



Kód studenta 26



8 Hranové obarvení (otázka studijního zaměření – 3 body)

Hranové k -obarvení grafu $G = (V, E)$ je přiřazení čísel (resp. barev) $1, 2, \dots, k$ hranám grafu tak, že žádné dvě hrany se společným vrcholem nemají stejnou barvu. Problém hranového barvení spočívá v nalezení hranového obarvení s nejmenším možným počtem barev.

Navrhněte $(2 - 1/\Delta)$ -aproximační algoritmus pro problém hranového barvení, kde Δ označuje maximální stupeň ve vstupním grafu.

Musí být polynomiální a provoz $2^{-1/\Delta}$

HIS ... par. kroje
SET COLOR
ROZVRHOVANIE

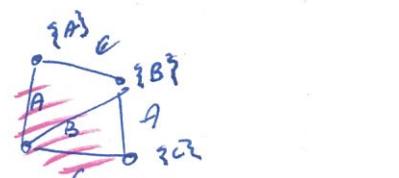
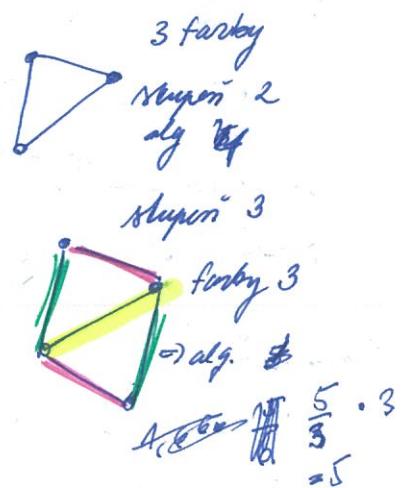
SAT

\rightarrow Najdu v řadě s najvyšším stupnem a tím můžu ofarbovat rozdílně

\rightarrow Poledom, dám novou barvu, zde si zakázní'
Barvy a tím pridám tuto barvu.

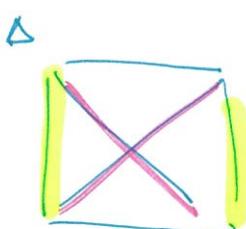
\rightarrow opět najdu s najvyšším stupnem, ofarboju, vzhledem na kalkulaci barev, ab už žádoucí nekontrolu taz dám novou a potom tento vrchol opět odobrav

\rightarrow Takto ofarbovat všechny hrany

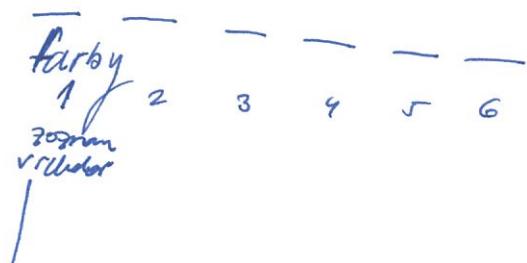


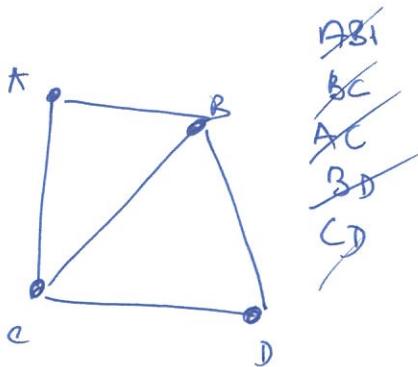
$$|E| = OPT \geq \Delta$$

$$\Delta^3$$



Chci modelovat hranu do mnoha barev, že nemají společný vrchol





$\{AB\}$ $\{BC\}$ $\{AC\}$
 $\{CD\}$ $\{BD\}$

A	B	C	D	A	B	C	D
B	C	D	A	B	C	D	A
C	D	A	B	C	D	A	B
D	A	B	C	D	A	B	C

Odgobač v kôši parkingu

Polynomialný je $\geq (k \leq |E|) \cdot O(|E|^2)$

Pre každý kôš mám zoznam ešte použitých vrcholov.

Ak hrana nespešuje žiadny tolky, teda je tam hodin. t. z. oba jej vrcholy sú v danej skupine ešte použití.
 \Rightarrow Najdem teda prvu farbu ktorú môžem istiť pre každú hrancu.

Chcem ukládať, že počet kôšov mi je väčší než $2 - \frac{1}{\Delta}$ OPT

$$ALG \leq \left(2 - \frac{1}{\Delta}\right) OPT$$

~~2OPT - 1~~

Viem, že hrany "najpočetnejšieho kôša" vytvoria aspoň 1 kôš.

