Zadání projektu do předmětu **Algoritmy II**
zimní semestr 2025/2026
prezenční a kombinované studium

Historie modifikací

29. října 2025 První verze

Obsah

Deadline

Vypracované řešení projektu je nutno odevzdat do

- prezenční forma studia – **7. prosince 2025 23:59** a
- kombinovaná forma studia – **14. prosince 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 umístěte 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 Konstrukce optimálního vyhledávacího stromu

Problém

V tomto zadání máte dánu posloupnost m slov (řetězců) $W = w_0, w_1, \ldots, w_{m-1}$. Slova v posloupnosti nemusí být nutně navzájem různá. Pro posloupnost W sestavte tři následující vyhledávací stromy:

- *binární vyhledávací strom*, dále jen BST, [1, 2]
- *AVL strom*, dále jen AVL [1, 3, 4] a nakonec
- *optimální binární vyhledávací strom*, dále jen OPT [1, 2].

BST a AVL vytvoříte tak, že do těchto stromů postupně vložíte slova w_0, w_1 až w_{m-1}. Sestavení OPT je trošku náročnější. Nejprve je nutné z posloupnosti W sestavit množinu všech unikátních slov $U = \{u_0, u_1, \ldots, u_{n-1}\}$. Pro každé unikátní slovo u_i je dále nutné určit jeho četnost výskytu f_i a nakonec jeho pravděpodobnost výskytu p_i, kde $p_i = \frac{f_i}{m}$. Na základě pravděpodobností $p_0, p_1, \ldots, p_{n-1}$ pak sestavíte OPT.

Je zřejmé, že každý z výše uvedených tří stromů bude obsahovat přesně n uzlů[3]. Každý uzel tak odpovídá jednomu unikátnímu slovu u_i. Pro každý uzel definujeme c_i jako počet porovnání nutných pro nalezení slova u_i v daném stromu.

Pro všechny tři sestrojené stromy určete následující hodnoty:

- výšku stromu, kde výšku definujeme jako maximální hloubku uzlu ve stromu, a
- průměrný počet porovnání nutných pro nalezení libovolného slova v daném stromu. Průměrný počet porovnání je definován jako

$$C = \sum_{i=0}^{n-1} c_i p_i.$$

OPT se nazývá optimální protože je sestaven tak, aby minimalizoval hodnotu C.

Ukázkový příklad

Mějme posloupnost

$$W = Anton, Caesar, Dora, Caesar, Caesar, Bruno, Bruno, Dora, Caesar, Dora$$

BST a AVL pro tuto posloupnost[4] jsou zobrazeny na obrázku 1. Pro sestavení OPT je nutné získat množinu unikátních slov U a pro každé unikátní slovo $u_i \in U$ musíme vypočítat jeho četnost výskytu f_i a pravděpodobnost výskytu p_i. Výsledky jsou uvedeny v tabulce 1. V tabulce 2 je uveden ukázkový výpočet průměrného počtu porovnání pro jednotlivé stromy. Výsledky srovnání[5] jednotlivých binárních stromů pro ukázkovou posloupnost slov W jsou uvedeny v tabulce 3.

Na obrázku 2 můžeme vidět, že pokud pořadí slov v posloupnosti změníme dostaneme jiný BST a AVL, než jaké jsme dostali pro původní posloupnost. A dokonce tyto stromy mohou být shodné s OPT. Optimální strom je ale vždy stejný, jeho podoba závisí na rozložení pravděpodobností výskytů jednotlivých slov. Výsledky pro tyto tři stromy jsou uvedeny v tabulce 4.

A nakonec výsledky pro delší posloupnost jsou uvedeny v tabulce 5 a sestrojené stromy na obrázcích 3, 4 a 5.

[3] Struktura BST, AVL a OPT se však může podstatným způsobem lišit.

[4] Kvůli úspoře místa jsou v uzlech stromů uvedeny jen počáteční písmena slov z ukázkové posloupnosti.

[5] Celkový počet slov a počet unikátních slov jsou v tabulce uvedeny spíše jen pro kontrolu. Stejně tak můžete postupovat i u vašeho řešení. Budete tak vědět, kolik slov bylo do stromu vloženo…

i	u_i	f_i	p_i
0	Anton	1	0,1
1	Bruno	2	0,2
2	Caesar	4	0,4
3	Dora	3	0,3
	Součet	10	1,0

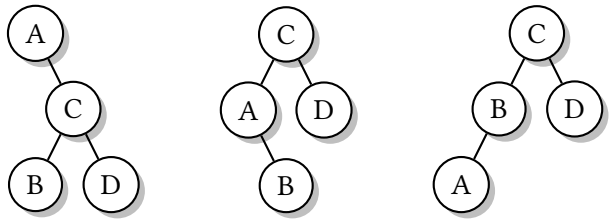
Tabulka 1: Ukázková množina unikátních slov U

			BST		AVL		OPT	
i	u_i	p_i	c_i	$c_i p_i$	c_i	$c_i p_i$	c_i	$c_i p_i$
0	Anton	0,1	1	0,1	2	0,2	3	0,3
1	Bruno	0,2	3	0,6	3	0,6	2	0,4
2	Caesar	0,4	2	0,8	1	0,4	1	0,4
3	Dora	0,3	3	0,9	2	0,6	2	0,6
		C		2,4		1,8		1,7

Tabulka 2: Výpočet průměrného počtu porovnání C

	BST	AVL	OPT
Celkový počet slov	10	10	10
Počet unikátních slov	4	4	4
Výška stromu	3	3	3
Průměrný počet porovnání C	2,4	1,8	1,7

Tabulka 3: Srovnání parametrů pro soubor `Test1.txt`

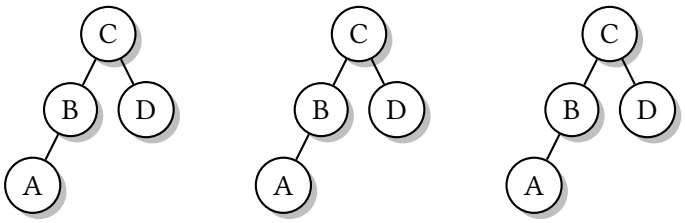


(a) BST (b) AVL (c) OPT

Obrázek 1: Binární stromy vytvořené ze souboru `Test1.txt`

	BST	AVL	OPT
Celkový počet slov	10	10	10
Počet unikátních slov	4	4	4
Výška stromu	3	3	3
Průměrný počet porovnání	1,7	1,7	1,7

