 atunci el se află în dreptul țintei m-r și se deplasează spre stânga.

Deoarece Robin Hood trage odată la p secunde, pe noi ne vor interesa doar timpii de forma T[k] = p*k. Notăm R[k] = T[k] % m și folosind R[k] putem determina exact poziția unde se va afla Robin Hood dar și direcția (stânga sau dreapta în care se deplasează).

Deoarece valorile R[k] au doar m valori posibile este evident că printre momentele în care Robin Hood trage va trebui să găsim două momente de timp în care acesta să fie în dreptul aceleiași ținte și îndreptat în același sens (stânga sau dreapta). Ne-ar interesa care este diferența în număr de secunde între două astfel de momente consecutive. Să considerăm T[k] și T[k+d] două astfel de momente și Per = T[k+d] – T[k] diferența de timp căutată.

Per = p(k+d) – pk = pd deci Per este divizibil cu p.

Dar pentru a avea aceeași poziție R[k+d] == R[k] adică p(k+d) % m == pk % m => pd % m == 0 => Per % m == 0 => Per este divizibil cu m.

Atunci Per este divizibil cu cmmmc(p,m). Deoarece vrem ca Per să fie minim luăm Per = cmmmc(p,m) și, foarte important valoarea lui D nu depinde de k!!! Asta ne spune că, după exact Per secunde, Robin Hood va ajunge la ținta 0 pe care o va atinge pentru prima dată. Din acel moment din Per în Per secunde el va repeta ce a făcut în primele Per secunde, deci nu va mai atinge alte ținte față de cele pe care le-a atins deja. Simulând astfel tot ce face Robin Hood în primele Per secunde dar numai în momente de forma p*k putem să determinăm timpul minim la care acesta trage în fiecare țintă pe care o poate atinge.

Pentru Little John situația este foarte asemănătoare cu o singură diferență, la momentul 0 inițial el se găsește în dreptul țintei n-1 și se deplasează spre stânga și ne interesează doar momentele de forma q*k. În astfel de momente poziția se poate calcula analog cu cazul lui Robin Hood dar folosind restul (n-1+q*k) % m pentru a determina poziția unde se găsește Little John la momentul de timp q*k. Deasemenea, pentru Little John trebuie simulat ce face Little John din q în q secunde, până la momentul cmmmc(q,m).

Odată simulate tragerile celor doi, folosind timpii minimi la care trag Robin Hood sau Little John în fiecare țintă se determină timpul minim la care va fi atinsă fiecare țintă.

Calculând maximul acestor valori peste toate țintele se va determina timpul minim căutat pentru prima cerință.

Pentru a calcula câte săgeți ating fiecare țintă se va relua simularea din p în p secunde pentru Robin Hood, respectiv din q în q secunde pentru Little John, dar de această dată doar până la timpul minim calculat anterior.

Pentru fiecare poziție prin care trecem în cele două simulări creștem cu 1 numărul de săgeți al poziției respective.

La final, știind câte săgeți au atins fiecare țintă se pot rezolva cerințele 2 și 3 la fel ca la celelalte soluții descrise în acest editorial.

Problema 2: Puzzle

Propusă de: prof. Costineanu Raluca, Colegiul Național "Ștefan cel Mare", Suceava

Descrierea unor soluţii posibile

Soluția 1.

Propusă de: prof. Costineanu Raluca, Colegiul Național "Ștefan cel Mare", Suceava

În timpul citirii datelor vom construi un vector care să rețină etichetele grupurilor pe piese care se pot forma conform regulilor descrise:
- prima etichetă citită este adăugată pe prima poziție a vectorului
- la fiecare dintre următoarele $N-1$ citiri:
  - dacă ultimul element din șir are cel puțin trei cifre în comun cu numărul citit atunci îi actualizăm acestuia valoarea (păstrând doar primele patru cifre și lipind de acestea ultimele patru cifre ale numărului citit)
  - altfel adăugăm la vector un nou element cu valoarea numărului citit

Cerința 1: vom afișa numărul de grupuri obținute la finalul prelucrării anterioare

Cerința 2: vom ordona vectorul obținut descrescător după numărul de cifre distincte conținut de fiecare număr, iar în caz de egalitate a numărului de cifre distincte vom ordona crescător după valoare. Afișam apoi primele K elemente din șir.

Soluția 2.

Propusă de: prof. Rotar Dorin-Mircea, Colegiul Național "Samuil Vulcan", Beiuș

În implementare soluției utilizarea vectorilor și a funcțiilor auxiliare ajută la gestionarea logică a unirii etichetelor și la verificarea compatibilității între piese, făcând codul modular și mai ușor de urmărit.

Cerința 1: Pentru prima cerință am folosit un vector de string în care reținetichetele grupurile formate în joc. Se începe cu un grup initial format din prima etichetă citită. Fiecare etichetă nouă citită este verificată dacă poate fi lipită la ultimul grup activ, bazat pe numărul de cifre comune. Dacă eticheta curentă se poate lipi, se actualizează eticheta grupului. Dacă nu, se începe un nou grup cu eticheta respectivă. La sfârșitul iterației prin etichete, numărul total de grupuri formate este înregistrat. Se afișează numărul de grupuri determinat.

