Zadání projektu do předmětu **Algoritmy I**
letní semestr 2024/2025
prezenční forma studia

Historie modifikací

31. března 2025 První verze
11. dubna 2025 Doplnění ukázek acyklických grafů, doplnění testovacích dat

Obsah

Deadline

Vypracované řešení projektu je nutno odevzdat do **18. května 2025 23:59**.

Obecné pokyny

- Každý student či studentka má přiděleno jedno zadání. Rozdělení zadání mezi studenty je součástí tohoto dokumentu.
- S případnými dotazy kontaktujte svého cvičícího (studenti prezenční formy studia) či tutora (studenti kombinované formy studia).
- Termíny obhajob budou vypsány v systému Edison.
- Deadline je konečný a nebude dále posunován. Na projekty odevzdané po tomto datu nebude brán zřetel. Projekt lze pochopitelně odevzdat a domluvit si termín obhajoby i dříve.
- Každý cvičící (tutor) Vám sdělí, jakým způsobem budete projekt odevzdávat – pomocí git repozitáře, emailem, uložením na sdílené úložiště, systému Kelvin a tak podobně. Obecně platí, že se odevzdávají soubory se zdrojovým kódem, hlavičkové soubory, soubory s projektem atd., jinak řečeno vše, co je potřebné pro bezproblémovou kompilaci odevzdaného projektu a nic navíc.
- Součástí zdrojových kódů Vašeho programu bude programátorská dokumentace ve formě dokumentačních komentářů, zpracovatelných programem Doxygen, viz `www.doxygen.org`. Vygenerovanou dokumentaci není nutné odevzdávat. Postačuje, pokud Vámi odevzdaný archiv bude obsahovat konfigurační soubor `doxyfile`, případné adresáře pro vygenerovanou dokumentaci, vkládané obrázky atd.
- Ačkoliv se zadání mohou jevit na první pohled složitá, nepropadejte panice. Vyřešení kteréhokoliv zadání by nemělo zkušenému programátorovi zabrat více než „jeden večer“. Studentům prvního a druhého ročníku to zabere asi více času, ale v žádném případě by řešení nemělo trvat „desítky a desítky“ člověkohodin práce. Stejně tak co se týče délky vytvořeného kódu. Pokud jste už napsali „tisíce“ řádků kódu a stále není konec v dohledu, je to špatně. Takový zdrojový kód rovnou zahoďte. Klíčem k řešení je tužka, papír a hlava. Zkuste si řešení nejdříve promyslet, kreslit si u toho různá schémata, náčrtky datových struktur, volání funkcí a tak dále. Projděte si literaturu. A až se dostaví „aha moment“ tak začněte psát kód.

Hodnocení projektů

Kritéria hodnocení řešení projektů jsou tato:

1. **Správnost řešení** Správnost řešení je podmínka nezbytná. Aplikace, která nebude poskytovat správné výsledky, bude hodnocena automaticky 0 body bez ohledu na další kritéria. Ke každému zadání jsou k dispozici testovací data a výsledky, lze si tedy ověřit správnost řešení.
2. **Volba vhodných datových struktur** Toto kritérium hodnotí, v závislosti na konkrétním zadání, volbu vhodné datové struktury a algoritmů pro manipulaci se zvolenou datovou strukturou.
3. **Dekompozice problému na menší celky**
   - V předmětu Algoritmy I je požadována procedurální dekompozice, čili dekompozice řešení do funkcí. Využití objektově orientovaného programování je v tomto předmětu dobrovolné.
   - V předmětu Algoritmy II je požadován objektový návrh řešení a tomu odpovídající implementace.

4. **Způsob implementace** Toto kritérium hodnotí oddělení deklarace a definice funkcí nebo tříd do h a cpp souborů, využívání konstant místo přímo zapsaných hodnot, využívání parametrů funkcí a výsledků funkcí místo využívání side efektu založeném na globálních proměnných. Dále sem patří i úroveň zápisu zdrojového kódu, například odsazování vnořených konstrukcí, vhodné pojmenování proměnných, funkcí, tříd, dodržování zvolené konvence pojmenování[1] proměnných, funkcí a tříd, dodržování zvolené konvence psaní bloků kódu[2] a tak dále.

5. **Efektivita implementovaného algoritmu** Smyslem tohoto kritéria není nutit vás k implementaci nejlepšího známého algoritmu tím nejlepším možným způsobem. Tímto kritériem si vyučující ponechávají prostor pro případné snížení bodového hodnocení za použití algoritmu zcela nevhodného, nesmyslného, zmateného. *Příklad*: Součástí řešení projektu je třídění pole či vektoru s n prvky. Pokud použijete algoritmus se složitostí $O(n \log_2 n)$ je vše v pořádku. Pokud použijete některý z jednodušších algoritmů se složitostí $O(n^2)$, asi vás vyučující při obhajobě upozorní, že to nebyla dobrá volba, ale stále je to v pořádku. Za zcela nesmyslný algoritmus je v tomto případě považován algoritmus se složitostí větší než $O(n^2)$, protože takový algoritmus z podstaty věci dělá zbytečnou práci.

6. **Dokumentace k projektu** Ke každé funkci, třídě, atributu třídy musí existovat alespoň krátký dokumentační komentář ve formátu zpracovatelném programem Doxygen. Pro zápis dokumentačních komentářů lze využít libovolný z formátů, které Doxygen podporuje. Vygenerovanou dokumentaci není nutné odevzdávat, ale je nutné odevzdat konfigurační soubor `doxyfile`, aby bylo možné dokumentaci bez problémů vygenerovat.

   K programu Doxygen existuje rozsáhlá dokumentace volně dostupná na URL `https://www.doxygen.nl/manual/index.html`. Pro dokumentaci řešení semestrálního projektu je vhodné si prostudovat následující části dokumentace:

   - „Getting started“, `https://www.doxygen.nl/manual/starting.html`,
   - „Documenting the code“, `https://www.doxygen.nl/manual/docblocks.html`
   - „Doxygen usage“, `https://www.doxygen.nl/manual/doxygen_usage.html`
   - „Doxywizard usage“, `https://www.doxygen.nl/manual/doxywizard_usage.html`

     Program Doxywizard slouží k pohodlnému vygenerování konfiguračního souboru `doxyfile`, který tak není nutné vytvářet ručně.

7. **Citace zdrojů** Žádný program nevzniká úplně z ničeho, ani řešení semestrálních projektů. K řešení projektů můžete používat odbornou literaturu, učebnice, příklady zdrojových kódů z ostatních předmětů, internetové zdroje. V tom případě je nutné uvést, že „Tento algoritmus jsem převzal z...“, „Tuto část kódu jsem převzal z...“. Tyto případné citace musí umístit do dokumentačních komentářů. Asi není nutné citovat Levitinovu knihu, že jsem si tam „přečetl něco o průchodu grafem do šířky“ nebo, že „toto jsme řešili na cvičení“. Ostatní zdroje by se však citovat měly.

[1] Typicky camelCase, PascalCase, méně vhodná je už například maďarská notace.
[2] Typicky – složená levá závorka za příkazem nebo na novém řádku.

1 Slévání seznamů

Problém

V tomto zadání máte dáno k seznamů přirozených čísel. Všechny seznamy jsou neprázdné a prvky v nich jsou uspořádány vzestupně. Pro spojení dvou takových seznamů do jednoho vzestupně setříděného seznamu existuje velice jednoduchý a efektivní algoritmus, viz ukázkový příklad. Tento algoritmus se anglicky nazývá *merging algorithm* a česky bychom jej mohli nazvat jako algoritmus *spojování* či *slévání*[3]. Algoritmus slévání seznamů je například součástí třídícího algoritmu Mergesort, který česky nazýváme třídění sléváním. Bližší informace lze nalézt v [1, kapitola 5.1], [2, kapitola 2.3] nebo na Wikipedii [3].

Vaším úkolem je implementovat algoritmus slévání seznamů pro $k \geq 2$. Jinak řečeno vámi implementovaný algoritmus nebude slévat dohromady jen dva seznamy, ale bude slévat k seznamů *najednou*.

