PROIECTAREA ALGORITMILOR – TEMA 2

DEADLINE HARD: ~~31.05.2021~~

02.06.2021

ULTIMA MODIFICARE: 28.05.2021

RESPONSABILI TEMĂ: MIHAI NAN, DAVID IANCU, DOINA CHIROIU, ANDREI CLICINSCHI, ANDREI PREDA

RESPONSABILI FORUM: IOAN–LUCIAN OPREA, TEODOR MATEI

UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

ANUL UNIVERSITAR 2020–2021

Cuprins

Problema 1 – 25 de puncte

1.1 Enunț

Gigel se află pe un pod deasupra unei întinderi de apă în forma unui grid cu N linii și M coloane. Deasupra acestei suprafețe au fost construite trei tipuri de poduri: verticale (**V**), orizontale (**O**) sau duble (**D**). Denumirile acestor tipuri de poduri indică direcțiile în care se poate deplasa Gigel pe un astfel de pod.

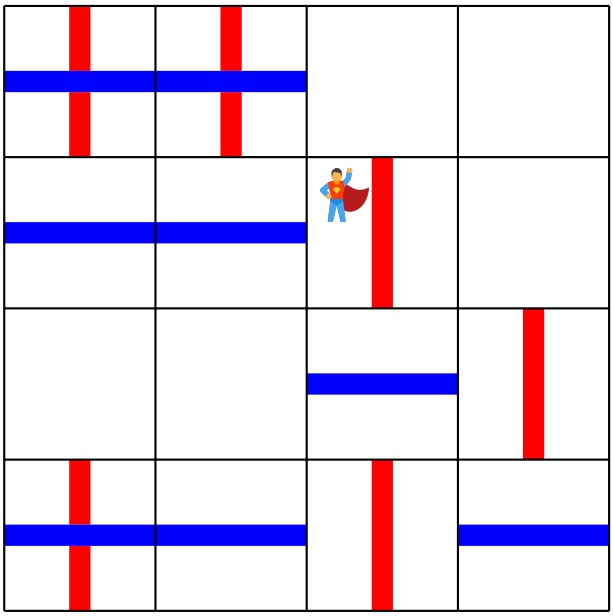
1. pod vertical (**V**) – permite doar deplasarea cu o celulă la Nord sau cu o celulă la Sud;
2. pod orizontal (**O**) – permite doar deplasarea cu o celulă la Vest sau cu o celulă la Est;
3. pod dublu (**D**) – permite deplasarea cu o singură celulă la Nord sau cu o singură celulă la Est sau cu o singură celulă la Sud sau cu o singură celulă la Vest.

Deoarece întinderea de apă este foarte mare, nu au existat fonduri pentru a fi construite poduri pe toate celulele. Astfel, aceste celule care nu conțin un pod nu sunt accesibile pentru Gigel. Cu alte cuvinte, dacă Gigel se află pe un pod el poate să se deplaseze doar în direcțiile permise de tipul podului dacă ajunge să fie poziționat într-o celulă accesibilă (celulă care conține un pod) sau dacă ajunge pe uscat (în afara gridului).
Voi va trebui să îl ajutați pe Gigel să ajungă pe uscat (în afara gridului), traversând un număr cât mai mic de poduri. Este garantat că Gigel se află inițial într-o celulă care conține un pod, însă nu este garantat că există o variantă prin care Gigel să ajungă pe uscat. În cazul în care nu există posibilitatea ca Gigel să ajungă pe uscat, va trebui să

afișați valoarea -1. Altfel, va trebui să determinați numărul minim de poduri pe care trebuie să le traverseze pentru a ajunge pe uscat.