Cerința 2: Pentru rezolvarea celei de-a doua cerințe am folosit un vector de perechi de numere întregi unde primul termen din pereche reprezintă numărul de cifre distincte a grupării, iar al doilea termen este eticheta grupării sub formă negativă. Etichetele grupurilor sunt sortate folosind sortarea prin selecție pentru primele K grupări după numărul de cifre distincte, iar în caz de egalitate, după valoarea numerică a etichetei, prioritate având cele cu valoarea mai mică. Se selectează primele K etichete conform ordinii stabilite. Se afișează etichetele selectate, separate prin spațiu.

Problema 3: P13

Propusă de: prof. Pițˌ-Rada Ionel-Vasile, Colegiul Național Traian, Drobeta-Turnu Severin

Descrierea unor soluţii posibile

Soluția 1.

Propusă de: prof. Panete Adrian, Colegiul Național "A. T. Laurian", Botoșani

Pentru fiecare poziție i din șir calculăm:

U[i] = câte cifre de 1 conține numărul de pe poziția i;

T[i] = câte cifre de 3 conține numărul de pe poziția i.

În funcție de paritățile numerelor U[i] și T[i] vom asocia poziției i un tip t[i] după cum urmează:

- t[i] = 0 dacă U[i] = T[i] = 0;
- t[i] = 1 dacă U[i] = impar T[i] = par;
- t[i] = 2 dacă U[i] = par T[i] = impar;
- t[i] = 3 dacă U[i] = impar T[i] = impar;
- t[i] = 4 dacă U[i] = par T[i] = par și măcar una dintre valorile U[i], T[i] este nenulă.

Numărul de pe poziția i este special atunci când t[i] = 4.

Cerința 1. Pentru cerința 1 este suficient să numărăm câte numere din șir au tipul 4.

Cerința 2. Pentru cerința 2 vom număra secvențele după capătul lor din dreapta. Notăm acest capăt cu dr.

Dacă înaintea poziției dr nu am numere speciale atunci nu avem secvențe soluție.

Considerăm acum s1 = cea mai apropiată poziție care conține număr special și s1 <= dr

Considerăm s2 cea mai apropiată poziție de s1 care conține număr special și s2 < s1 (dacă nu există o astfel de poziție se ia s2 = 0).

Să considerăm un interval soluție [st,dr].

Observăm că:

1. st <= s1 pentru a avea cel puțin un număr special în interval;

2. st >= s2 + 1 pentru a nu avea un al doilea număr special în interval, sau pentru a avea capătul stânga >= 1;

3. st >= dr-k+1 (pentru ca lungimea secvenței să nu depășească k).

Astfel valoarea minimă pe care o poate lua capătul st este stMin = max(s2+1,dr-k+1) iar valoarea maximă este stMax=s1.

Dacă stMin <= stMax se adună la soluție toate variantele posibile în care putem alege poziția st adică stMax-stMin+1.

Cerința 3. Pentru fiecare interval [st,dr] de lungime k vom memora:

cntU = numărul de 1 din interval;

cntT = numărul de 3 din interval.

Last[i] <= dr ultima poziție situată în fața poziției dr unde avem un număr de tip i (0, 1, 2, 3 sau 4).

Asta înseamnă că prin lipirea tuturor numerelor din interval se va forma un număr NR cu cntU cifre de 1 și cntT cifre de 3 iar acest număr va avea un tip TP cu valoarea 0, 1, 2, 3 sau 4 calculat conform regulii descrise mai sus.

De asemenea observăm că pe intervalul [st,dr] există măcar un număr de tip i dacă last[i] >=st și astfel vom ști sigur pentru fiecare tip dacă avem sau nu numere de acel tip pe interval. Dacă NR este de tip 0 atunci răspunsul la cerința 3 este -1 deoarece indiferent de eliminări vom rămâne cu număr de tip 0.

- Dacă NR este de tip 4 atunci răspunsul la cerința 3 este 0 pentru că NR deja este special.

- Dacă NR este de tip 1 avem două alternative:
  - eliminăm un număr de tip 1 (o eliminare);
  - eliminăm un număr de tip 2 și unul de tip 3 (două eliminări).

- Dacă NR este de tip 2 avem două alternative:
  - eliminăm un număr de tip 2 (o eliminare);
  - eliminăm un număr de tip 1 și unul de tip 3 (două eliminări).

- Dacă NR este de tip 3 avem două alternative:
  - eliminăm un număr de tip 3 (o eliminare);
  - eliminăm un număr de tip 1 și unul de tip 2 (două eliminări).

În toate aceste cazuri va trebui să verificăm dacă avem valori de tipurile necesare eliminărilor, dar asta știm folosind valorile last[i].

Deasemenea trebuie să ne asigurăm că, după eliminare, ne mai rămân valori de 1 sau 3. Deoarece noi știm exact din care poziții vom face una sau două eliminări, vom ști dacă nu am obținut eliminarea a cntU de 1 și a cntT de 3 (situație în care nu putem folosi acel tip de eliminare).