Ukázkový příklad

Předpokládejme, že máme dáno $k = 2$ a tedy dva vzestupně setříděné seznamy přirozených čísel $A_0 = (2, 3, 4, 5)$ a $A_1 = (1, 2, 4, 6, 7)$. Označme symbolem $h(A_i)$ první prvek, hlavu, seznamu A_i pro $i = 0, \ldots, k-1$. Proces slévání seznamů A_0 a A_1 do výsledného seznamu B můžeme popsat následovně:

A_0	A_1	Srovnání $h(A_i)$	B
$(2, 3, 4, 5)$	$(1, 2, 4, 6, 7)$	start	$()$
$(2, 3, 4, 5)$	$(1, 2, 4, 6, 7)$	$h(A_1) < h(A_0)$	(1)
$(2, 3, 4, 5)$	$(2, 4, 6, 7)$	$h(A_0) = h(A_1)$	$(1, 2)$
$(3, 4, 5)$	$(2, 4, 6, 7)$	$h(A_1) < h(A_0)$	$(1, 2, 2)$
$(3, 4, 5)$	$(4, 6, 7)$	$h(A_0) < h(A_1)$	$(1, 2, 2, 3)$
$(4, 5)$	$(4, 6, 7)$	$h(A_0) = h(A_1)$	$(1, 2, 2, 3, 4)$
(5)	$(4, 6, 7)$	$h(A_1) < h(A_0)$	$(1, 2, 2, 3, 4, 4)$
(5)	$(6, 7)$	$h(A_0) < h(A_1)$	$(1, 2, 2, 3, 4, 4, 5)$
$()$	$(6, 7)$	kopie zbytku A_1	$(1, 2, 2, 3, 4, 4, 5)$
$()$	$()$	konec	$(1, 2, 2, 3, 4, 4, 5, 6, 7)$

Podobně můžeme realizovat proces slévání tří seznamů.

A_0	A_1	A_2	Srovnání $h(A_i)$	B
$(2, 3, 4, 5)$	$(1, 2, 4, 6, 7)$	$(2, 3, 6)$	start	$()$
$(2, 3, 4, 5)$	$(1, 2, 4, 6, 7)$	$(2, 3, 6)$	$h(A_1) < h(A_0) = h(A_2)$	(1)
$(2, 3, 4, 5)$	$(2, 4, 6, 7)$	$(2, 3, 6)$	$h(A_0) = h(A_1) = h(A_2)$	$(1, 2)$
$(3, 4, 5)$	$(4, 6, 7)$	$(2, 3, 6)$	$h(A_2) < h(A_0) < h(A_1)$	$(1, 2, 2, 2)$
$(3, 4, 5)$	$(4, 6, 7)$	$(3, 6)$	$h(A_0) = h(A_2) < h(A_1)$	$(1, 2, 2, 2, 3)$
$(4, 5)$	$(4, 6, 7)$	$(3, 6)$	$h(A_2) < h(A_0) = h(A_1)$	$(1, 2, 2, 2, 3, 3)$
$(4, 5)$	$(4, 6, 7)$	(6)	$h(A_0) = h(A_1) < h(A_2)$	$(1, 2, 2, 2, 3, 3, 4)$
(5)	$(4, 6, 7)$	(6)	$h(A_1) < h(A_0) < h(A_2)$	$(1, 2, 2, 2, 3, 3, 4, 4)$
(5)	$(6, 7)$	(6)	$h(A_0) < h(A_1) = h(A_2)$	$(1, 2, 2, 2, 3, 3, 4, 4, 5)$
$()$	$(6, 7)$	(6)	$h(A_1) = h(A_2)$	$(1, 2, 2, 2, 3, 3, 4, 4, 5, 6)$
$()$	(7)	(6)	$h(A_2) = h(A_1)$	$(1, 2, 2, 2, 3, 3, 4, 4, 5, 6, 6)$
$()$	(7)	$()$	kopie zbytku A_1	$(1, 2, 2, 2, 3, 3, 4, 4, 5, 6, 6, 7)$

[3] Ve smyslu slévání, smíchání, dvou tekutin dohromady, nikoliv ve smyslu slévání kovu do formy.

Poznamenejme, že v okamžiku, kdy výběr minima z hlav seznamů není jednoznačný, to jest, kdy několik hlav má stejnou minimální hodnotu, můžeme do výsledného seznamu zařadit kteroukoliv z těchto hlav. Seznam ze kterého hlavu odebereme, můžeme zvolit zcela náhodně, můžeme zvolit například seznam s nejnižším indexem, ale také třeba seznam ve kterém je nejméně prvků nebo naopak nejvíce prvků. A v případě, kdy v procesu slévání zůstane už jen jediný seznam, lze zbylé prvky z tohoto seznamu rovnou vložit do výsledného seznamu. Důvod je zřejmý.

Poznámky

- Vstupní seznamy čísel jsou uloženy v textových souborech. V každém souboru je právě jeden seznam. Prvky seznamu jsou vždy uloženy na samostatném řádku. Například seznam A_0 z ukázkového příkladu bude uložen takto:

  ```
  2
  3
  4
  5
  ```

- Celkem je k dispozici 100 testovacích seznamů. Seznamy jsou uloženy v textových souborech `0.txt` až `99.txt`. Předpokládejte, že hodnota k bude vždy v rozsahu 2 až 100 a pro dané k se budou slévat seznamy obsažené v souborech označených čísly 0 až $k - 1$.
- Výsledný seznam vypište do textového souboru ve stejném formátu jako vstupní seznamy.
- Vstupem do aplikace budou tři hodnoty – k, adresář s testovacími soubory a jméno výstupního textového souboru.
- Kontrolu správnosti implementovaného algoritmu můžete provést velice jednoduše. Všechny slévané seznamy vložíte do jednoho pole, či instance třídy `vector` a setřídíte běžným třídícím algoritmem, klidně můžete použít algoritmus ze standardní knihovny. Výsledek musí odpovídat výsledku slévacího algoritmu.
- Požaduje se implementace algoritmu, který slévá daných k seznamů najednou. Řešení, při kterém jsou opakovaně slévány dva seznamy není akceptovatelné.

2 Scheduler procesoru

Problém

Moderní SoC (System-on-a-chip) založené na architektuře ARM mají typicky tři druhy jader, která jsou různě výkonná:

- Cortex-X2 s vysokým výkonem,
- Cortex-A710 se středním výkonem a
- Cortex-A510 s nízkým výkonem.

Náš hypotetický SoC má tři jádra. Jedno jádro je vysoce výkonné, jedno středně a jedno nízko výkonné jádro.

Každé z těchto jader je vhodné pro různě náročné, tedy typově rozdílné, výpočetní úlohy. Uvažujme, že výpočetní čas úlohy závisí rovněž na typu jádra, na kterém výpočet probíhá. Nejde totiž jen o výkon, ale také o režijní náklady při volbě nevhodného typu jádra. Rozlišujeme tři typy úloh:

- Náročná (H) – trvá 1 jednotku času na vysoce výkonném jádru (Cortex-X2), 2 jednotky na ostatních,
- Středně náročná (M) – trvá 1 jednotku času na středně výkonném jádru (Cortex-A710), 2 jednotky na ostatních,
- Nenáročná (L) – trvá 1 jednotku času na jádru s nízkým výkonem (Cortex-A510), 2 jednotky na ostatních.

Úlohy se zpracovávají po dávkách (tzv. batch), kde každá dávka obsahuje 8 úloh. V každém kroku je dávka 8 úloh sestavena, následně rozdělena mezi jádra a zpracována. Po zpracování všech úloh z dané dávky se pokračuje analogicky s další dávkou 8 úloh. Každé jádro má frontu úloh, které má zpracovat, zpracování probíhá paralelně. Dávka úloh je považována za zpracovanou po dokončení všech úloh z dané dávky.

Implementujte funkce, resp. metody tříd, které umožní provést následující dvojici simulací.

- V první simulaci jsou úlohy z dávek rozděleny systémem round-robin[4]. To znamená, že první úloha je přiřazena vysoce výkonnému jádru, druhá středně výkonnému, třetí jádru s nízkým výkonem, čtvrtá vysoce výkonnému, pátá středně výkonnému a tak dále, až do rozdělení celé dávky. Úlohy jsou tedy rozděleny bez ohledu na jejich typ a vhodnost jádra. Vašim úkolem je provést simulaci tohoto scénáře a zjistit, kolik jednotek času celá simulace zabrala.
- Ve druhé simulaci je vaším úkolem navrhnout systém, dle kterého budou úlohy z dávky rozděleny mezi jádra lepším způsobem, to jest tak, aby došlo ke snížení času nutného pro zpracování dávky úloh[5]. Simulaci proveďte a zjistěte, kolik jednotek času celá simulace zabrala. Porovnejte čas s první simulací.

Ukázkový příklad

Na obrázku 1 vidíme ukázku rozdělení úloh pomocí systému round-robin a první 3 kroky zpracování. Všimněte si, že v případě přiřazení úlohy k nevhodnému jádru se položka ve frontě zdvojuje, čímž je simulován dvojnásobný čas zpracování. Zpracování této dávky zabere 5 jednotek času. Na obrázku 2 vidíme rovněž příklad lepšího rozdělení úloh, kdy zpracování dávky úloh zabere pouze 3 jednotky času.