Tabulka 4: Srovnání parametrů pro soubor `Test1b.txt`

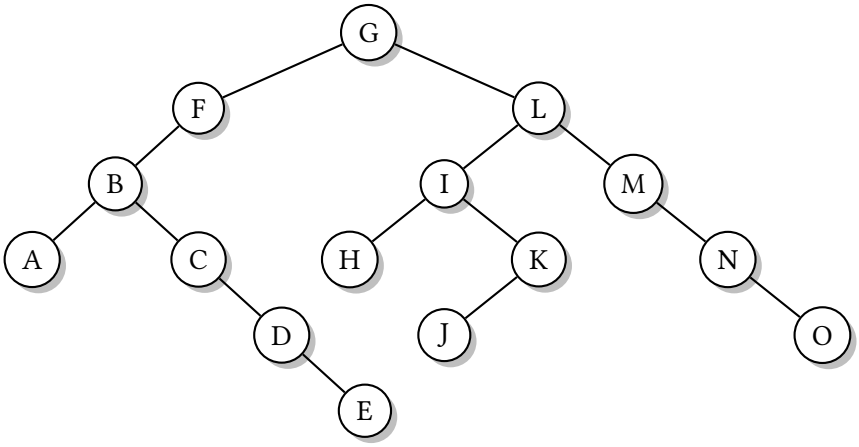


(a) BST (b) AVL (c) OPT

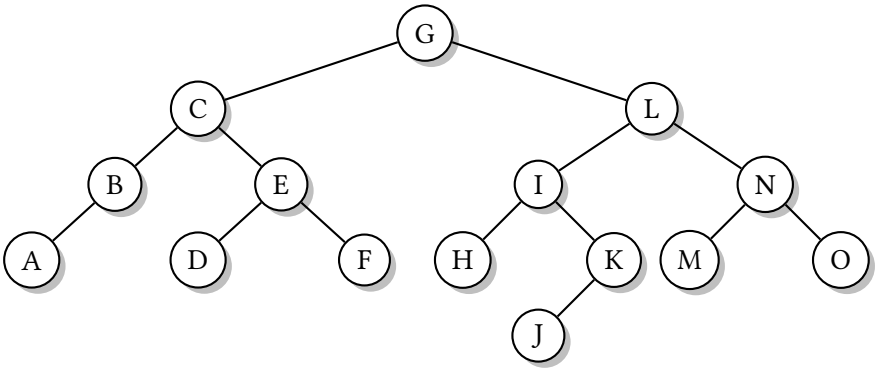
Obrázek 2: Binární stromy vytvořené ze souboru `Test1b.txt`

	BST	AVL	OPT
Celkový počet slov	100	100	100
Počet unikátních slov	15	15	15
Výška stromu	6	5	5
Průměrný počet porovnání	3,05	2,76	2,56

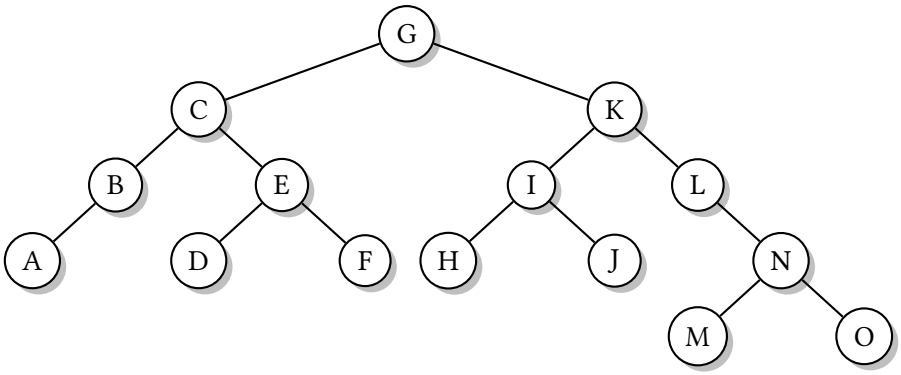
Tabulka 5: Srovnání parametrů pro soubor `Test2.txt`



Obrázek 3: BST vytvořený ze souboru `Test2.txt`



Obrázek 4: AVL vytvořený ze souboru `Test2.txt`



Obrázek 5: OPT vytvořený ze souboru `Test2.txt`

Poznámky

- Vstupem do vašeho programu bude textový soubor obsahující seznam slov. Každé slovo je uloženo na samostatném řádku. Stromy na obrázku 1 vznikly zpracováním textového souboru `Test1.txt`, který má tento obsah:

  ```
  Anton
  Caesar
  Dora
  Caesar
  Caesar
  Bruno
  Bruno
  Dora
  Caesar
  Dora
  ```

  Obsah tohoto souboru odpovídá posloupnosti slov z ukázkového řešení.

- Požadované parametry BST, AVL a OPT vypište na standardní výstup.
- Vaše řešení otestujte na všech testovacích souborech uvedených v zadání projektu.

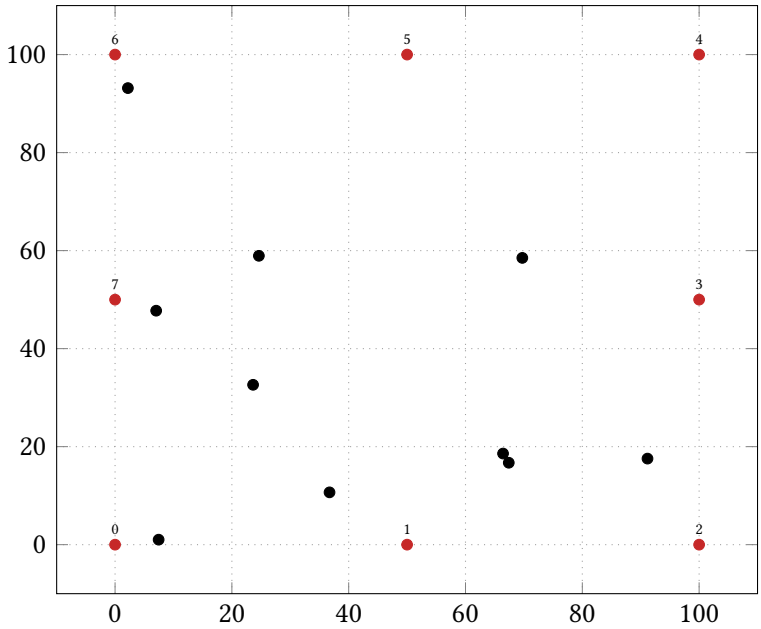
2 Ropné plošiny

Problém

Vládce pouštního státu Wadiya hodlá rozšířit těžbu ropy i do pevninského šelfu. V šelfovém moři již stojí několik ropných plošin, které je však nutné propojit potrubím mezi sebou a následně připojit k přečerpávací stanici, odkud se bude ropa dále přečerpávat do tankerů. Vaším úkolem je navrhnout soustavu potrubí pro dopravu ropy od ropných plošin k přečerpávací stanici, při dodržení následujících podmínek.