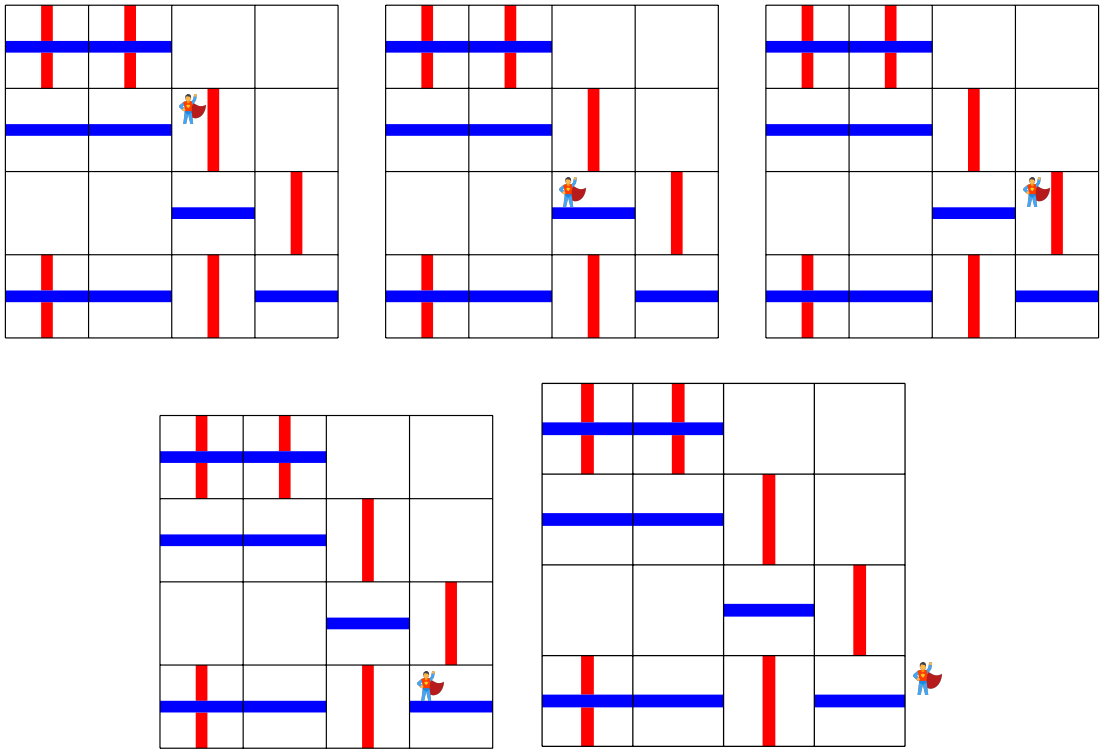
1.2 Exemplu detaliat

Intrare:



Rezultat: 4

Explicație:



1.3 Date de intrare

Fișierul de intrare `poduri.in` va conține pe prima linie două numere naturale separate prin câte un spațiu: N și M.
Pe a doua linie din fișier vor exista două numere naturale separate prin câte un spațiu: X și Y; reprezentând coordonatele podului pe care se află Gigel inițial.
Următoarele N linii vor conține câte M caractere fiecare (câte unul pentru fiecare celulă din grid). Caracterul `V` este folosit pentru a marca zonele care conțin poduri verticale, caracterul `O` este folosit pentru a marca zonele care conțin poduri orizontale, caracterul `D` este folosit pentru a marca zonele care conțin poduri duble, iar caracterul `.` (punct) este folosit pentru a evidenția celulele din libere din grid (cele care nu conțin un pod).

1.4 Date de ieșire

În fișierul `poduri.out` se va scrie numărul minim de poduri pe care trebuie să le traverseze Gigel pentru a ajunge pe uscat.

1.5 Restricții și precizări

- $1 \leq N, M \leq 2050$

Observație

Dacă Gigel nu poate ajunge pe uscat, atunci se va afișa rezultatul `-1`.
Vom considera că indexarea pozițiilor în grid începe cu 1.