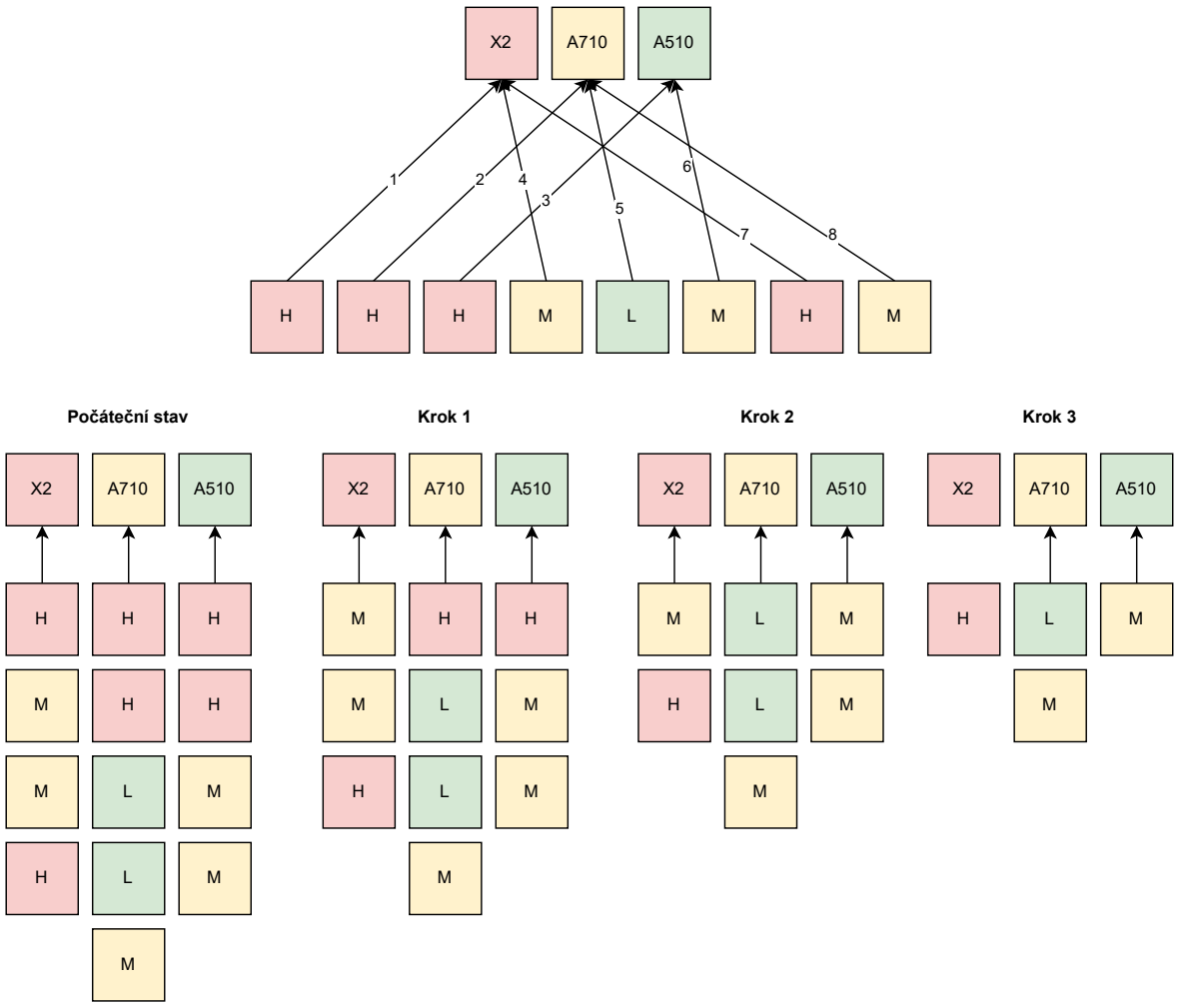
Poznámky

- Vstupem je textový soubor. Na prvním řádku je uveden celkový počet úloh v souboru.
- Na každém následujícím řádku je pouze jeden znak, který označuje typ úlohy (H, M nebo L).
- Vstupní soubor s jednou dávkou úloh může vypadat například následovně:

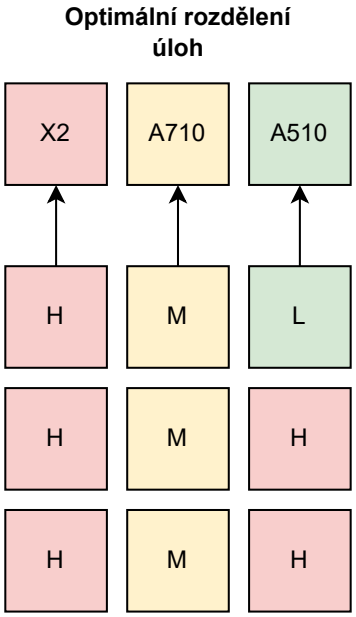
```
8
H
H
H
M
L
M
H
M
```

[4] V češtině bychom jej mohli nazvat „chodí pešek okolo“.

[5] Samozřejmě, v některých případech může být rozdělení úloh systémem round-robin nejlepší možné. Ale většinou tomu tak nebude.



Obrázek 1: Diagram popisující simulaci rozdělení úloh systémem round-robin



Obrázek 2: Diagram s ukázkou lepšího rozdělení

- Výstupy simulací vypište ve vhodném tvaru na standardní výstup.

3 Transformace orientovaného grafu na orientovaný acyklický graf

Problém

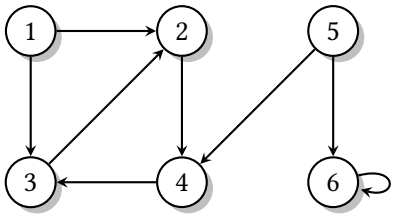
V tomto zadání máte za úkol zpracovat data z české mutace Wikipedie. Ze surových dat byl vytvořen orientovaný graf $G = (V, E)$, kde G je množina vrcholů a E je množina hran. Vrcholy grafu jsou označeny celočíselnými identifikátory většími než nula. Vrcholy grafu odpovídají jednotlivým stránkám Wikipedie a hrany grafu reprezentují odkazy mezi stránkami[6]. Vaším úkolem je:

- navrhnout a implementovat vhodnou reprezentaci grafu G v operační paměti,
- implementovat funkci pro načtení grafu G z textového souboru,
- implementovat funkci, která na standardní výstup vypíše počet vrcholů a počet hran grafu G a
- implementovat funkci, která bude vracet **true**, pokud graf G je acyklický. Jinak bude vracet **false**.
- A dále implementujte funkci, která transformuje původní graf G na acyklický graf G'. Toho můžete dosáhnout například tak, že při průchodu grafem G do hloubky odstraníte všechny zpětné hrany.

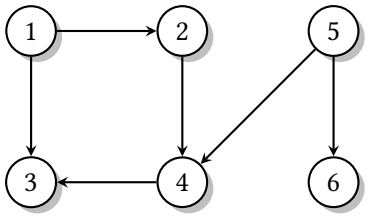
Vytvořená aplikace:

1. nejprve načte graf G do reprezentace operační paměti a následně
2. na standardní výstup vypíše počet vrcholů a počet hran grafu G a informaci o tom, zda je graf G acyklický nebo nikoliv.

[6] Jen pro kontrolu, graf G obsahuje 2 673 524 vrcholů a 40 769 236 hran.



(a) původní graf G_0



(b) acyklický graf G'_0

Obrázek 3: Ukázkový graf

3. Dále provede transformaci grafu G na acyklický graf G' a
4. na závěr pro kontrolu vypíše o grafu G' stejné informace jako o grafu G.