$OPT \geq \Delta$, keďže by pre daný vrchol boli najaké 2 rovnale.

keď vytvoríme nový kôš, vieme, že pre tento predchádzajúci obsahovali ešte 1 vrchol. (INAK BY SME HRANU UŽ NIEKAM PRIDALI)
 takže, ich bolo maximálne $\deg A + \deg B$ pre hrancu AB
 $\deg A - 1 + \deg B - 1$
 (keď 1 hrana medzi nim máme bežom)

Pri pridávaní novej hrany je počet aktuálnych kôšov

Pri vytváraní nového kôša je počet kôšov $\leq \deg A + \deg B - 2$

$$\leq 2 \cdot \Delta - 2$$

Pri pridávaní poslednej hrany

$$ALG \leq 2\Delta - 2 + 1 \rightarrow \text{novy kôš}$$

$$2\Delta - 1 \leq 2OPT - 1 \leq 2OPT \dots \text{jedna } 2 \text{ approximation}$$

✓

3



2 body

Kód studenta 26



7 Kódy (otázka studijního zaměření – 3 body)

Definujte pojem minimální vzdálenosti pro samooprávné kódy.

Uvažme binární lineární kód $C \subseteq \mathbb{Z}_2^8$ generovaný slovy, která mají právě 2 jedničky na sudých pozicích a právě 2 jedničky na lichých pozicích, tedy

$$C = \text{span}\{(x_1, \dots, x_8) \in \mathbb{Z}_2^8 : x_1 + x_3 + x_5 + x_7 = 2, x_2 + x_4 + x_6 + x_8 = 2\}.$$

(Uvedené sčítání jednotlivých prvků kódového slova je bráno v celých číslech.)

Určete parametry tohoto kódu (délka, velikost, vzdálenost) a sestrojte jeho kontrolní matici.

vzdálenost mezi slovami je počet karet v kt. sa slova líšia
 $= |\{i \mid x_i \neq y_i\}|$

minimálna vzdálosť je najmenšia vzdálosť medzi dvojicami názvov kódov

ale to mož byť 2^k ... možnosť dvojk...

je to $(8, 36, 2)$ -kód

velkost posetia do kt. je generovany je 2^8

At nazvané len 1 prenášateľ

Obyčajne tak sa to nazívá

Z akého miestna miestna viedz?

ak je minimálna vzdálosť? 2. Kresnenie jedného ťaťu ažom alebo pridám pozad písma jednotku miním kde zmeniť este až oči 1 ďalší

11100000 $\xrightarrow{\text{vzdálosť } 2}$ 11100100

00001111

kolko je takých slov?

kontrolná matica A dàť po prenášaní kódom nulu ak tam nebola žiadna chyba

$$y \cdot A = 0 \quad \text{pre } k \text{ kódy}$$

$$y \in \{0, 1\}^8$$

$$A \in \mathbb{Z}^{8 \times 8}$$

Kolko je takých slov, že ~~síčet~~
~~pocet~~ na párych je 2 a na nepárych je 2

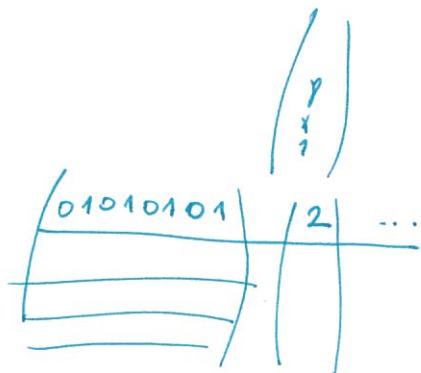
$$\binom{4}{2} \cdot \binom{4}{2} \neq \frac{4 \cdot 3 \cdot 2 \cdot 1}{2 \cdot 2} = \underline{\underline{64}}$$

36? K čemu je toto číslo

vn m říká f.f = 64
úber jmeník na nezářich poziciach

úber 2sídel + 180° se 4 jmeníkům jde se slyšit nula.

yb Ag



$(01010101)^T$ kontroly, či je síčet 2 — pouze možnou' ponu, kde cifra
následuje (10 10 10 10)