1.6 Testare și punctare

- Punctajul maxim pentru această problemă este **25 de puncte**.
- Timpul de execuție:
  - **C/C++: 0.5 secunde**
  - **Java: 1 secundă**

Important

Sursa care conține funcția `main` trebuie obligatoriu denumită: `poduri.c`, `poduri.cpp` sau `Poduri.java`.

1.7 Exemple

1.7.1 Exemplu 1

Exemplu 1		
poduri.in	poduri.out	Explicație
4 4 2 3 DD.. OOV. ..OV DOVO	4	Exemplul detaliat anterior

1.7.2 Exemplu 2

Exemplu 1		
poduri.in	poduri.out	Explicație
10 8 3 7 ..VOVDDD D.DDO.DD DDVDDDDD DDDDVVDD VD.DOOD. VV.VVODD VOOOVO.. OVOODDOO .DDD...D VV.VD.OD	2	Pe poziția (3, 7) se află un pod dublu și Gigel se poate deplasa pe orizontală, ajungând în poziția (3, 8). Pe poziția (3, 8) se află tot un pod dublu și Gigel se poate deplasa pe orizontală și ajunge pe uscat.

2

Problema 2 – 45 de punct

2.1 Enunț

Gigel a fost angajat la Google și dorește să determine câte persoane distincte folosesc sistemul de poștă electronică. El are acces la o listă cu posibile persoane și pentru fiecare persoană o listă de e-mail-uri. Din păcate, multe dintre aceste persoane sunt fictive, fiind vorba despre aceeași persoană, deoarece o persoana poate avea mai multe liste de e-mail-uri. Doua persoane considerate distincte pot avea același nume, dar e-mailurile trebuie să fie diferite. Dacă totuși două persoane cu nume diferit au cel puțin un e-mail comun, înseamnă că este vorba de aceeași persoana.
Trebuie să îl ajutați pe Gigel să determine câte persoane distincte există și pentru fiecare persoană în parte să se afișeze lista de e-mailuri, persoanele fiind sortate după numărul de e-mailuri, apoi lexicografic, iar e-mailurile afișate în ordine lexicografică. Dacă există mai multe nume pentru aceeași persoană, se va afișa cel minim lexicografic.

2.2 Date de intrare

Fișierul de intrare `adrese.in` conține pe prima linie numărul natural N (numărul de persoane care urmează să fie analizate).
Pentru fiecare din cele N persoane se vor furniza următoarele date:

- pe prima linie va exista numele persoanei și un număr natural K reprezentând numărul de adrese cunoscute pentru persoana respectivă, cele două fiind despărțite printr-un spațiu;
- pe următoarele K linii se vor găsi adresele de e-mail (câte o adresă pe linie).

2.3 Date de ieșire

În fișierul de ieșire `adrese.out` se va scrie pe prima linie numărul M de persoane distincte.
Pentru fiecare din cele M persoane distincte se vor scrie următoarele date:

- pe prima linie se va scrie numele persoanei și numărul T de adrese de e-mail pe care această persoană le deține, cele două vor fi separate printr-un spațiu;
- pe următoarele T linii se vor scrie adrese de e-mail (câte o adresă pe linie).

2.4 Restricții și precizări

- $1 \leq N \leq 1000$
- $1 \leq K_i \leq 500, \forall i \in \{1, 2, \ldots, N\}$
- Numele persoanelor vor fi formate din maximum 20 de caractere și nu vor conține caracterul spațiu.
- Adrese de e-mail vor fi formate din maximum 50 de caractere și nu vor conține caracterul spațiu.

2.5 Testare și punctare

- Punctajul maxim pentru această problemă este **45 de puncte**.
- Timpul de execuție :
  - **C/C++: 1 secunde**
  - **Java: 3 secunde**

Important

Sursa care conține funcția `main` trebuie obligatoriu denumită: `adrese.c`, `adrese.cpp` sau `Adrese.java`.