Ukázkový příklad

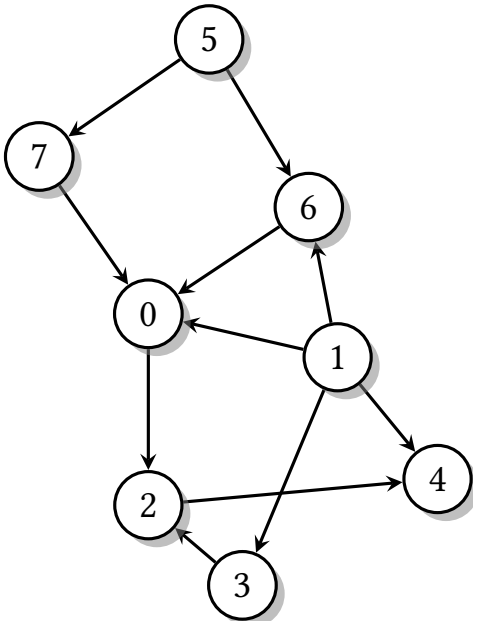
Předpokládejme, pro jednoduchost, graf G_0 na obrázku 3a. Na sousedním obrázku 3b je zobrazen jeden z možných acyklických grafů G'_0. Z obrázku je patrné, že je nutno odstranit smyčku u vrcholu 6 a dále je nutné „přetnout“ cyklus mezi vrcholy 2, 4 a 3, odstraněním jedné ze tří hran tohoto cyklu. Je lhostejno kterou, proto není řešení jednoznačné. Ale v každém případě vznikne acyklický graf. Ukázky dalších acyklických grafů jsou uvedeny na obrázku 4.

Poznámky

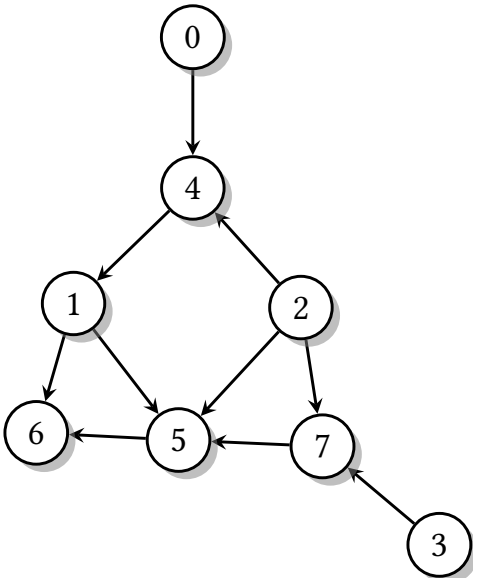
- Graf je uložen v textovém souboru ve formě seznamu hran. Každá hrana je uložena na samostatném řádku jako dvojice kladných celých čísel představující počáteční vrchol a koncový vrchol hrany. Číselné identifikátory obou vrcholů jsou odděleny bílými znaky. Ukázkový graf G_0 na obrázku 3 bude uložen například takto:

```
1       2
3       2
4       3
5       4
6       6
2       4
1       3
5       6
```

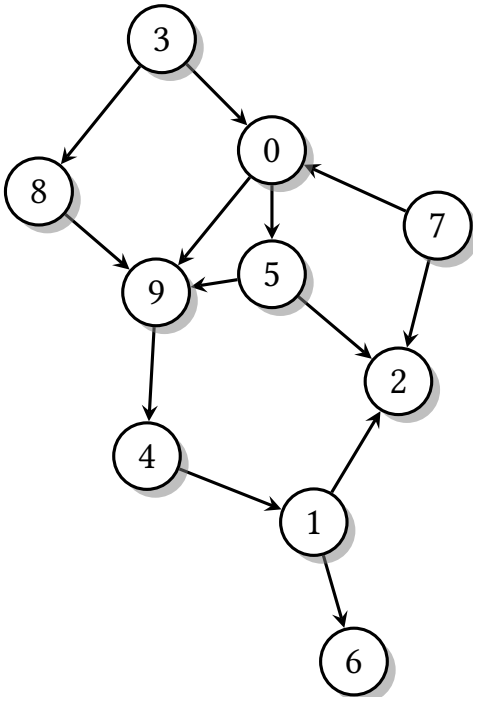
- Než se pustíte do programování doporučuji si pozorně přečíst kapitoly 1.4, 3.5 a 4.2 z Levitinovy knihy [1]. Stejně tak kapitolu 20 z knihy [2]. Množství informací o grafech najdete v českém učebním textu docenta Kováře z naší univerzity [4]. Dále by mohl být užitečný i článek na anglické Wikipedii, https://en.wikipedia.org/wiki/Topological_sorting.
- Dále je nutné upozornit, že využití jakéhokoliv rekurzivního algoritmu, který bude rekurzivně procházet všechny vrcholy nebo hrany grafu, je předem odsouzeno k nezdaru. Vrcholů v grafu české Wikipedie je už takové množství, že to systémový zásobník sloužící k jinak perfektnímu fungování rekurzivních funkcí nezvládne. Proto je nutné takové algoritmy implementovat *nerekurzivně*.
- Vzhledem k množství dat, které je nutno zpracovat, bude vhodné využít i jiné datové struktury než jen `vector`, případně pole.
- Transformace grafu na acyklický graf tak, že z původního grafu jsou odstraněny všechny hrany, je považována za nepřípustné řešení zadání.



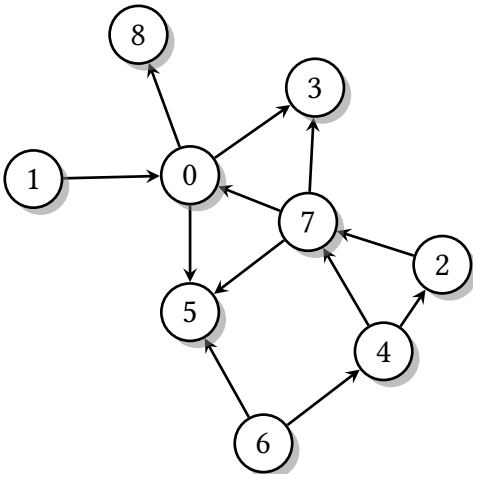
(a) acyklický graf G'_1



(b) acyklický graf G'_2



(c) acyklický graf G'_3



(d) acyklický graf G'_4

Obrázek 4: Ukázky acyklických grafů

4 Četnosti vzdáleností v grafu

Problém

V tomto zadání máte za úkol zpracovat data z české mutace Wikipedie. Ze surových dat byl vytvořen orientovaný graf $G = (V, E)$, kde G je množina vrcholů a E je množina hran. Vrcholy grafu jsou označeny celočíselnými identifikátory většími než nula. Vrcholy grafu odpovídají jednotlivým stránkám Wikipedie a hrany grafu reprezentují odkazy mezi stránkami[7]. Vaším úkolem je:

1. navrhnout a implementovat vhodnou reprezentaci grafu G v operační paměti,
2. implementovat funkci pro načtení grafu G z textového souboru a
3. implementovat funkci pro výpočet vzdáleností všech dosažitelných vrcholů grafu G od daného počátečního, startovacího, vrcholu s. Vzdálenost $d(u, v)$ dvojice vrcholů u, v grafu G definujeme jako počet hran na nejkratší orientované cestě z vrcholu u do vrcholu v[8]. Pokud taková cesta neexistuje, definujeme $d(u, v) = \infty$ a říkáme, že vrchol v je z vrcholu u nedosažitelný.
4. Vzdálenosti vypočtěte pro tyto počáteční vrcholy s:
   - stránku VŠB-TU Ostrava, $s = 22197$,
   - stránku Univerzity Karlovy v Praze, $s = 1083$.
   - Na tomto místě můžete namítnout, že mám jít s celým zadáním projektu už do Prčic a že si jdete dát pivo. Takže další dva počáteční vrcholy budou: stránka obce Prčice, $s = 122606$ a stránka o pivu $s = 6916$.
5. Z takto vypočtených vzdáleností, pro každý počáteční vrchol s, sestavte tabulku jejich četností. To jest tabulku, která bude udávat kolikrát se určitá vzdálenost ve všech vypočtených vzdálenostech, se stejným vrcholem s, vyskytuje. Výsledek vypište do textového souboru.

Ukázkový příklad

Předpokládejme, pro jednoduchost, graf G_0 na obrázku 5. Jako počáteční vrchol zvolíme vrchol $s = 2$. Vzdálenosti všech vrcholů grafu G_0 od vrcholu s a odpovídající četnosti vzdáleností jsou uvedeny v tabulce 1. Poznamenejme, že v tomto grafu jsou všechny vrcholy z vrcholu s dosažitelné.