Kód studenta 26

9 Incidence bodů a přímek (otázka studijního zaměření – 3 body)

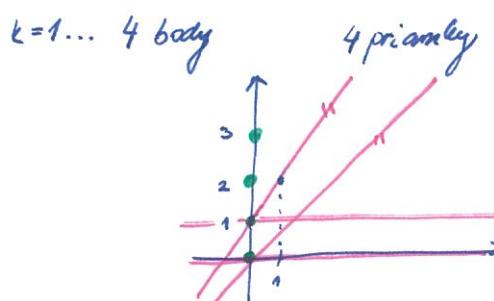
Májme k přirozené číslo a májme dánou množinu $4k^4$ bodů v rovině

$$B = \{0, 1, \dots, k-1\} \times \{0, 1, \dots, 4k^3 - 1\}.$$

Dále májme množinu $4k^5$ přímek P obsahující všechny přímky s rovnici $y = ax + b$, kde $a \in \{0, 1, \dots, 2k^2 - 1\}$ a $b \in \{0, 1, \dots, 2k^3 - 1\}$.

1. Určete počet incidencí B a P .

2. Máže mít systém $4k^4$ bodů a $4k^5$ přímek asymptoticky více incidencí než systém (B, P) výše? (Pokud k odpovědi potřebuje nějaké tvrzení (větu) ze sylabu přednášky Kombinatorická a výpočetní geometrie I, tak toto tvrzení formulujte a odvodte, jak z něho řešení plyne. Nemusíte ale tvrzení dokazovat.)



$$\begin{aligned} y &= ax + b & a \in \{0, \dots, 1\} \\ y &= x + b & b \in \{0, \dots, 1\} \\ y &= x + 1 \\ y &= 0 & [0,1], [1,2] \\ y &= 1 \end{aligned}$$

každá přímka prochází každými
nedělenými rozměry, tedy body
místo v celočíselných souřadnicích

\Rightarrow každá přímka jedna $\Rightarrow 4$ množství

$$4 \cdot 8 - 1 = 31 \cdot 2 = 62$$

$k=2 \dots$ málo bodů ... příliš velká \Rightarrow

chánu nášit, kolik bodů každá přímka má.

$$y = ax + b \quad \text{ má bod } \cancel{\{0,1\}} \cup \{1\} \\ [0,b] \cup [a+1, a+b] \text{ a dále celočíselné}$$

Body nám tvoří množinu měřitelné $k \times 4k^3$

Pokud máme přímky.

Pro tě, že $a=0$, tedy každá z nich je incidentní s k body:

$$(2k^3 - 1) \leq 4k^3 - 1, \text{ takže i ty "vyšší" stále body}$$

pro $a=1$ má to rozměry přímky, které jdou na jednotku x o jednotky "y" hore

$$\text{t.z. že za } k-1 \text{ v } x \text{ máme v } y = (1 \cdot (k-1)) + b \Rightarrow y = k-1 + b \leq k-1 + 2k^3 - 1$$

takže máme k bodů

$$= 2k^3 + k - 2 \leq 4k^3 - 1$$

$$k-2 \leq 2k^3 - 1$$

$$k-1 \leq 2k^3 \dots \text{duo}$$

$a=0 \dots k$ bodov na každej priamke

$a=1 \dots y=ax+b$

$$y \geq 1 \cdot 2k - 1 \cdot 2k^3 - 1 \geq 2k^3 - 1$$

$$k = 2k^3 - 1$$

každá z nich má ešte prekážku k bodom

$a=2 \dots y=2x+b$

$$y = 2(k-1) + 2k^3 - 1 \stackrel{?}{=} 4k^3 - 1$$

$$2k^3 - 1$$

$$\Rightarrow y = a(k-1) + 2k^3 - 1 \stackrel{?}{=} 4k^3 - 1$$

$$ak - a + 2k^3 - 1 \stackrel{?}{=} 4k^3 - 1$$

$$ak - a \stackrel{?}{=} 2k^3$$

$$2k^3 - ak - a \geq 0$$

$$k \in \mathbb{N}$$

$$2k^3 - a(k-1) \geq 0$$

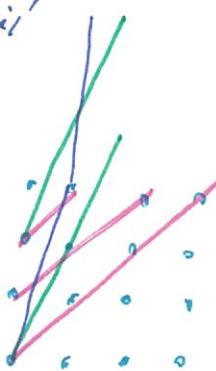
\Rightarrow Tabuľky sú najvyššie položenia a najistomšia predne k bodom (1 bod za každú posuň jednotku doprava)

\Rightarrow každá priamka predne k bodom (je incidentná k bodom)