Evident că pentru fiecare tip 1, 2 sau 3 vom încerca prima dată alternativa cu o eliminare. Dacă nu este posibilă o eliminare atunci încercăm alternativa cu două eliminări. Dacă nici aceasta nu este posibilă atunci răspunsul va fi -1.

Observație finală: prin cunoașterea valorilor U[i] și T[i] este foarte ușor să recalculăm valorile cntU, cntT, last[i] pentru orice interval [st,dr] de lungime k știind aceste valori corespunzătoare intervalului [st-1,dr-1].

Soluția 2.

Propusă de: prof. Pițˌ-Rada Ionel-Vasile, Colegiul Național Traian, Drobeta-Turnu Severin

Cerința 1: Se parcurg numerele citite și se verifică la fiecare dintre ele proprietatea de număr special și se actualizează contorul când este cazul.

Cerința 2: Presupunem că numerele date sunt păstrate într-un vector V[1], V[2], ..., V[n].

Pentru fiecare număr special (să zicem V[p]) vom căuta spre stânga și spre dreapta două poziții p1 și p2 astfel încât să avem 1<=p1<=p<=p2<=n și p-p1+1<=K și p2-p+1<=K și în secvența V[p1],...,V[p2] să avem ca număr special doar pe V[p].

Observăm că, în secvența determinată orice secvență inclusă care are lungimea cel puțin K va conține și numărul special V[p].

Numărul total de secvențe incluse în secvența V[p1],...V[p2] este $\mathbf{1+2+...+(p2-p1+1)}$ care conform formulei lui Gauss este egal cu $\mathbf{(p2-p1+1)*(p2-p1+2)/2}$.

Din aceste secvențe va trebui să eliminăm secvențele incluse în secvențele V[p1],...,V[p-1] (cele din stânga numărului special V[p]) și V[p+1],...,V[p2] (cele din dreapta numărului special V[p]) care sunt în număr de $\mathbf{(p-p1)*(p-p1+1)/2}$ și respectiv $\mathbf{(p2-p)*(p2-p+1)/2}$, deoarece ele nu conțin numărul special V[p]. Deasemenea trebuie să eliminăm secvențele care au lungime mai mare decât K. Acestea există numai dacă p2-p1+1>K. În acest caz avem secvențele care au capătul stâng x în [p1...p2-K] și capătul drept în [x+K...p2]. Numărul lor este 1+2+...+(p2-p1-K+1) și conform formulei Gauss avem $\mathbf{(p2-p1-K+1)*(p2-p1-K+2)/2}$.

Cerința 3: Analizăm pe rând fiecare secvență de lungime K.

Singurele cazuri în care nu se poate obține prin concatenare un număr special, după eventuale ștergeri, sunt următoarele:

(**1**) în numerele din secvență nu apar cifre impare;

(**2**) singurele numere din secvență care conțin cifre impare trebuie șterse pentru a rezolva imparitatea numărului de apariții ale cifrelor impare.

Cazul (2) conține două subcazuri:

(2.1) avem un singur număr care conține cifre impare, dar care "strică" paritatea cifrelor impare din secvență;

(2.2) avem doar două numere care conțin cifre impare și unul "strică" paritatea cifrei impare 1 din secvență, iar al doilea "strică" paritatea cifrei impare 3 din secvență.

Analizând cele două subcazuri observăm că ar trebui numărate separat elementele care conțin număr total impar de cifre de 1 (vom nota numărul acestora cu **y1**) și respectiv de 3 (numărul acestora cu **y3**), respectiv ambele (număr notat cu **y13**), respectiv ar trebui numărate elementele care conțin cifre impare, dar în număr par de apariții (numere speciale, vom nota cu **ysp**).
Dacă dintre **y1**, **y3** și **y13** sunt toate **impare**, atunci nu trebuie șters nimic.
Dacă dintre **y1**, **y3** și **y13** toate sunt **pare**, atunci nu trebuie șters nimic.
Dacă dintre **y1**, **y3** și **y13 unul** este **par nenul**, iar celelalte două sunt impare, atunci trebuie șters un număr care să scadă numărul par.
Dacă dintre **y1**, **y3** și **y13** două sunt pare, iar al treilea este impar, atunci trebuie șters un număr care să scadă pe cel impar.
Dacă dintre **y1**, **y3** și **y13** unul este nul iar celelalte sunt impare, atunci trebuie șterse două numere care să scadă pe cele două impare.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Costineanu Raluca, Colegiul Național "Ștefan cel Mare", Suceava
- Prof. Nicoli Marius, Colegiul Național "Frații Buzești", Craiova
- Prof. Panete Adrian, Colegiul Național "A. T. Laurian", Botoșani
- Prof. Pintescu Alina Colegiul Național "Gheorghe Șincai", Baia Mare
- Prof. Piț-Rada Ionel-Vasile, Colegiul Național Traian Drobeta-Turnu Severin
- Prof. Rotar Dorin-Mircea, Colegiul Național "Samuil Vulcan", Beiuș
- Prof. Rotaru Elena, Colegiul Național, Iași
- Prof. Șerban Marinel, Colegiul Național "Emil Racoviță", Iași