- Potrubí je vedeno vždy přímočaře, délka potrubí odpovídá euklidovské vzdálenosti mezi dvěma body[6].
- V systému potrubí nesmí vzniknout cyklus, aby ropa v systému nekroužila dokola. Ropa musí téct od ropné plošiny směrem k přečerpávací stanici. Je ale možné propojovat ropné plošiny potrubím mezi sebou. Není nutné vždy propojovat ropnou plošinu přímo s přečerpávací stanicí.

[6]Velikost samotné ropné plošiny nebo přečerpávací stanice zanedbáváme.



Obrázek 6: Ropné plošiny a možné přečerpávací stanice (červeně)

Přečerpávací stanice	Celková délka potrubí
0	244,797
1	247,696
2	256,928
3	262,278
4	288,623
5	283,205
6	244,449
7	244,660

Tabulka 6: Délka potrubní soustavy pro možné polohy přečerpávací stanice

- O kapacitu potrubí se nestarejte[7].
- Součástí návrhu potrubního systému je i výběr lokality pro výstavbu přečerpávací stanice. Přečerpávací stanice má být v systému jen jedna. Stanici je možné vybudovat v jedné z osmi možných lokalit. Tyto lokality jsou součástí zadání. Vaším úkolem je vybrat lokalitu přečerpávací stanice tak, aby délka celého potrubního systému byla co nejkratší.

Ukázkový příklad

V ukázkovém příkladu předpokládejme, že máme deset ropných plošin a osm možných přečerpávacích stanic rozmístěných tak jak je vidět na obrázku 6. Délky nejkratších potrubních systémů vedoucích k osmi možným polohám přečerpávací stanice jsou uvedeny v tabulce 6. Z ní je patrné, že nejkratší potrubní systém vede ke stanici číslo 6 a systém je dlouhý 244,449 délkových jednotek.

[7] Obecně můžeme říct, že kratší potrubí libovolných parametrů bude levnější, než delší potrubí stejných parametrů.

Poznámky

- Souřadnice ropných plošin jsou uloženy v textém souboru, souřadnice každé plošiny na samostatném řádku. Souřadnice jsou uloženy jako desetinná čísla, nejprve souřadnice x, potom souřadnice y. Souřadnice jsou odděleny bílým znakem. Ropné plošiny z ukázkového příkladu, viz obrázek 6 jsou uloženy v textovém souboru takto:

```
69.728 58.516
66.430 18.591
7.038 47.741
91.161 17.560
36.719 10.675
24.623 58.941
2.196 93.164
7.459 1.018
67.409 16.716
23.630 32.620
```

- Souřadnice přečerpávacích stanic jsou uloženy stejně jako souřadnice ropných plošin. Přečerpávací stanice z ukázkového příkladu, viz obrázek 6 jsou uloženy v textovém souboru takto:

```
0.000 0.000
50.000 0.000
100.000 0.000
100.000 50.000
100.000 100.000
50.000 100.000
0.000 100.000
0.000 50.000
```

- Nalezené řešení vypište ve vhodném tvaru na standardní výstup.

3 Největší souvislá oblast

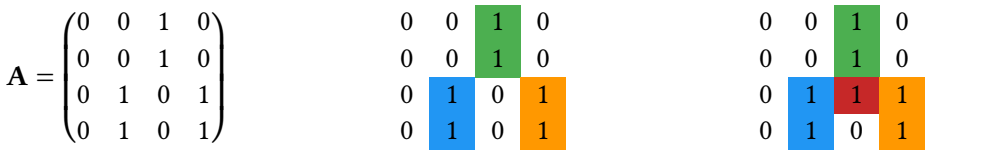
V tomto zadání máte danou čtvercovou matici řádu n. Matice je binární, tj. obsahuje pouze hodnoty 0 a 1. V této matici nás zajímají souvislé oblasti prvků s hodnotou 1. Souvislá oblast je složena z navzájem sousedních prvků. Dva prvky se považují za sousední, pokud spolu sousedí vodorovně nebo svisle (nahoře, dole, vlevo, vpravo)[8].

Vaším úkolem je implementovat algoritmus, který v této matici změní nejvýše jeden její prvek z 0 na 1 tak, aby došlo k propojení nejméně dvou souvislých oblastí jedniček a nově vznikla souvislá oblast s co největším počtem jedniček. Pokud takový prvek neexistuje, algoritmus vypíše, že řešení neexistuje.

Příklad

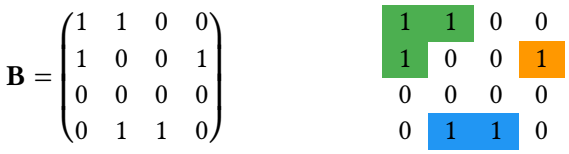
Na obrázku 7 můžeme vidět čtvercovou matici **A** řádu 4, která obsahuje celkem tři souvislé oblasti jedniček. Změnou prvku $a_{3,3}$ z 0 na 1, na obrázku je označen červeně, tyto oblasti spojíme a získáme jednu souvislou oblast se sedmi jedničkami. Pokud bychom z 0 na 1 změnili prvek $a_{2,2}$, resp. $a_{2,4}$, spojili bychom dvě oblasti, ale výsledná oblast by měla jen pět jedniček a tudíž by byla menší, než oblast získaná změnou prvku $a_{3,3}$.

[8] Každý prvek v matici má tak maximálně čtyři sousedy, prvky sousedící diagonálně neuvažujeme.



(a) vstupní matice (b) souvislé oblasti jedniček (c) největší propojená oblast

Obrázek 7: Největší souvislá oblast, řešitelné zadání



(a) vstupní matice (b) souvislé oblasti jedniček

Obrázek 8: Největší souvislá oblast, neřešitelné zadání

Na dalším obrázku 8 můžeme vidět čtvercovou matici **B** řádu 4, která také obsahuje tři souvislé oblasti jedniček. Nicméně změnou žádného prvku matice **B** z 0 na 1 nespojíme aspoň dvě oblasti dohromady a tudíž úloha nemá řešení.