2.6 Exemple

2.6.1 Exemplul 1

Exemplu 1		
adrese.in	**adrese.out**	Explicație
5 Dlkra 2 serrnkv@stud.pub.ro nbjhxbe@stud.pub.ro Fmbvd 2 sxxrmgt@yahoo.com cflarbq@stud.pub.ro Qppah 1 nlcxses@outlook.com Fubto 2 ynsucvp@stud.pub.ro slcxwqc@hotmail.com Exmxb 1 pyxrnit@outlook.com	5 Exmxb 1 pyxrnit@outlook.com Qppah 1 nlcxses@outlook.com Dlkra 2 nbjhxbe@stud.pub.ro serrnkv@stud.pub.ro Fmbvd 2 cflarbq@stud.pub.ro sxxrmgt@yahoo.com Fubto 2 slcxwqc@hotmail.com ynsucvp@stud.pub.ro	Toate persoanele sunt distincte, deoarece nu au adrese comune. Ele sunt afișate în varianta sortată cerută în enunț.

2.6.2 Exemplul 2

Exemplu 1		
adrese.in	**adrese.out**	Explicație
5 David 3 davidiancu@yahoo.com davidiancudvd@gmail.com funkydvd@gmail.com Mihai 2 mihainan@yahoo.com mihai.nan.cti@gmail.com Traian 3 traianiancu@yahoo.com traiantrn@gmail.com funkydvd@gmail.com Mihai 2 mihait@yahoo.com mihai24@gmail.com Mihai 2 mihai2345@yahoo.com mihainan@yahoo.com	3 Mihai 2 mihai24@gmail.com mihait@yahoo.com Mihai 3 mihai.nan.cti@gmail.com mihai2345@yahoo.com mihainan@yahoo.com David 5 davidiancu@yahoo.com davidiancudvd@gmail.com funkydvd@gmail.com traianiancu@yahoo.com traiantrn@gmail.com	David și Traian sunt aceeași persoană (au în comun e-mailul funkydvd@gmail.com) Primul și al 3-lea Mihai sunt aceeași persoană (au în comun adresa mihainan@yahoo.com), iar al 2-lea Mihai este o persoana diferită.

3

Problema 3 – 45 de puncte

3.1 Enunț

Fiind pandemie, David a descoperit un nou joc. El are la dispoziție un număr mare de piese de LEGO de dimensiune maxim K, care au proprietatea specială că se pot *lipi* una de cealaltă, astfel încât piesa rezultată va avea dimensiunea egală cu suma dimensiunilor celor două piese. Pentru că se plictisește foarte tare, el vrea să selecteze N piese de LEGO astfel încât să poată forma cât mai multe dimensiuni diferite, combinând piesele între ele. În plus, își impune o restricție suplimentară – se pot folosi maxim T piese. De exemplu, dacă $K = 3$, $N = 2$ și $T = 3$, pentru dimensiunile pieselor de bază 1 și 2 putem obține dimensiunile $1, 2, 3, 4, 5, 6$, însă dacă vom considera dimensiunile de bază 1 și 3, putem obține dimensiunile $1, 2, 3, 4, 5, 6, 7, 9$. Din această cauză, îi vine în minte următoarea întrebare – care este cel mai mare număr de dimensiuni diferite care poate fi obținut din N piese de bază de dimensiune maxim K, având la dispoziție maxim T piese de orice fel? Totuși, David își dă seama că răspunsul la această întrebare nu este așa de greu precum pare, așa că acum vrea să afle care este cel mai mare număr de dimensiuni consecutive care poate fi obținut din N piese de bază. În exemplul de mai sus, deși se pot obține 8 dimensiuni diferite, doar 7 sunt consecutive, acesta fiind numărul dorit. De asemenea, David dorește să afle și mărimea pieselor de bază cu care se pot forma cele mai multe dimensiuni consecutive. În exemplul de mai sus, dimensiunile sunt 1 și 3. Pentru că pot exista mai multe soluții, pe David îl interesează cea minimă lexicografic (vă oferă și un hint: una din piese trebuie să aibă dimensiune 1).