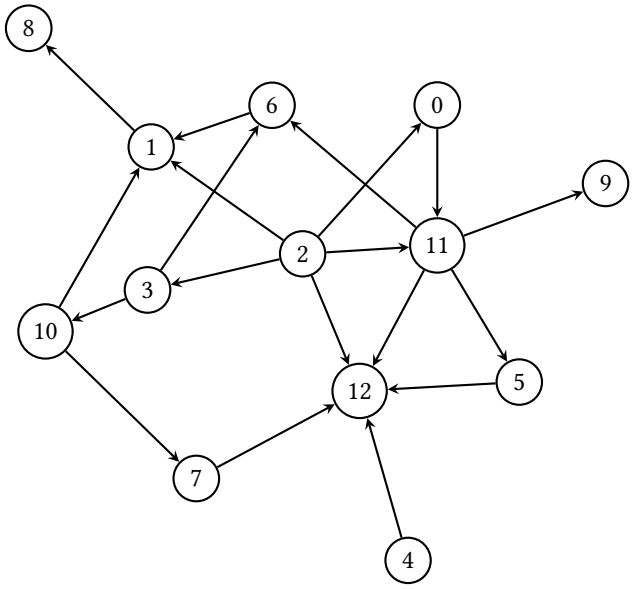
Poznámky

- Graf je uložen v textovém souboru ve formě seznamu hran. Každá hrana je uložena na samostatném řádku jako dvojice kladných celých čísel představující počáteční vrchol a koncový vrchol hrany. Číselné identifikátory obou vrcholů jsou odděleny bílými znaky. Ukázkový graf G_0 na obrázku 5 bude uložen například takto:

```
11 5
11 9
2 3
2 12
2 1
5 12
6 1
```

[7] Jen pro kontrolu, graf G obsahuje 2 673 524 vrcholů a 40 769 236 hran.

[8] Pozor, graf G je orientovaný, orientovaná cesta z vrcholu u do vrcholu v nemusí být shodná z orientovanou cestou z v do u nebo orientovaná cesta z v do u nemusí vůbec existovat.



Obrázek 5: Ukázkový graf G_0

v	$d(s, v)$
0	1
1	1
2	0
3	1
5	2
6	2
7	3
8	2
9	2
10	2
11	1
12	1

(a) Vzdálenosti vrcholů od počátku $s = 2$

Vzdálenost d	Četnost
0	1
1	5
2	5
3	1

(b) Tabulka četností

Tabulka 1: Výsledky pro ukázkový graf G_0

```
10  7
2  11
3  6
10  1
11  12
1  8
7  12
3  10
0  11
11  6
2  0
4  12
```

- Dále je nutné upozornit, že využití jakéhokoliv rekurzivního algoritmu, který bude rekurzivně procházet všechny vrcholy nebo hrany grafu, je předem odsouzeno k nezdaru. Vrcholů v grafu české Wikipedie je už takové množství, že to systémový zásobník sloužící k jinak perfektnímu fungování rekurzivních funkcí nezvládne. Proto je nutné takové algoritmy implementovat *nerekurzivně*.
- Vzhledem k množství dat, které je nutno zpracovat, bude vhodné využít i jiné datové struktury než jen `vector`, případně pole.

5 Simulace výtahu ve výrobě

Problém

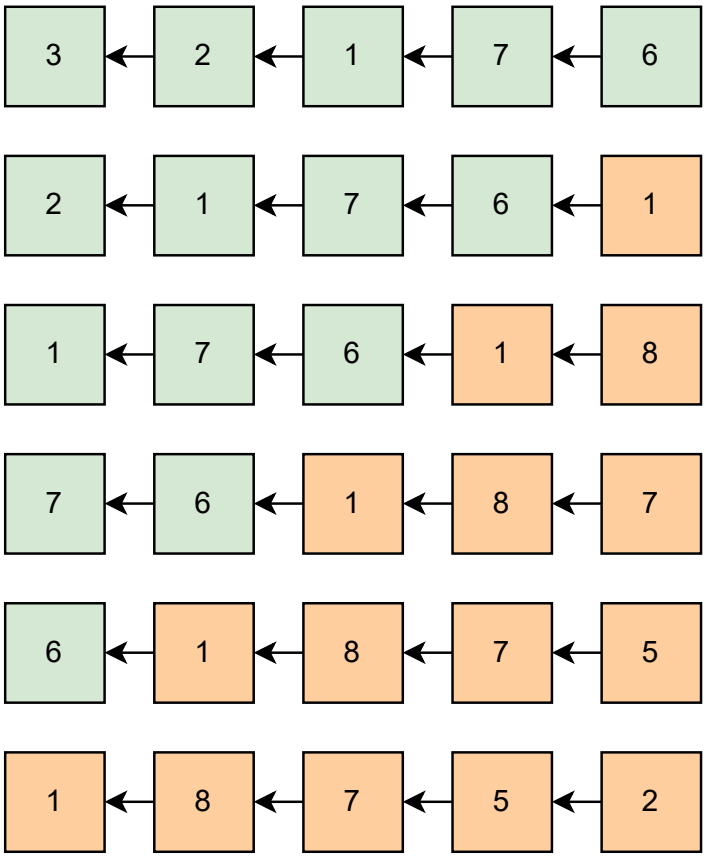
Vaším úkolem je simulace a optimalizace části výrobního procesu v průmyslové výrobě. Výrobní linka je rozdělena do několika částí, které jsou umístěny v různých patrech výrobní haly. Dopravu do těchto pater obstarává malý výtah obsluhovaný robotem. Robot vkládá součástky do výtahu v přízemí a součástky následně výtahem putují do jednoho z osmi pater[9]. Každá součástka je označena číslem 1 až 8, které určuje, do kterého patra musí být dopravena. Výtah je schopen najednou přepravit maximálně tři součástky. Výtah je schopen jet pouze z přízemí do vybraného patra a poté se vrátit, nelze tedy dělat „zastávky“. Přesun výtahu z jednoho patra do druhého považujeme za jednotku času, bez ohledu na to, kolik součástek veze. Pokud výtah pojede např. do čtvrtého patra bude doba zpracování 8 časových jednotek, protože 4 časové jednotky jede výtah do čtvrtého patra a 4 časové jednotky se vrací[10]. Robot má k dispozici odkládací prostor, který pojme maximálně 10 součástek. Se součástkami v odkládacím prostoru může libovolně manipulovat – různě je třídit, vybírat součástky podle svého uvážení atd. Odkládací prostor je, do vyčerpání všech součástek, průběžně doplňován. Doplňování probíhá ale vždy až po naložení výtahu.

Implementujte funkce, resp. metody třídy, které umožní provést následující dvojici simulací.

- V první simulaci probíhá doprava triviálním způsobem. Součástky jsou vždy dopravovány pouze po jedné. Robot součástky nakládá v tom pořadí, v jakém jsou doplňovány do odkládacího prostoru. Zjistěte celkovou dobu přepravy všech součástek.
- Ve druhé simulaci je vaším úkolem systém dopravy zlepšit tak, aby se snížil celkový čas dopravy všech součástek. Proveďte simulaci a zjistěte, kolik jednotek času simulace zabrala. Porovnejte čas s první simulací. Je jasné, že zlepšení bude spočívat v lepším využití kapacity výtahu a v lepší organizaci součástek v odkládacím prostoru.

[9] Patra jsou číslována od 1 do 8, 0 označuje přízemí.

[10] Doba pro nakládání a vykládání součástek se neuvažuje.



Obrázek 6: Diagram s ukázkou procesu dopravy pomocí triviálního postupu.

Ukázkový příklad

Na obrázku 6 vidíme ukázku procesu dopravy pomocí triviálního postupu. Zelenou barvou jsou vyznačeny původní součástky v odkládacím prostoru, oranžovou barvou jsou vyznačeny postupně doplňované součástky. V každém kroku je jedna součástka odstraněna a jedna doplněna. Tímto způsobem proces funguje až do vyčerpání všech součástek, které mají být přepraveny.

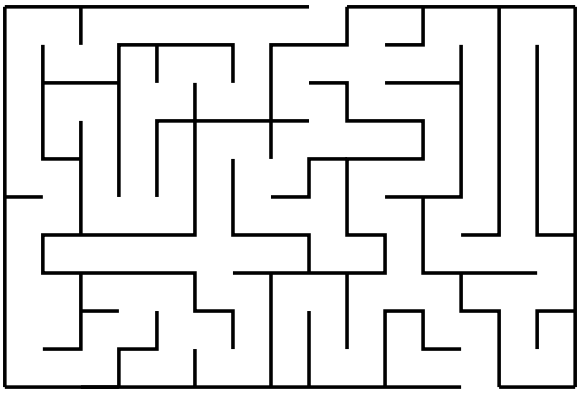
Čas pro zpracování všech zelených součástek na obrázku 6 spočítáme následující rovnice

$$t_{green} = 2 \cdot (3 + 2 + 1 + 7 + 6) = 2 \cdot 19 = 38. \tag{1}$$

Poznámky

- Vstupem je textový soubor. Na prvním řádku je uveden celkový počet součástek k dopravě.
- Na každém následujícím řádku je pouze jedno celé číslo $p \in \langle 1, 8 \rangle$, které označuje patro, kam je nutné součástku dopravit.
- Vstupní soubor pro 10 součástek vypadá například takto

```
10
3
2
1
7
6
```



Obrázek 7: Ukázkové bludiště

1
8
7
5
2

6 Generování bludiště

Problém

V tomto zadání máte za úkol vygenerovat bludiště. Algoritmus generování bludiště je randomizovanou verzí průchodu grafem do hloubky.