Poznámky

- Každé zadání, matice, je uložena v samostatné textovém souboru. V textovém souboru je celkem $n + 1$ řádků. První řádek obsahuje pouze číslo n, to jest řád matice. Dále v souboru následuje n řádků, kde každý řádek obsahuje n nul a jedniček oddělených bílým znakem. Matice **A** ze zadání na obrázku 7 bude v textovém souboru uložena takto:

```
4
0 0 1 0
0 0 1 0
0 1 0 1
0 1 0 1
```

- Jako řešení vypište na standardní výstup řádek a sloupec prvku vstupní matice, který se změnil z 0 na 1 a dále počet jedniček v takto vytvořené souvislé oblasti. Pokud řešení neexistuje, vypište o tom na standardní výstup zprávu.

4 Závislosti zdrojových kódů

Problém

V tomto zadání máte za úkol řešit kompilaci zdrojových kódů rozsáhlého projektu. Takový projekt se obvykle skládá z mnoha a mnoha souborů se zdrojovým kódem. U projektů implementovaných v jazyce C++ jde typicky o `cpp` soubory a do nich vkládané hlavičkové soubory s příponou `h`. Představme si nyní, že jsme v roli vývojáře pracujícího na projektu, provedli změnu v jednom ze souborů projektu. Pokud chceme projekt po této změně zkompilovat, abychom získali aktuální verzi, máme dvě možnosti:

1. zkompilovat úplně všechny zdrojové kódy, což ale může být docela zdlouhavý proces, nebo
2. zkompilovat jen změněné části projektu.

a.cpp

```
#include "a.h"
```

b.cpp

```
#include "b.h"
```

c.cpp

```
#include "c.h"
```

a.h

```
#include <iostream>
#include <vector>
#include "b.h"
```

b.h

```
#include <fstream>
#include <vector>
```

c.h

```
#include <cstdlib>
#include <iostream>
#include "a.h"
```

iostream

```
#include <ios>
```

fstream

```
#include <ios>
```

Obrázek 9: Zdrojové a hlavičkové soubory, jednoduchý příklad. Hlavičkový soubor `ios`, zde není uvedený, protože se už neodkazuje na další soubory.

Pokud chceme kompilovat jen nezbytně nutné části projektu, čili jen některé zdrojové kódy, musíme znát závislosti mezi jednotlivými soubory se zdrojovým kódem. Závislost vznikne například pomocí direktivy `#include`. Tyto závislosti můžeme reprezentovat orientovaným grafem G, kde vrcholy grafu reprezentují jednotlivé soubory a hrany reprezentují závislosti souborů. Vede-li například orientovaná hrana z vrcholu u do vrcholu v, znamená to, že soubor u závisí na souboru v. Z tohoto grafu můžeme ale také určit, které soubory musíme znovu zkompilovat, když provedeme změnu v souboru reprezentovaném nějakým vrcholem w.

Vaším úkolem je implementovat algoritmus, který pro zadaný graf závislostí mezi soubory G a daný soubor (vrchol grafu) v určí, které soubory jsou změnou souboru v ovlivněny a musí se znovu zkompilovat. Následně tento algoritmus spusťte pro všechny vrcholy v grafu G.

Ukázkový příklad

Pro ukázku předpokládejme, že máme tři soubory se zdrojovým kódem – `a.cpp`, `b.cpp` a `a.cpp`. Každý z těchto zdrojových kódů má svůj vlastní hlavičkový soubor – `a.h`, `b.h` a `a.h`. Hlavičkové soubory jsou navzájem dále provázány pomocí direktivy `#include` a dále se tyto hlavičkové soubory odkazují na hlavičkové soubory ze standardní knihovny, například `fstream`.

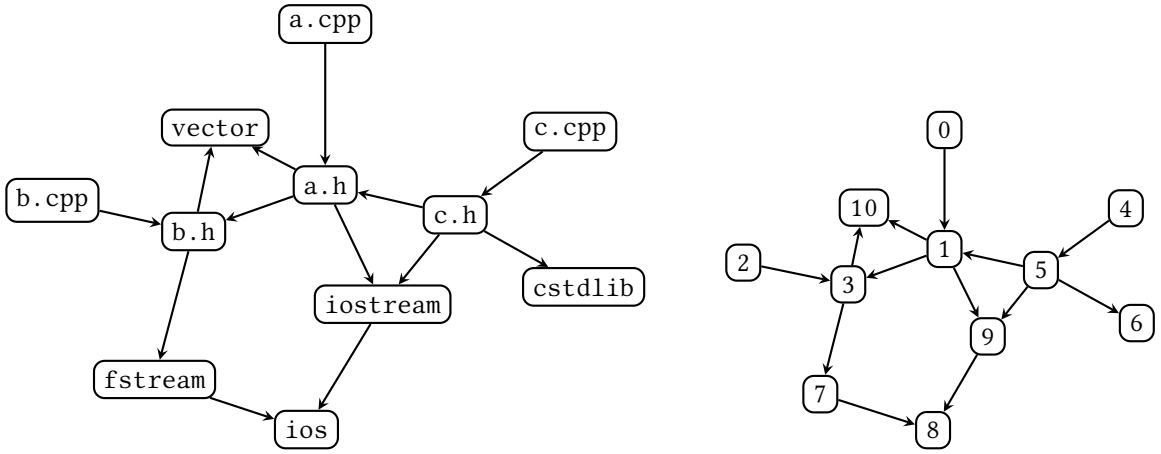
Obsah ukázkových zdrojových a hlavičkových souborů můžeme vidět na obrázku 9. Veškerý obsah jak zdrojových, tak hlavičkových souborů je nezajímavý a je pro přehlednost zcela vynechán. Co je ale na ukázkových souborech důležité je vzájemná provázanost mezi soubory daná direktivami `#include`. Z obrázku tak plyne, že soubor `a.cpp` se odkazuje na soubor `a.h` a ten se odkazuje na soubory `iostream`, `vector` a `b.h`. A tyto soubory se odkazují na další soubory a další soubory. Výsledkem je tak graf závislostí mezi soubory G zobrazený na obrázku 10a. Pro zjednodušení zpracování v počítači můžeme jména souborů nahradit celočíselnými identifikátory pomocí tabulky 7. Použijeme-li tyto identifikátory místo jmen souborů dostáváme graf na obrázku 10b. Výsledky, tj. které soubory musíme znovu zkompilovat při změně daného souboru, jsou uvedeny v tabulce 8.

Graf závislostí pro větší skupinu souborů je uveden na obrázku 11.