\Rightarrow dohody sú teda $k \cdot (\text{počet priamok})$

$$= k \cdot (2k^2) \cdot (2k^3) \text{ incidentných}$$

$$= 4k^6$$

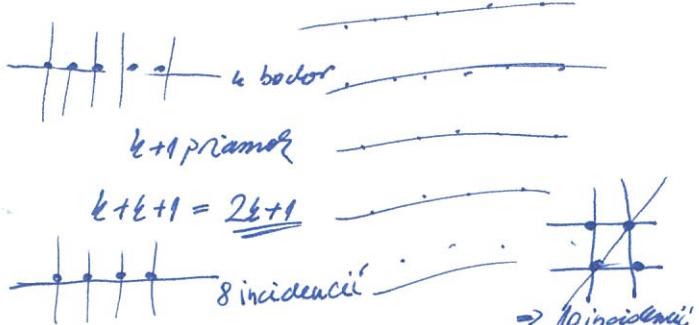


? Incidence ME

je to množina a teda

každý bod sa rada
maximálne rada?

Taz najvyššie bodne



Tie priamky rovnejne vytvárajú
snáškum sa rádiť, či je tam dokončené
bodov, aby aj najvyššie položenia
najistomšia mala dokončené

$$\begin{array}{c} \text{bodov} \\ \text{or} \\ \text{predny bod } b \end{array} \quad \begin{array}{c} \text{norná rada} \\ \uparrow \end{array}$$

$$(2k^2 - 1) \cdot (k-1) + 2k^3 - 1 \stackrel{?}{=} 4k^3 - 1$$

$$\begin{aligned} 2k^3 - 2k^2 - k + 1 + 2k^3 - 1 &\stackrel{?}{=} 4k^3 - 1 \\ \cancel{2k^3} - \cancel{2k^2} - \cancel{k} + 1 &= 0 \end{aligned}$$

$$k \text{ je ravan } 1$$

$$2k^2 + k - 1 \geq 0$$

$$\Delta = 1 + 8 = 9$$

$$\frac{-1 \pm 3}{4} \quad \begin{array}{c} \frac{1}{2} \\ -1 \end{array}$$

$$(x - \frac{1}{2})(x + 1)$$

$$x_1 = -\frac{1}{2}, x_2 = 1$$

$$\begin{array}{c|ccccc} + & & - & & + \\ \hline & + & & - & + & + \\ & -1 & & \frac{1}{2} & & \end{array}$$

Takže to platí pre $\forall k \in \mathbb{N}$

3

Homework

2. $4k^4$ bodej

$$\begin{array}{c} \cdot \\ \vdots \\ \cdot \\ \vdots \\ \cdot \\ \cdot \end{array} \left. \begin{array}{c} \cdot \\ \vdots \\ \cdot \\ \vdots \\ \cdot \\ \cdot \end{array} \right\} 2k^2 \\ \underbrace{\quad}_{2k^2} \end{array}$$

$4k^5$ priamuž

$$2k^2 + 2k^2 = 4k^2$$

$$2(2k^2 \cdot 2k^2) + 8k^4$$

$$\Rightarrow 2k^2 \cdot 4k^2 = \underline{\underline{8k^4}}$$

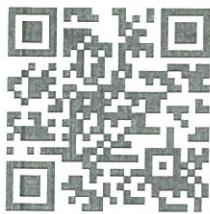
k^3 priamuž \rightarrow kôdží max $2k^2$ ináč

$$k^3 \cdot 2k^2 = 2k^5$$

$$\Rightarrow 8k^4 + 2k^5 = \dots k^5 \dots$$
 eška $\beta \beta \beta$:)

Existuje reba, iš to zhora obmedzuje JAKÝ Z CHYBÍ ZNÉMÍ

komplexum s, ke ich môže byť využitých viac.



Kód studenta 26

6 Optimalizační metody (3 body)

Nechť $Ax \leq b$ je systém lineárních nerovnic o n proměnných. Vynásobením každé nerovnosti kladnou konstantou můžeme docílit, že první sloupec matice A je vektor obsahující pouze složky 0, -1 and 1. Systém $Ax \leq b$ můžeme tudíž ekvivalentně zapsat jako

$$\begin{aligned} a'_i x' &\leq b_i \quad (i = 1, \dots, m_1), \\ -x_1 + a'_j x' &\leq b_j \quad (j = m_1 + 1, \dots, m_2), \\ x_1 + a'_k x' &\leq b_k \quad (k = m_2 + 1, \dots, m), \end{aligned}$$