3.2 Date de intrare

Fișierul de intrare `lego.in` va conține pe prima linie 3 numere naturale separate prin câte un spațiu: K, N și T.

3.3 Date de ieșire

În fișierul de ieșire `lego.out` se va scrie pe prima linie un număr natural, reprezentând cel mai mare număr de dimensiuni consecutive care poate fi obținut.
Pe a doua linie a fișierului se vor scrie, despărțite printr-un spațiu, dimensiunile pentru piesele de bază cu care se pot forma cele mai multe dimensiuni consecutive.

3.4 Restricții și precizări

- $1 \leq K \leq 50$
- $1 \leq N \leq 10$
- $1 \leq T \leq 20$

3.5 Testare și punctare

- Punctajul maxim pentru această problemă este **45 de puncte**.
- Timpul de execuție:
  - **C/C++: 5 secunde**
  - **Java: 10 secunde**

Important

Sursa care conține funcția `main` trebuie obligatoriu denumită: `lego.c`, `lego.cpp` sau `Lego.java`.

3.6 Exemple

3.6.1 Exemplu 1

Exemplu 1		
`lego.in`	`lego.out`	Explicație
`3 2 3`	`7` `1 3`	1 = 1 2 = 1 + 1 3 = 3 4 = 1 + 3 5 = 1 + 1 + 3 6 = 3 + 3 7 = 3 + 3 + 1

3.6.2 Exemplu 2

Exemplu 1		
`lego.in`	`lego.out`	Explicație
`5 3 3`	`15` `1 4 5`	1 = 1 2 = 1 + 1 3 = 1 + 1 + 1 4 = 4 5 = 5 6 = 1 + 5 7 = 1 + 1 + 5 8 = 4 + 4 9 = 4 + 5 10 = 5 + 5 11 = 5 + 5 + 1 12 = 4 + 4 + 4 13 = 4 + 4 + 5 14 = 4 + 5 + 5 15 = 5 + 5 + 5

4

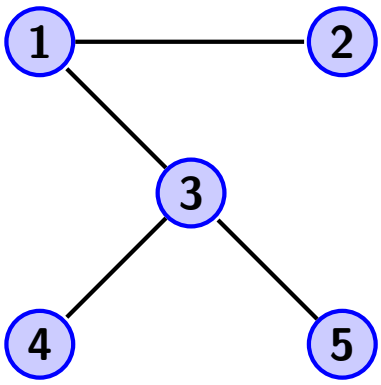
Bonus – 25 de puncte

4.1 Enunț

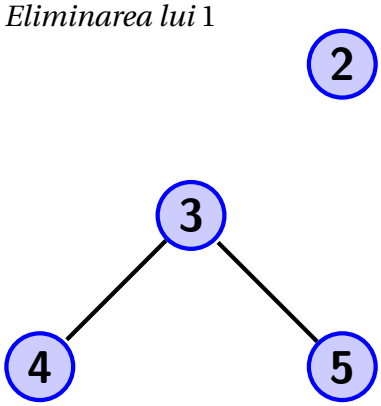
Gigel s-a angajat la o firmă de rețelistică. Aici descoperă o rețea internă formată din N calculatoare conectate între ele prin M legături bidirecționale. Fiecare calculator aparține unui angajat din firmă. Din moment ce pandemia nu le mai permite angajaților acestei firme să se întâlnească zilnic la locul de muncă pentru a socializa și a se cunoaște mai bine, managerul companiei s-a decis să le impună acestora să comunice între ei folosind rețeaua internă. Astfel, fiecare angajat este obligat să comunice cu toți ceilalți colegi folosind un apel inițiat de el. Pentru ca angajatul A să comunice cu angajatul B trebuie ca între calculatoarele acestora să existe o conexiune care să asigure schimbul de pachete astfel încât un pachet să treacă exact o dată pe la fiecare calculator implicat în conexiunea ce permite comunicarea. Gigel, fiind nemulțumit de salariul pe care îl primește, se gândește să își dea demisia, dar nu înainte de a se asigura că strică rețeaua internă a firmei. Astfel, vă cere ajutorul pentru a identifica, pentru fiecare calculator din rețea, câte apeluri nu se vor mai putea iniția dacă este eliminat din rețea calculatorul respectiv.