Poznámky

- Vstupem do vašeho programu bude textový soubor obsahující specifikaci grafu závislostí souborů v pomyslném projektu. Vrcholy grafu jsou označeny nezápornými celými čísly. Graf je uložen ve formě seznamu hran. Na každém řádku vstupního textového souboru je uložena jedna hrana grafu ve tvaru `u -> v`, kde `u`, `v` jsou vrcholy grafu a orientovaná hrana vede z vrcholu `u` do vrcholu `v`. Graf z obrázku 10b tak bude uložen ve tvaru:

  ```
  0 -> 1
  ```



(a) jména souborů (b) číselné identifikátory

Obrázek 10: Jednoduchá ukázka závislosti zdrojových kódů

Soubor	Id
a.cpp	0
a.h	1
b.cpp	2
b.h	3
c.cpp	4
c.h	5

Soubor	Id
cstdlib	6
fstream	7
ios	8
iostream	9
vector	10

Tabulka 7: Soubory a jejich číselné identifikátory

Modifikovaný soubor	Nutná kompilace souborů
0	0
1	0, 1, 4, 5
2	2
3	0, 1, 2, 3, 4, 5
4	4
5	4, 5
6	4, 5, 6
7	0, 1, 2, 3, 4, 5, 7
8	0, 1, 2, 3, 4, 5, 7, 8, 9
9	0, 1, 4, 5, 9
10	0, 1, 2, 3, 4, 5, 10

Tabulka 8: Výsledky pro soubory a jejich závislosti zobrazené na obrázku 10

```
2 -> 3
4 -> 5
1 -> 3
1 -> 10
1 -> 9
3 -> 10
3 -> 7
5 -> 9
5 -> 6
5 -> 1
9 -> 8
7 -> 8
```

- Výsledek vypište ve vhodné formě na standardní výstup.

5 Centrum grafu

Problém

V tomto zadání budeme pracovat s neorientovanými grafy. Vzdálenost mezi dvěma vrcholy v grafu měříme jako délku nejkratší cesty mezi nimi. Při řešení praktických problémů má smysl se podívat na extrémní vzdálenosti v grafu. Předpokládejme, že máme dán graf G a vrchol $v \in V(G)$. Excentricita vrcholu v je největší ze všech vzdáleností vrcholu v od ostatních vrcholů v grafu G. Excentricitu značíme $ecc(v)$. Excentricitě vrcholu se v literatuře říká také výstřednost vrcholu. V souvislých grafech je excentricita vrcholu v měřítkem, jak „daleko“ je k ostatním vrcholům v grafu, viz obrázek 12. Vrcholy s velkou excentricitou leží „na kraji“ grafu a vrcholy s malou excentricitou leží „uvnitř grafu“. Množina vrcholů, ze kterých je „blízko“ do všech ostatních vrcholů, tvoří centrum grafu. Centrum grafu je tedy podgraf daného grafu a tvoří jej vrchol, či vrcholy, s nejmenší excentricitou. Vaším úkolem je pro daný graf vypsat vrcholy, které tvoří jeho centrum.

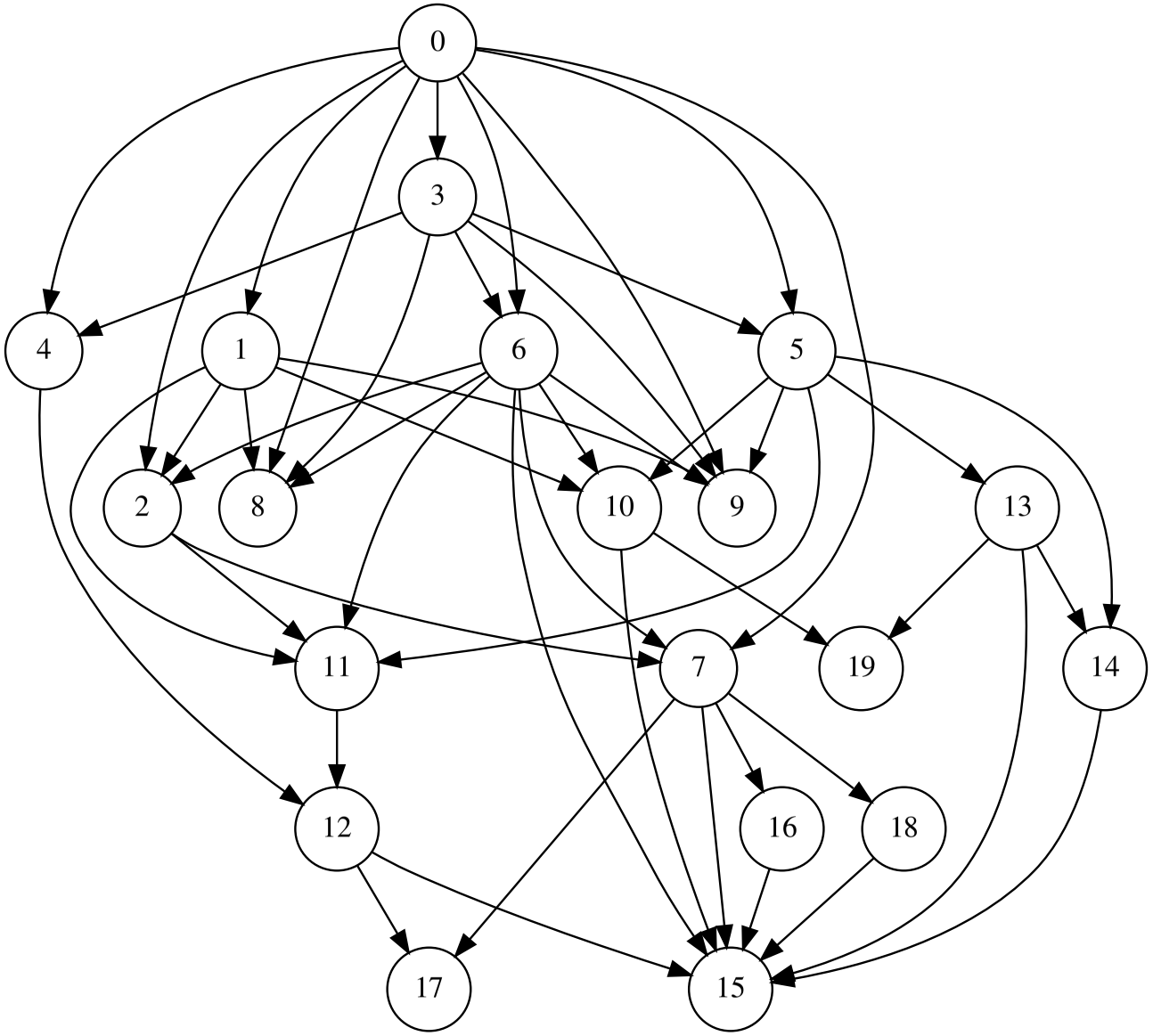
Ukázkový příklad

V ukázkovém grafu na obrázku 12 jsou pro každý vrchol červeně vypsána jeho excentricita. Z obrázku je patrné, že centrum grafu tvoří vrchol 5 s excentricitou 2.