Na začátku máte dánu mřížku čtvercových buněk[11], přičemž:

- každá buňka představuje vrchol grafu,
- každá buňka je spojena hranou se čtyřmi sousedními buňkami, s buňkou nad, pod vlevo a vpravo od dané buňky, pokud tyto buňky existují a
- každá buňka je na začátku ohraničena čtyřmi stěnami.

Tímto způsobem vznikne graf, který projdeme známým algoritmem do hloubky. Na rozdíl od běžného algoritmu sousední, dosud nenavštívené vrcholy grafu, bereme v náhodném pořadí. Proto mluvíme o randomizovaném algoritmu – sousední buňky nejsou vybírány podle nějakého pravidla, například po směru pohybu hodinových ručiček, ale zcela náhodně. Další rozdíl spočívá v tom, že pokud projdeme hranou z jednoho vrcholu do druhého, tak mezi odpovídajícími buňkami odstraníme stěnu.

Ukázkový příklad

Výsledkem aplikace může být například bludiště na obrázku 7. Bludiště se pochopitelně může při každém spuštění aplikace vygenerovat jiné.

Poznámky

- Mřížka bude mít n řádků a m sloupců. Tyto hodnoty zadejte jako argumenty příkazového řádku.

[11] Pro názornost si představme čtverečkovaný papír.

- Třetím argumentem na příkazovém řáku bude počáteční hodnota generátoru náhodných čísel. Počáteční hodnota generátoru se nastavuje pomocí funkce std :: srand, viz https://en.cppreference.com/w/cpp/numeric/random/srand. Tento argument nemusí být na příkazovém řádku uveden. V tomto případě se počáteční hodnota generátoru náhodných čísel ponechá na výchozí hodnotě, funkce std :: srand se nebude volat.
- Vygenerované bludiště uložte do souboru ve formátu SVG. Krátký kurz o formátu SVG naleznete na URL https://www.w3schools.com/graphics/svg_intro.asp.
- Bludiště musí mít jeden vchod a jeden východ, které nejsou navzájem totožné. Vchod a východ vygenerujte vždy na protilehlých stranách bludiště.

Odkazy

1. LEVITIN, Anany. *Introduction to the Design and Analysis of Algorithms.* 3rd ed. Boston: Pearson, 2012. ISBN 978-0-13-231681-1.
2. CORMEN, Thomas H.; LEISERSON, Charles Eric; RIVEST, Ronald L.; STEIN, Clifford. *Introduction to algorithms.* Fourth edition. Cambridge, Massachusetts: The MIT Press, 2022. ISBN 978-026--2046-305.
3. *Merge algorithm* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2025-03-17]. Dostupné z: https://en.wikipedia.org/wiki/Merge_algorithm.
4. KOVÁŘ, Petr. *Teorie grafů.* Ostrava, 2022. Dostupné také z: https://mi21.vsb.cz/modul/teorie-grafu.

Rozdělení zadání mezi studenty

	Login	Příjmení jméno	Zadaný projekt
1	ABI0008	Abilmazhinova Aruzhan	1
2	ADI0010	Adilgali Assanali	2
3	AMA0023	Aman Ataniyaz	3
4	AMB0041	Ambrož Daniel	4
5	ANT0100	Anton Štěpán	5
6	ANT0101	Antoš Matěj	6
7	BAL0305	Balhar Jaroslav	1
8	BAL0330	Balus David	2
9	BAN0070	Bant Tomáš	3
10	BAR0853	Bartusek Matěj	4
11	BAR0866	Bartoš Ámos	5
12	BAS0083	Basel Filip	6
13	BEC0075	Bečka Daniel	1
14	BEL0178	Belyayeva Yelizaveta	2
15	BEN0339	Beneš Zdeněk	3
16	BEN0341	Beneš Václav	4
17	BER0307	Berezovský Peter	5
18	BER0328	Bernaťák Václav	6
19	BER0330	Bernatík Richard	1
20	BEU0005	Beunier Martin	2
21	BIA0033	Biagini Veronica	3
22	BIE0069	Bierhanzlová Lucie	4
23	BIL0201	Bílý Martin	5
24	BJA0009	Bjaček Bernard	6
25	BLA0384	Bláha Jakub	1
26	BLA0385	Blahuta Vojtěch	2
27	BOD0045	Bodnar Vladyslav	3
28	BOR0219	Borisov Anton	4
29	BOS0042	Bošanský Erik	5

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
30	BRA0241	Brázda Pavel	6
31	BRU0098	Brudovský Andreas	1
32	BUS0054	Buš Jan	2
33	CAN0054	Čanecký Lukáš	3
34	CEC0166	Čečotka Tomáš	4
35	CER0619	Černý Radek Václav	5
36	CIE0138	Cieslar Jiří	6
37	CIG0041	Cigánek Jan	1
38	CVI0013	Cvik Lukáš	2
39	CVI0014	Cvikl Adam	3
40	CVI0016	Čvirik Samuel	4
41	DAB0010	Dabrowski Adam	5
42	DAN0208	Danihel Lukáš	6
43	DAN0209	Daník Šimon	1
44	DEL0063	Delong David	2
45	DLA0021	Dlabola Maxim	3
46	DLO0026	Dlouhý Michal	4
47	DOB0171	Dobiášovský Michal	5
48	DOD0012	Do Duc Trung	6
49	DOM0072	Domitra Ondřej	1
50	DOS0232	Dostal Josef	2
51	DOS0233	Dostálík Adam	3
52	DRA0180	Dráber Martin	4
53	DRA0191	Drabina Matej	5
54	DRO0115	Drössler Lukáš	6
55	DRO0116	Drobyna Oleksandr	1
56	DUB0093	Dubovský Adrián	2
57	DZI0038	Dzifčák Miroslav	3
58	ERB0015	Erben Roman	4
59	FAB0083	Fabisz Samuel	5
60	FEH0012	Fehér Timur Artur	6

pokračování na další stránce...

	Login	Příjmení jméno	Zadaný projekt
61	FEI0025	Feichtinger Jakub	1
62	FIT0007	Fitřík Jan	2
63	FOJ0149	Fojtík Jakub	3
64	FOR0088	Formánek Filip	4
65	FRA0216	Frank Filip	5
66	FRA0219	Frank Robert	6
67	GAL0206	Galia Jakub	1
68	GAL0207	Galica Jakub	2
69	GAL0218	Galčík Juraj	3
70	GAN0051	Gánovský Peter	4
71	GAN0052	Gaňa Sebastián	5
72	GAR0165	Gardoš Filip	6
73	GAV0059	Gavelčík Matěj	1
74	GEI0005	Geiler Ilja	2
75	GER0097	Geryk Ondřej	3
76	GLU0025	Glumbík Jakub	4
77	GOG0016	Gogolev Artemii	5
78	GOL0116	Gold Matěj	6
79	GRA0126	Graf Erik	1
80	GRI0073	Grichová Adéla	2
81	GRY0115	Grygarčík Tomáš	3
82	GRY0122	Grygar Ondřej	4
83	GRY0125	Grygarčík Ondřej	5
84	GUZ0034	Guziur Ondřej	6
85	HAD0067	Haderka Martin	1
86	HAL0266	Halama Marek	2
87	HAL0268	Halfar Matej	3
88	HAL0269	Halfar Vojtěch	4
89	HAM0133	Hamran Martin	5
90	HAN0412	Hanusek Adam	6
91	HAN0413	Hanskyi Nikita	1

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
92	HAN0416	Hanzlík Adam	2
93	HAR0199	Harok Antonín	3
94	HAR0202	Hartiník Tomáš	4
95	HAR0203	Hartmann Kryštof	5
96	HAR0205	Haraimovych Yaroslav	6
97	HEB0015	Hebda David	1
98	HEI0081	Heider Filip	2
99	HEJ0094	Hejtman Matěj	3
100	HEN0083	Hendrichová Nikol	4
101	HLI0033	Hliva Matyáš David	5
102	HLU0053	Hlubina Martin	6
103	HOK0012	Hok Vojtěch	1
104	HOL0577	Holínka David	2
105	HOL0601	Holub Martin	3
106	HOM0085	Homolka Vít	4
107	HOS0126	Hosnédl Karel	5
108	HOS0127	Hostiev Oleh	6
109	HRA0312	Hradil Marek	1
110	HRC0018	Hrček Adam	2
111	HRN0040	Hrňa Lukáš	3
112	HRO0111	Hrozová Julie	4
113	HRO0117	Hroch Petr	5
114	HRU0295	Hruška Daniel	6
115	HRU0299	Hruboš Michal	1
116	HUD0160	Huďa Matěj	2
117	HUR0091	Hurin Tymur	3
118	HYN0045	Hýnar Martin	4
119	CHA0264	Chaňo Viktor	5
120	CHA0265	Chabroň Denis Kristián	6
121	CHA0270	Charbulák Lukáš	1
122	CHE0084	Chehil Yurii	2