4.2 Exemplu detaliat

Fie următoarea graful care descrie următoarea rețea:

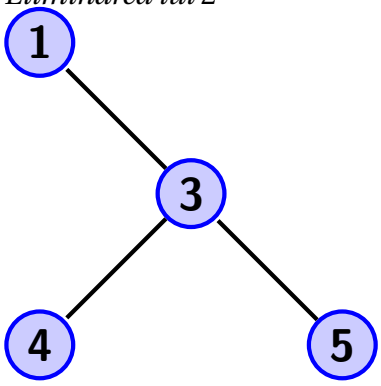


Eliminarea lui 1



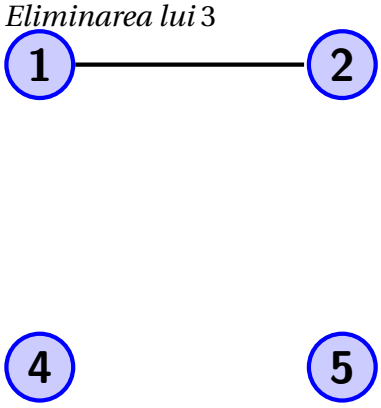
Dacă îl eliminăm pe 1, atunci o să dispară toate apelurile inițiate de el (4), toate apelurile inițiate de 2 (4), apelurile inițiate de 3 către 1 și 2 (2), apelurile inițiate de 4 către 1 și 2 (2), apelurile inițiate de 5 către 1 și 2 (2). În total: $4+4+2+2+2=14$.

Eliminarea lui 2



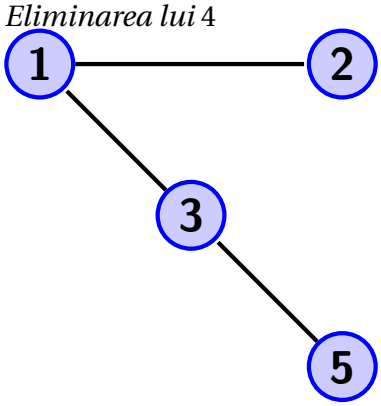
Dacă îl eliminăm pe 2, atunci o să dispară toate apelurile inițiate de el (4), apelurile inițiate de $1,3,4$ și 5 către 2 (4). În total: $4+4=8$.

Eliminarea lui 3



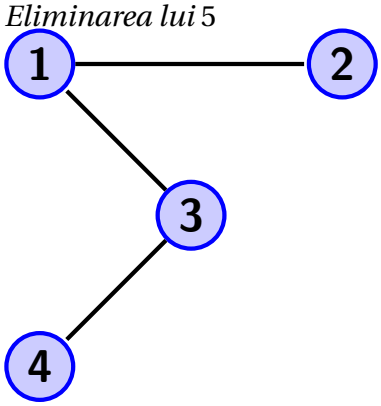
Dacă îl eliminăm pe 3, atunci o să dispară toate apelurile inițiate de el (4), toate apelurile inițiate de 4 (4), toate apelurile inițiate de 5 (4), apelurile inițiate de 1 către 3,4 și 5 (3) și apelurile inițiate de 2 către 3,4 și 5 (3). În total: $4+4+4+3+3=18$.

Eliminarea lui 4



Dacă îl eliminăm pe 4, atunci o să dispară toate apelurile inițiate de el (4), apelurile inițiate de 1,2,3 și 5 către 4 (4). În total: $4+4=8$.

Eliminarea lui 5



Dacă îl eliminăm pe 5, atunci o să dispară toate apelurile inițiate de el (4), apelurile inițiate de 1,2,3 și 4 către 5 (4). În total: $4+4=8$.