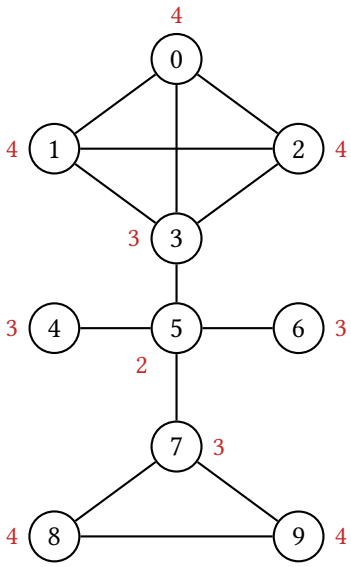
Poznámky

- Každý vrchol grafu je označen, celým číslem i. Graf je uložen v textovém souboru v následujícím formátu. Na každém řádku je uložena dvojice čísel celých čísel i a j oddělených tabulátorem. Čísla i a j představují vrcholy grafu mezi kterými existuje hrana. Ukázkový graf z obrázku 12 bude uložen takto:

```
0	1
0	2
0	3
1	2
1	3
2	3
3	5
4	5
5	6
5	7
```



Obrázek 11: Komplexnější ukázka závislostí zdrojových kódů



Obrázek 12: Excentricity (zobrazeny červeně) vrcholů grafu

7	8
7	9
8	9

6 Indexování textu

Problém

V tomto zadání máte za úkol sestavit index pro vyhledávání v množině textových dokumentů. Indexování textových dokumentů se provádí ze stejného důvodu jako indexování záznamů v databázi – urychlení vyhledávání. V dnešní době se technologie indexování textových dokumentů nejčastěji pro vyhledávání na webu.

Předpokládejme, že máme množinu n dokumentů $\{D_0, D_1, \ldots, D_{n-1}\}$. Proces indexování se skládá z několika kroků, které je nutné provést pro každý dokument D_i:

1. Po načtení rozebereme dokument na jednotlivá slova. Tomuto procesu se říká lexikální analýza. Slovo definujeme jako konečnou neprázdnou posloupnost malých a velkých písmen anglické abecedy a znaku apostrof, která začíná malým nebo velkým písmenem. Tím dostaneme seznam slov L_i. V tomto seznamu jsou obsažena všechna slova z daného dokumentu v pořadí, ve kterém se v dokumentu nachází a to včetně duplicit.

2. Všechna slova v seznamu L_i převedeme na malá písmena.

3. Ze seznamu L_i vymažeme všechna stop slova (angl. stopwords). Stop slova jsou typicky slova, která nenesou žádný význam a jsou nutná jen z důvodů gramatických, například spojky, předložky, pomocná slovesa a tak dále.

4. Všechna slova v L_i převedeme na základní tvar, tzv. *stem*[9]. Tomuto procesu se říká stemming a provádí jej algoritmus zvaný stemmer. Implementaci Porterova stemmeru najdete například zde https://tartarus.org/martin/PorterStemmer/.

[9] Což zhruba odpovídá tomu, čemu v češtině říkáme „kořen slova“.

5. Do indexu pro každý stem v seznamu L_i zařadíme trojici $(i,$ `stem, pořadí slova v` $L_i)$, kde i je identifikátor dokumentu.

Důležitou součástí zadání je návrh vhodné datové struktury pro reprezentaci indexu. Tato datová struktura musí umožňovat efektivně pro daný stem vyhledat seznam identifikátorů dokumentů, kde se hledaný stem vyskytuje, a současně pro každý takový dokument seznam pozic, na kterých se hledaný stem v dokumentu vyskytl.

Ukázkový příklad

Předpokládejme, že máme vytvořit index pro tyto tři dokumenty:

Dokument	Text dokumentu
D_0	The computer a monitor.
D_1	Computer or mouse mouse.
D_2	Mouse not monitor the computer.

Lexikální analýzou těchto dokumentů, a převodem na malá písmena, dostáváme seznamy slov

L_0 = (the,computer,a,monitor)
L_1 = (computer,or,mouse,mouse)
L_2 = (mouse,not,monitor,the,computer)

Po odstranění stop slov dostáváme

L_0 = (computer,monitor)
L_1 = (computer,mouse,mouse)
L_2 = (mouse,monitor,computer)

A konečně po stemingu dostáváme

L_0 = (comput,monitor)
L_1 = (comput,mous,mous)
L_2 = (mous,monitor,comput)

Do indexu tak zařadíme tyto trojice

```
(0, comput, 0)
(0, monitor, 1)
(1, comput, 0)
(1, mous, 1)
(1, mous, 2)
(2, mous, 0)
(2, monitor, 1)
(2, comput, 2)
```

Pokud budeme pak v indexu chtít vyhledat například slovo „mouse“, převedeme jej na stem „mous“ a v indexu nalezneme, že se tento stem vyskytuje v dokumentu 1 na pozici 1 a 2, a dále v dokumentu 2 na pozici 0.

Poznámky

- V našem zadání pro jednoduchost předpokládejme, že množina dokumentů určených k indexaci je uložena v jednom textovém souboru. Každý řádek v tomto textovém souboru představuje jeden dokument. Číslo řádku v textovém souboru, počínaje 0, budeme považovat za identifikátor dokumentu.
- Opět pro jednoduchost předpokládejme, že text je v angličtině a neobsahuje tedy text s diakritikou.
- Seznam stop slov je součástí zadání.
- Implementaci Porterova stemmeru můžete pochopitelně převzít, není nutné jej implementovat znovu.
- Pro reprezentaci indexu navrhněte vhodnou datovou strukturu. Stejně tak navrhněte vhodnou datovou strukturu ve které se bude vracet výsledek hledání stemu v indexu.
- Výsledek hledání stemu vypište ve vhodné formě na standardní výstup.
- V implementaci není nezbytně nutné budovat seznam slov v jednotlivých dokumentech. Ty pak převádět na malá písmena, čistit od stop slov a převádět na stemy. Při použití vhodné implementace lze budovat rovnou index.