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
123	CHE0086	Chervinskyi Nikita	3
124	CHM0086	Chmelík Mojmír	4
125	CHO0281	Cholak Maksym	5
126	CHO0288	Chobot Marek	6
127	CHO0289	Chovanec Richard	1
128	CHR0202	Chrascina Dominik	2
129	CHV0053	Chvostková Veronika	3
130	CHY0105	Chytil Jakub	4
131	CHY0107	Chynarbekov Samarbek	5
132	IND0039	Indrák Jan	6
133	IND0045	Indruch Jakub	1
134	JAC0038	Jacková Adriána	2
135	JAC0039	Jáchym Václav	3
136	JAN0992	Janíček Adam	4
137	JAN0993	Jančík Vojtěch	5
138	JAN0994	Janusz Oliver	6
139	JAR0193	Jarabica Martin	1
140	JEL0103	Jelínek Patrik	2
141	JEZ0107	Ježik Dávid	3
142	JOC0037	Jochymková Kateřina	4
143	JUR0543	Jurošků Ashley	5
144	JUR0544	Jurica Tomáš	6
145	JUR0545	Jurky Michal	1
146	JUR0546	Jurčaga Tomáš	2
147	JUR0547	Jurásek Štěpán	3
148	JUR0552	Juřík Petr	4
149	KAC0103	Káčerik Michal	5
150	KAD0222	Kadlčík Jiří František	6
151	KAL0324	Kaldybayeva Assel	1
152	KAL0339	Kalabiha Myroslav	2
153	KAL0362	Kaleta Jakub	3

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
154	KAN0268	Kaňa Jakub	4
155	KAN0310	Kantor Robin	5
156	KAN0312	Kantor Šimon	6
157	KAN0313	Kaňák Ondřej	1
158	KAN0314	Kaňok Daniel	2
159	KAN0316	Kanovský David	3
160	KAN0321	Kandiushyn Dmytro	4
161	KAP0122	Kapitulčin Adam	5
162	KAR0321	Karpjuk René	6
163	KAR0323	Kareš Pavel	1
164	KEM0041	Kempný Lukáš	2
165	KHA0047	Khamzina Shakira	3
166	KHO0028	Khomulenko Ilya	4
167	KIJ0017	Kijonka Adam	5
168	KIK0015	Kiktova Olena	6
169	KIR0045	Kirschner Damián	1
170	KLA0136	Klapetek Jakub	2
171	KLA0139	Klanicová Hana	3
172	KLI0302	Klimíček Ondřej	4
173	KLO0088	Klozík Tomáš	5
174	KNA0060	Knapek Jakub	6
175	KNE0061	Knettig Vojtěch	1
176	KOK0068	Kokeš Radim	2
177	KOL0563	Kolařík Mikoláš	3
178	KOL0602	Koliba Marek	4
179	KOL0607	Kolmačka Ondřej	5
180	KOL0620	Kollár David	6
181	KOL0629	Kolomazník Jiří	1
182	KOM0137	Koman Ondřej	2
183	KON0481	Konovalov Viktor	3
184	KON0524	König Petr	4

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
185	KON0530	Kondratov Vladyslav	5
186	KOR0302	Kornuta Kristián	6
187	KOR0342	Kordula Jan	1
188	KOS0405	Košíček Marek	2
189	KOS0431	Košař Adam	3
190	KOS0434	Košař Jáchym	4
191	KOT0325	Kotyra Adam	5
192	KOT0326	Kotrba Matěj	6
193	KOV0426	Koval Jakub	1
194	KOZ0380	Kožiarik Marko	2
195	KOZ0381	Kožák Pavol	3
196	KOZ0382	Kozel Marek	4
197	KOZ0384	Kozel Ladislav	5
198	KRA0741	Kramář Rostislav	6
199	KRA0745	Kratoš Samuel	1
200	KRA0748	Kramný František	2
201	KRC0096	Krčmář Radomír	3
202	KRE0463	Kreutz Adam	4
203	KUB0760	Kubík Martin	5
204	KUB0809	Kubesa Jakub	6
205	KUB0812	Kubica Adam	1
206	KUB0813	Kubát Tomáš	2
207	KUC0436	Kučera Ondřej	3
208	KUC0437	Kučera David	4
209	KUC0438	Kučo Samuel	5
210	KUD0142	Kudela Alexander	6
211	KUD0147	Kudla Patrik	1
212	KUD0148	Kudělka Vojtěch	2
213	KUL0168	Kulštejn Ondřej	3
214	KUN0183	Kunštár Jan	4
215	KUP0144	Kupar Bohdan	5

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
216	KYR0017	Kyralová Zita	6
217	LAT0054	Látal Lukáš	1
218	LES0076	Leška Tomáš	2
219	LIC0113	Lichý Petr	3
220	LIT0041	Litvan Jan	4
221	LOM0016	Lomaka Kyrylo	5
222	LUB0039	Lubij Roman	6
223	LUD0046	Ludvíček Šimon	1
224	LUK0183	Lukashenko Artem	2
225	LUK0191	Lukeš Jan	3
226	LYS0058	Lysenko Pavlo	4
227	MAC0689	Maceček Patrik	5
228	MAC0693	Macura Denis	6
229	MAC0694	Macura Jakub	1
230	MAJ0187	Majvelder Filip	2
231	MAK0090	Makhsutbekov Danial	3
232	MAL0486	Maléř Jan	4
233	MAR0991	Marvan Vojtěch	5
234	MAR1021	Marek Šimon	6
235	MAR1025	Mařík Michal	1
236	MAS0198	Maslaňak Adam	2
237	MAT0495	Matula Miroslav	3
238	MAT0514	Matsyfuk Artem	4
239	MAT0544	Matýsek Jiří	5
240	MAT0563	Matějů Ondřej	6
241	MAT0586	Matyskiewicz Marek	1
242	MAX0019	Maximova Veronika	2
243	MAY0030	Mayer Dominik	3
244	MEI0035	Meinhard Michael	4
245	MEN0116	Menzarev Oleksandr	5
246	MIC0548	Michna Matúš	6

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
247	MIH0059	Mihok Daniel	1
248	MIK0537	Mik Ondřej	2
249	MIK0538	Mikulčík Štěpán	3
250	MIS0104	Misař Viliam	4
251	MIT0114	Mitura Karel	5
252	MLC0053	Mlčák Jan	6
253	MOR0276	Moravec Daniel	1
254	MOT0116	Motyčka Jaromír	2
255	MOZ0030	Mozol Maroš	3
256	MRV0019	Mrvík Ondřej	4
257	MUR0106	Murzinenko Danylo	5
258	MUZ0044	Mužík Jakub	6
259	NAJ0062	Najser Aleš	1
260	NAV0234	Navrátil Jan	2
261	NED0078	Nedojedlý Ondřej	3
262	NEU0127	Neumann Filip	4
263	NGU0370	Nguyenová Hong Hanh	5
264	NIE0094	Niedelský Vojtěch	6
265	NIE0095	Niemczyková Dorota	1
266	NOV0774	Novák Ondřej	2
267	NOV0776	Novotný Jiří	3
268	NOV0786	Novák Lukáš	4
269	OBR0063	Obrusník Václav	5
270	OME0015	Omelka Patrik	6
271	OND0311	Ondrušák Ondřej	1
272	OND0312	Ondo Matěj	2
273	OSM0030	Ošmera František	3
274	OTA0036	Otáhal Vojtěch	4
275	OTR0026	Otruba Michal	5
276	OTT0026	Otta Vojtěch	6
277	OZA0021	Ožana Vojtěch	1

pokračování na další stránce...