4.3 Date de intrare

Fișierul de intrare `retea.in` conține pe prima linie două numere naturale separate prin câte un spațiu: N și M.
Pe următoarele M linii se vor găsi câte două numere naturale: x și y; despărțite printr-un spațiu (având semnificația că există în rețea legătură directă între x și y).

4.4 Date de ieșire

În fișierul de ieșire `retea.in` se vor scrie N linii, iar pe fiecare linie va exista un unic număr natural reprezentând numărul de apeluri care nu se vor mai putea iniția dacă este eliminat din rețea calculatorul cu indicele liniei.

4.5 Restricții și precizări

- $1 \leq N \leq 10000$
- $1 \leq M \leq 20000$

Observație

Este garantat că există o cale între oricare două noduri din rețea.

4.6 Testare și punctare

- Punctajul maxim pentru această problemă este **25 de puncte**.
- Timpul de execuție:
  - **C/C++: 0.2 secunde**
  - **Java: 0.4 secunde**

Important

Sursa care conține funcția `main` trebuie obligatoriu denumită: `retea.c`, `retea.cpp` sau `Retea.java`.

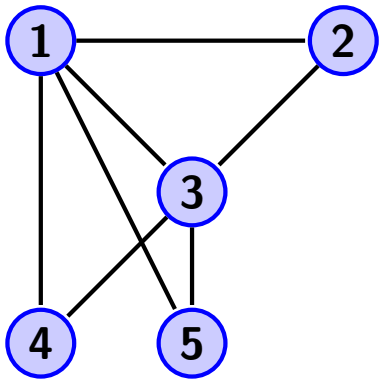
4.7 Exemple

4.7.1 Exemplu 1

Exemplu 1		
`retea.in`	`retea.out`	Explicație
5 4 1 2 1 3 3 4 3 5	14 8 18 8 8	Exemplul detaliat anterior

4.7.2 Exemplu 2

Exemplu 1		
`retea.in`	`retea.out`	Explicație
5 7 1 2 1 3 1 4 1 5 2 3 3 4 3 5	8 8 8 8 8	Orice nod am elimina, apelurile care o să dispară sunt cele inițiate de el (4) și cele inițiate de către celelalte noduri către el (4). În total, o să dispară $4+4=8$ pentru fiecare nod.



5

Punctare

• Punctajul temei este de 125 de puncte, distribuit astfel:

- Problema 1: 25 de puncte
- Problema 2: 45 de puncte
- Problema 3: 45 de puncte
- 5 puncte vor fi acordate pentru comentarii și README
- 5 puncte vor fi acordate pentru coding style in mod automat de catre checker.
- Se pot realiza depunctari de pană la 20 de puncte pentru coding style neadecvat la corectarea manuală a temelor.

Punctajul pe README, comentarii și coding style este condiționat de obținerea a unui punctaj strict pozitiv pe cel puțin un test.
Se poate obține un bonus de 25 de puncte rezolvând problema bonus. Acordarea bonusului **NU** este condiționată de rezolvarea celorlate probleme. În total se pot obține 150 de puncte (**NU** se trunchiază).
Pentru detalii puteți să vă uitați și peste regulile generale de trimitere a temelor.
• O temă care **NU** compilează va fi punctată cu 0.
• O temă care **NU** trece niciun test pe vmchecker va fi punctată cu 0.
• Vor exista mai multe teste pentru fiecare problemă în parte. Punctele pe teste sunt independente, punctajul pe un anumit test nefiind condiționat de alte teste.
• Fiecare problemă va avea o limită de timp pe test (precizată mai jos și pe pagina cu enunțul). Dacă execuția programului pe un test al acelei probleme va dura mai mult

decât limita de timp, veți primi automat 0 puncte pe testul respectiv și execuția va fi întreruptă.
• În fișierul README va trebui să descrieți soluția pe care ați ales-o pentru fiecare problemă, să precizați complexitatea pentru fiecare și alte lucruri pe care le considerați utile de menționat.