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. MAREŠ, Martin; VALLA, Tomáš. *Průvodce labyrintem algoritmů* [online]. Praha: CZ.NIC, z.s.p.o., 2017 [cit. 2021-04-19]. ISBN 978-80-88168-19-5.
4. WIRTH, Niklaus. *Algoritmy a štruktúry údajov*. 1. vyd. Bratislava: Alfa, 1988. ISBN 063-030-87.

Rozdělení zadání mezi studenty

Prezenční forma studia

	Login	Příjmení jméno	Zadaný projekt
1	ADA0305	Adámek Daniel	1
2	ANL0009	Anlauf Dominik	2
3	ANT0100	Anton Štěpán	3
4	ASC0002	Ascher Adam	4
5	BAL0330	Balus David	5
6	BAR0866	Bartoš Ámos	6
7	BEC0075	Bečka Daniel	1
8	BEN0339	Beneš Zdeněk	2
9	BER0307	Berezovský Peter	3
10	BER0328	Bernaťák Václav	4
11	BER0330	Bernatík Richard	5
12	BIE0069	Bierhanzlová Lucie	6
13	BIL0201	Bílý Martin	1
14	BLA0385	Blahuta Vojtěch	2
15	BRA0241	Brázda Pavel	3
16	BRU0098	Brudovský Andreas	4
17	BUS0054	Buš Jan	5
18	CAN0054	Čanecký Lukáš	6
19	CVI0013	Cvik Lukáš	1
20	CVI0014	Cvikl Adam	2
21	DAN0208	Danihel Lukáš	3
22	DAN0209	Daník Šimon	4
23	DOB0171	Dobiášovský Michal	5
24	DOD0012	Do Duc Trung	6
25	DRO0115	Drössler Lukáš	1
26	DZI0038	Dzifčák Miroslav	2
27	FAB0083	Fabisz Samuel	3
28	FEI0025	Feichtinger Jakub	4
29	FIT0007	Fitřík Jan	5

pokračování na další stránce...

	Login	Příjmení jméno	Zadaný projekt
30	FOJ0149	Fojtík Jakub	6
31	FOR0088	Formánek Filip	1
32	GAL0207	Galica Jakub	2
33	GAN0051	Gánovský Peter	3
34	GAR0165	Gardoš Filip	4
35	GEI0005	Geiler Ilja	5
36	GRA0126	Graf Erik	6
37	GRI0073	Grichová Adéla	1
38	GRY0125	Grygarčík Ondřej	2
39	GUZ0034	Guziur Ondřej	3
40	HAD0067	Haderka Martin	4
41	HAL0269	Halfar Vojtěch	5
42	HAN0412	Hanusek Adam	6
43	HAN0413	Hanskyi Nikita	1
44	HAR0199	Harok Antonín	2
45	HAR0205	Haraimovych Yaroslav	3
46	HAV0306	Havelka Václav	4
47	HEC0083	Heczko Vojtěch	5
48	HEI0081	Heider Filip	6
49	HEJ0094	Hejtman Matěj	1
50	HLU0053	Hlubina Martin	2
51	HOL0601	Holub Martin	3
52	HOM0085	Homolka Vít	4
53	HOS0126	Hosnédl Karel	5
54	HRC0018	Hrček Adam	6
55	HRN0040	Hrňa Lukáš	1
56	HRO0111	Hrozová Julie	2
57	HRO0117	Hroch Petr	3
58	HYN0045	Hýnar Martin	4
59	CHA0265	Chabroň Denis Kristián	5
60	CHE0084	Chehil Yurii	6

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
61	CHE0086	Chervinskyi Nikita	1
62	CHM0086	Chmelík Mojmír	2
63	CHO0281	Cholak Maksym	3
64	CHO0288	Chobot Marek	4
65	CHO0289	Chovanec Richard	5
66	CHR0202	Chrascina Dominik	6
67	CHV0053	Chvostková Veronika	1
68	CHY0105	Chytil Jakub	2
69	IND0045	Indruch Jakub	3
70	JAN0992	Janíček Adam	4
71	JAN0993	Jančík Vojtěch	5
72	JAR0183	Jaroš František	6
73	JAR0193	Jarabica Martin	1
74	JEZ0107	Ježik Dávid	2
75	JUR0545	Jurky Michal	3
76	JUR0546	Jurčaga Tomáš	4
77	KAL0324	Kaldybayeva Assel	5
78	KAL0339	Kalabiha Myroslav	6
79	KAL0362	Kaleta Jakub	1
80	KAN0268	Kaňa Jakub	2
81	KAN0310	Kantor Robin	3
82	KAN0313	Kaňák Ondřej	4
83	KAN0314	Kaňok Daniel	5
84	KAN0316	Kanovský David	6
85	KAN0321	Kandiushyn Dmytro	1
86	KAR0321	Karpjuk René	2
87	KHA0047	Khamzina Shakira	3
88	KHO0028	Khomulenko Ilya	4
89	KIJ0017	Kijonka Adam	5
90	KLI0302	Klimíček Ondřej	6
91	KNA0060	Knapek Jakub	1

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
92	KOK0068	Kokeš Radim	2
93	KOL0602	Koliba Marek	3
94	KOL0607	Kolmačka Ondřej	4
95	KOL0620	Kollár David	5
96	KOL0629	Kolomazník Jiří	6
97	KOS0431	Košař Adam	1
98	KOT0325	Kotyra Adam	2
99	KOT0326	Kotrba Matěj	3
100	KOV0426	Koval Jakub	4
101	KOZ0381	Kožák Pavol	5
102	KOZ0382	Kozel Marek	6
103	KRA0677	Kraváriková Glória	1
104	KRA0745	Kratoš Samuel	2
105	KRE0463	Kreutz Adam	3
106	KUB0809	Kubesa Jakub	4
107	KUB0812	Kubica Adam	5
108	KUC0436	Kučera Ondřej	6
109	KUC0437	Kučera David	1
110	KUC0438	Kučo Samuel	2
111	KUD0148	Kudělka Vojtěch	3
112	LIC0113	Lichý Petr	4
113	LIT0041	Litvan Jan	5
114	LYS0058	Lysenko Pavlo	6
115	MAC0689	Maceček Patrik	1
116	MAJ0187	Majvelder Filip	2
117	MAR1025	Mařík Michal	3
118	MAT0495	Matula Miroslav	4
119	MAT0514	Matsyfuk Artem	5
120	MAT0586	Matyskiewicz Marek	6
121	MAX0019	Maximova Veronika	1
122	MAZ0126	Mazúr Kristián	2