	Login	Příjmení jméno	Zadaný projekt
278	PAH0008	Paholík Tomáš	2
279	PAL0310	Palyza Filip	3
280	PAL0331	Pala Adam	4
281	PAP0123	Papež Ondřej	5
282	PAP0124	Papala Matěj	6
283	PAR0182	Parshyn Denys	1
284	PAS0181	Paszanda David	2
285	PAS0217	Pastva Marek	3
286	PAT0116	Pátek Petr	4
287	PAV0519	Pavlica Václav	5
288	PAV0531	Pavlík Petr	6
289	PAV0575	Pavelka Vojtěch	1
290	PAV0622	Pavlasová Adéla	2
291	PAW0042	Pawlik Dominik	3
292	PEG0012	Pěgřimon Adrian	4
293	PEK0083	Pekárek Vojtěch	5
294	PER0215	Perinová Mai Anh	6
295	PET0466	Petřík Tomáš	1
296	PHU0010	Thi Quynh Trang Phung	2
297	PIN0105	Pindur Vojtěch	3
298	PIT0081	Pitek Daniel	4
299	PLA0195	Plaček Roman	5
300	PLA0208	Platoš Jan	6
301	PLE0116	Pletnicki Jan	1
302	POC0035	Pochyla Jan	2
303	POC0036	Pochman Jakub	3
304	POK0132	Pokorný Ondřej	4
305	POL0548	Poloz Maksym	5
306	POL0588	Polášek Petr	6
307	POL0589	Polok Lukáš	1
308	POL0591	Pol Tomáš	2

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
309	POL0592	Polanský Lukáš Vojtěch	3
310	POL0593	Polok Denis	4
311	POL0594	Poledník Lukáš	5
312	PON0078	Poncza Michal	6
313	POP0098	Poppek Lukáš	1
314	POP0100	Popov Oleksandr	2
315	POS0322	Poštulka Jan	3
316	PRA0186	Pravda Daniel	4
317	PRC0027	Prchalová Jana	5
318	PRE0130	Přeček Miroslav	6
319	PRO0414	Proshak Andriy	1
320	PRU0071	Prucek Marek	2
321	PTA0066	Ptaszek Rudolf	3
322	PYS0045	Pyszko Adam	4
323	RAC0074	Rácz Filip	5
324	RAS0158	Raszyk Rene	6
325	REB0021	Rebidáš Šimon	1
326	REP0061	Repka Vladimír	2
327	REZ0144	Řezníček Alex	3
328	ROS0108	Rosa Filip	4
329	ROT0029	Rotari Dmitri	5
330	ROU0075	Roun Jiří	6
331	ROU0077	Roubal Pavel	1
332	RUC0074	Ručková Lucie	2
333	RUC0075	Rucki Alexandr	3
334	RUS0130	Rusnok Michael	4
335	RUS0131	Rusz Vojtěch	5
336	RYS0078	Ryskulova Aidyn	6
337	SAF0114	Šafránek Matěj	1
338	SAF0115	Safarov Aidarbek	2
339	SAI0044	Sairankhan Shynggys	3

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
340	SAL0184	Saloň Gabriel	4
341	SAL0194	Salát Petr	5
342	SAL0195	Salawa Jan	6
343	SAM0135	Samiec Ondřej	1
344	SED0303	Sedláček Petr	2
345	SEN0091	Šenkýř Ondřej	3
346	SER0081	Serikbolov Yeldar	4
347	SES0021	Šestaubr Matěj	5
348	SHA0068	Shalonyi Oleksandr	6
349	SHA0073	Shalda Andrii	1
350	SHA0074	Shakiruly Daniyar	2
351	SCH0469	Scheidel Martin	3
352	SCH0470	Schaffartzik Patrik	4
353	SCH0471	Schuchert Jeremy	5
354	SCH0472	Schwan Lukáš	6
355	SIK0237	Sikora Daniel	1
356	SIK0238	Sika Ondřej	2
357	SIM0441	Šimek Filip	3
358	SIM0442	Šimíček Tomáš	4
359	SIM0445	Šimeček Štěpán	5
360	SIR0081	Šír Adam	6
361	SIX0011	Sixta Pavel	1
362	SKA0226	Skaloš Jakub	2
363	SKI0027	Skilskyy Stanislav	3
364	SKL0075	Sklář Martin	4
365	SKR0165	Škrabal David	5
366	SKU0132	Skupina Michal	6
367	SMA0064	Smagulov Amanbol	1
368	SOB0131	Sobek Matěj	2
369	SOB0139	Soboleva Yuliana	3
370	SOB0140	Sobek Vojtěch	4

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
371	SOK0056	Šokala Vojtěch	5
372	SOL0143	Solichová Nikola	6
373	SOS0047	Sosík Matyáš	1
374	SOT0058	Šotola Jakub	2
375	SPA0201	Špačinský František Jan	3
376	SPA0202	Špalt Vojtěch	4
377	SPI0106	Špicl František	5
378	SPI0110	Špiller Jakub	6
379	SRE0020	Šretrová Miriam	1
380	SRN0030	Srněnská Eliška	2
381	STA0609	Šťastný Radovan	3
382	STA0675	Staňková Tereza	4
383	STA0711	Šťastný Marian	5
384	STE0609	Štefek Lukáš	6
385	STE0610	Štegner Filip	1
386	STE0611	Stehlík Martin	2
387	STE0612	Štefek Jiří	3
388	STE0614	Štecák Darren	4
389	STO0253	Stoklasa Radek	5
390	STO0257	Stonawski Tobijas	6
391	STR0648	Strakoš Pavel	1
392	STU0211	Stupka Ondřej	2
393	SUC0150	Suchý Norbert	3
394	SUL0148	Suleimenov Yesset	4
395	SVA0221	Švanda Jan	5
396	SYK0126	Sýkora Ondřej	6
397	SYM0005	Symerský Pavel	1
398	SZA0044	Szajter Filip	2
399	SZA0045	Szarowski Adam	3
400	TAG0005	Tagan Aktore	4
401	TAN0081	Tancoš Dušan	5

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
402	TAT0025	Tatarchuk Alina	6
403	TEM0029	Temirbek Zamr	1
404	TEO0014	Teofil David	2
405	TER0032	Terekhova Karina	3
406	TES0113	Těšínský Matouš	4
407	THE0030	Theuer Jan	5
408	TKA0086	Tkadlec Vítek	6
409	TOK0031	Toktarov Sanzhar	1
410	TOK0032	Tokmagambet Arlan	2
411	TOM0518	Tomiczek Filip	3
412	TOM0526	Tomasch Sebastián	4
413	TRU0117	Trubák Vít	5
414	TUM0038	Tuma Filip	6
415	TUM0039	Tůma Pavel	1
416	TUR0199	Turek Martin	2
417	TUY0007	Tuyakova Aruana	3
418	UCI0002	Učík Jan	4
419	URB0310	Urban Vitalii	5
420	VAC0300	Václavík Radim	6
421	VAH0022	Vahalík Filip	1
422	VAL0527	Valouch Matěj	2
423	VAL0537	Válek Rostislav	3
424	VAN0363	Vaněk Ondřej	4
425	VAN0376	Vantuch Lucian	5
426	VAS0271	Vašulín Jan	6
427	VAV0279	Vavrys Marek	1
428	VAV0280	Vavřinec Šimon	2
429	VEC0101	Večeřa Jakub	3
430	VEL0126	Velička Jakub	4
431	VER0086	Verner Jakub	5
432	VLC0127	Vlček Radim	6

pokračování na další stránce...

	Login	Příjmení jméno	Zadaný projekt
433	VOI0010	Voityuk Stanislav	1
434	VOJ0129	Vojtěch Jakub	2
435	VOJ0143	Vojtek Marek	3
436	VOJ0161	Vojtek Jakub	4
437	VOJ0163	Vojan Václav	5
438	VOK0033	Voksa František	6
439	VYL0034	Vyleťal Tomáš	1
440	WEB0020	Weber Daniel	2
441	WEL0007	Welszar Dominik	3
442	WIL0042	Wildmann Jáchym	4
443	WIS0027	Wiszczor Adam	5
444	WOJ0082	Wojak Patrik	6
445	WYB0009	Wybitul Adam	1
446	YEP0006	Yepyk Bohdan	2
447	ZAH0111	Zahradníček Robin	3
448	ZAJ0140	Zajíc Filip	4
449	ZAK0115	Žák Matěj	5
450	ZAM0064	Žamboch Vilém	6
451	ZAM0074	Zámostný Jan	1
452	ZAP0127	Zapletal Jan	2
453	ZAR0073	Záruba Václav	3
454	ZAR0074	Žárski Adam	4
455	ZEL0135	Zelman Mykola	5
456	ZHA0067	Zhanalinov Sanzhar	6
457	ZHA0076	Zhakenov Margulan	1
458	ZHU0038	Zhumabayeva Meruyert	2
459	ZID0103	Žídek Adam	3
460	ZIZ0058	Žižková Lenka	4
461	ZMI0010	Zmija Martin	5
462	ZOL0013	Zolfi Masoud	6