5.1 Checker

• Arhiva se va trimite pe vmchecker, unde tema se va testa folosind un set de teste private.
• Pentru testarea locala, aveți disponibil un set de teste publice (de aceeași dificultate) pe pagina cu resurse a temei.
• Checkerul se poate rula fara niciun parametru, caz in care va verifica toate problemele. De asemenea se mai poate rula cu un parametru pentru a rula o anumită problemă:
./check.sh <1 | 2| 3 | 4>
./check.sh <poduri | adrese | lego | retea >
./check.sh cs

• **Punctajul pe teste** este cel de pe vmchecker și se acordă ruland tema doar cu testele private.
• Checkerul verifică doar existența unui README cu denumire corectă și conținut nenul.
Punctajul final pe README și comentarii se acordă la corectarea manuală a temei.
• La corectare se poate depuncta pentru **erori de coding style** care nu sunt semnalate de checker.
• Corectorii își rezervă dreptul de a scădea puncte pentru orice problemă găsită în implementare, dacă vor considera acest lucru necesar.
• Pentru citirea in Java se recomanda folosirea **BufferedReader**.

6

Format arhivă

• Temele pot fi testate automat pe vmchecker. Acesta suportă temele rezolvate în C/C++ și Java. Dacă doriți să realizați tema în alt limbaj, trebuie să-i trimiteți un email lui Traian Rebedea (traian.rebedea@cs.pub.ro), în care să îi cereți explicit acest lucru.
• Arhiva cu rezolvarea temei trebuie să fie **.zip**, având un nume de forma **Grupa_NumePrenume_Tema2.zip** (ex: 399CX_PuiuGigel_Tema2.zip sau 399CX_BucurGigel_Tema2.zip) și va conține:

- Fișierul/fișierele sursă
- Fișierul **Makefile**
- Fișierul **README** (fără extensie)

• Fișierul pentru make trebuie denumit obligatoriu **Makefile** și trebuie să conțină următoarele reguli:

- **build**, care va compila sursele și va obține executabilele
- **run-p1**, care va rula executabilul pentru problema 1
- **run-p2**, care va rula executabilul pentru problema 2
- **run-p3**, care va rula executabilul pentru problema 3
- **clean**, care va șterge executabilele generate
- **run-p4**, care va rula executabilul pentru problema bonus (**doar dacă** ați implementat și bonusul)

Atenție!

Funcția **main** din rezolvarea unei probleme se va găsi într-o sursă ce trebuie obligatoriu denumită astfel:

- **poduri.c**, **poduri.cpp** sau **Poduri.java** - pentru problema 1
- **adrese.c**, **adrese.cpp** sau **Adrese.java** - pentru problema 2
- **lego.c**, **lego.cpp** sau **Lego.java** - pentru problema 3
- **retea.c**, **retea.cpp** sau **Retea.java** - pentru problema bonus

Atenție!

Tema va fi compilată și testată **DOAR pe Linux**.

Atenție!

Numele regulilor și a surselor trebuie să fie exact cele de mai sus. Absența sau denumirea diferită a acestora va avea drept consecință obținerea a 0 puncte pe testele asociate problemei rezolvate de regula respectivă.

Atenție!

Pentru cei ce folosesc C/C++ **NU** este permisă compilarea cu opțiuni de optimizare a codului (O1, O2, etc.).

Atenție!

Orice nerespectare a restricțiilor duce la pierderea punctajului (după regulile de mai sus).

7 Links

- Regulament general teme PA
- Google C++ Style Guide
- Google Java Style Guide
- Debugging și Structuri de Date

8 Modificări

- Corectarea conținutului fișierului `adrese.out` pentru **Exemplul 2** (Vezi 2.6.2).
- A fost extinsă limita de timp pentru Problema 2 (Vezi 2.5).