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
123	MIC0548	Michna Matúš	3
124	MIS0104	Misař Viliam	4
125	MIT0114	Mitura Karel	5
126	MLC0053	Mlčák Jan	6
127	MUZ0044	Mužík Jakub	1
128	NAJ0062	Najser Aleš	2
129	NEU0127	Neumann Filip	3
130	NGU0370	Nguyenová Hong Hanh	4
131	NIE0087	Niemczyková Eliška	5
132	NIE0094	Niedelský Vojtěch	6
133	NIE0095	Niemczyková Dorota	1
134	NOV0786	Novák Lukáš	2
135	OME0015	Omelka Patrik	3
136	OND0311	Ondrušák Ondřej	4
137	OND0312	Ondo Matěj	5
138	OTR0026	Otruba Michal	6
139	OZA0021	Ožana Vojtěch	1
140	PAH0008	Paholík Tomáš	2
141	PAL0331	Pala Adam	3
142	PAP0123	Papež Ondřej	4
143	PAP0124	Papala Matěj	5
144	PAS0217	Pastva Marek	6
145	PAT0116	Pátek Petr	1
146	PAV0519	Pavlica Václav	2
147	PAV0531	Pavlík Petr	3
148	PAV0575	Pavelka Vojtěch	4
149	PAZ0043	Pazderskyi Maksym	5
150	PEG0012	Pěgřimon Adrian	6
151	PEK0083	Pekárek Vojtěch	1
152	PER0215	Perinová Mai Anh	2
153	PET0466	Petřík Tomáš	3

pokračování na další stránce...

	Login	Příjmení jméno	Zadaný projekt
154	PHU0010	Thi Quynh Trang Phung	4
155	PIT0081	Pitek Daniel	5
156	PLA0208	Platoš Jan	6
157	POC0035	Pochyla Jan	1
158	POL0588	Polášek Petr	2
159	POL0591	Pol Tomáš	3
160	POL0592	Polanský Lukáš Vojtěch	4
161	PON0078	Poncza Michal	5
162	POP0100	Popov Oleksandr	6
163	PRA0186	Pravda Daniel	1
164	PRC0027	Prchalová Jana	2
165	PRE0130	Přeček Miroslav	3
166	PRO0414	Proshak Andriy	4
167	PYS0045	Pyszko Adam	5
168	REB0021	Rebidáš Šimon	6
169	ROS0108	Rosa Filip	1
170	ROT0029	Rotari Dmitri	2
171	ROU0075	Roun Jiří	3
172	RUS0130	Rusnok Michael	4
173	RYC0095	Rychlý Štěpán	5
174	SAF0114	Šafránek Matěj	6
175	SAI0044	Sairankhan Shynggys	1
176	SEN0091	Šenkýř Ondřej	2
177	SER0081	Serikbolov Yeldar	3
178	SES0021	Šestaubr Matěj	4
179	SHA0074	Shakiruly Daniyar	5
180	SCH0470	Schaffartzik Patrik	6
181	SIM0442	Šimíček Tomáš	1
182	SKA0226	Skaloš Jakub	2
183	SKI0027	Skilskyy Stanislav	3
184	SKL0075	Sklář Martin	4

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
185	SKU0132	Skupina Michal	5
186	SMA0064	Smagulov Amanbol	6
187	SOB0131	Sobek Matěj	1
188	SOB0139	Soboleva Yuliana	2
189	SOK0056	Šokala Vojtěch	3
190	SOT0058	Šotola Jakub	4
191	SPA0202	Špalt Vojtěch	5
192	SPI0110	Špiller Jakub	6
193	SRE0020	Šretrová Miriam	1
194	STA0609	Šťastný Radovan	2
195	STA0711	Šťastný Marian	3
196	STE0610	Štegner Filip	4
197	STE0611	Stehlík Martin	5
198	STU0211	Stupka Ondřej	6
199	SUL0148	Suleimenov Yesset	1
200	SZA0044	Szajter Filip	2
201	TAN0081	Tancoš Dušan	3
202	TAT0025	Tatarchuk Alina	4
203	TEO0014	Teofil David	5
204	TOK0030	Tokarský Robin	6
205	TOK0031	Toktarov Sanzhar	1
206	TOM0518	Tomiczek Filip	2
207	TOM0526	Tomasch Sebastián	3
208	TRA0162	Trabalík Matej	4
209	TRU0117	Trubák Vít	5
210	TUM0038	Tuma Filip	6
211	TUR0199	Turek Martin	1
212	TYM0011	Tymoshenko Bohdan	2
213	URB0310	Urban Vitalii	3
214	VAB0011	Vabroušek Petr	4
215	VAL0527	Valouch Matěj	5

pokračování na další stránce...

	Login	Příjmení jméno	Zadaný projekt
216	VAN0376	Vantuch Lucian	6
217	VAS0271	Vašulín Jan	1
218	VAV0280	Vavřinec Šimon	2
219	VEC0101	Večeřa Jakub	3
220	VEL0126	Velička Jakub	4
221	VER0086	Verner Jakub	5
222	VOJ0129	Vojtěch Jakub	6
223	VOJ0161	Vojtek Jakub	1
224	VOJ0163	Vojan Václav	2
225	VYL0034	Vyleťal Tomáš	3
226	WEB0020	Weber Daniel	4
227	WIS0027	Wiszczor Adam	5
228	WOJ0082	Wojak Patrik	6
229	WYB0009	Wybitul Adam	1
230	ZAJ0140	Zajíc Filip	2
231	ZAK0115	Žák Matěj	3
232	ZAM0064	Žamboch Vilém	4
233	ZAM0074	Zámostný Jan	5
234	ZEL0135	Zelman Mykola	6
235	ZEN0038	Zenchenko Artemii	1
236	ZHA0067	Zhanalinov Sanzhar	2
237	ZID0103	Žídek Adam	3
238	ZIZ0050	Žižka Vladislav Stephen	4
239	ZMI0010	Zmija Martin	5

Kombinovaná forma studia

	Login	Příjmení jméno	Zadaný projekt
1	DRA0190	Drábik Róbert, Ing.	1
2	FEI0017	Feifič Jan	2
3	HUD0132	Hudeček Jakub	1

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
4	KUP0148	Kupec Timoteus	2
5	MAC0690	Machů Zdeněk, Ing.	1
6	MAR1023	Marejka Miloš	2
7	MAT0513	Matula Daniel	1
8	PFE0013	Pfeffer Steven	2
9	STA0727	Starý Radek	1
10	SZO0046	Szotek David	2
11	ZOU0031	Zoubek Martin	1