$\left\{ \begin{array}{l} i+j+k \text{ rám} \\ m \text{ rám} \end{array} \right.$

kde $x' = (x_2, \dots, x_n)$ a kde a'_1, \dots, a'_n jsou řádky A bez první složky. Pak se můžeme zbavit x_1 : dokažte, že systém $Ax \leq b$ má řešení právě když systém

$$\begin{aligned} a'_i x' &\leq b_i \quad (i = 1, \dots, m_1), \\ a'_j x' - b_j &\leq b_k - a'_k x' \quad (j = m_1 + 1, \dots, m_2, k = m_2 + 1, \dots, m) \quad (m_2 - m_1) \cdot (m - m_2) \\ \underbrace{a'_1}_{\leq x_1} x' &\leq b_k - a'_k x' \end{aligned}$$

má řešení. Ukažte, že opakováním použití tohoto kroku je možné vyřešit systém lineárních nerovnic (nebo dokázat, že systém nemá řešení). Jaká bude časová složitost takového postupu?

A' ... po přemazání
bez první řádky

$$\begin{aligned} a'_j x' + a'_k x' &\leq b_j + b_k \\ a'_j x' - b_j &\leq b_k - a'_k x' \end{aligned}$$

~~1. krok je jednoduchý, ale je to jen začátek~~

~~je důležité mít všimnut, že počet nerovnic sa zvýší.~~

1. Nasobením nerovnic kladnou konstantou je obdobně krok 'uprava' uvedený v zadání.
2. ~~Na súťažiach nerovnice sú tiež ekvivalentné úpravy~~ → to je pravda, ale pozor, že základné pojednávanie je v zadanií "počet nerovnic".
⇒ ~~Keďže na súťažiach sú tiež ekvivalentné úpravy~~ a teda platí že $Ax \leq b$ má řešenie, teda aj po úprave má.

Riešenia sa dejú preniesť

Problém sa nám následujícím uvedením odkazuje, pričom budeme hľadať riešenie v tom "prostorisku" ktoré podľom pomernej k vytvoreniu riešenia odmenkuje výšku.

→ cez kalkulačku odstránenie

$$T(n) = 1 + \frac{1}{2} T(n-1) + n$$

Počet rovníc ale rastie exponenciálne

$$O(1 \cdot m) + O(m) + O(m^2) + O(m^3) + \dots \text{ Počet?} \\ \Rightarrow O(\underbrace{m^2 + m^3 + \dots}_{m^n}) \text{ ?}$$

Počet rovníc

$$\frac{m}{2} + \frac{m}{2} = m$$

Počet rovníc

$$\frac{m}{2} \cdot \frac{m}{2} = \frac{m^2}{4} \dots$$

$$\frac{m^2}{4} \cdot \frac{m^2}{4} = \frac{m^4}{16}$$

Teda riešenie z vyšej dimenzie
existuje. T. z. že z ochotenia x_2, \dots, x_n ktoré splňa
tie podmienky

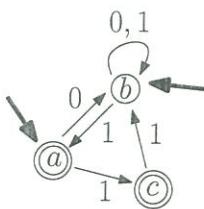
3' N



Kód studenta 26

5 Automaty (3 body)

- Převeďte nedeterministický konečný automat $A = (\{a, b, c\}, \{0, 1\}, \delta, \{a, b\}, \{a, c\})$ z obrázku na ekvivalentní deterministický automat.
- Do jakých tříd jazyků v Chomského hierarchii patří jazyk $L(A)$?

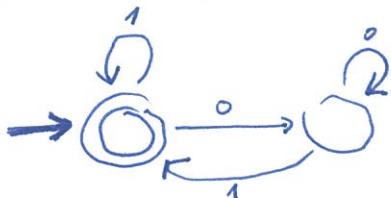


~~Kdežto je to nedeterministický (deterministický automata) kon~~

- 2.) Do třídy L_3 - regulární jazyky a další
- 1.) pojima: pravidla ktoro
jazyky súčasne končiace na 1
- vložený význam vlastnosti



Ekvivalentný automat:





3



Kód studenta 26

3 Objektový interface pro REST API server (3 body)

Mějme specializovaný web server pro REST API implementovaný v mainstreamovém objektově orientovaném staticky typovaném jazyce (C++, C# nebo Java) s následujícími vlastnostmi. Server umožňuje, aby se do něho dynamicky vkládaly služby, přičemž služba je objekt třídy implementující rozhraní **IService**, který zapouzdřuje několik REST metod. Služby jsou identifikovány řetězci, které musí být unikátní v rámci serveru, REST metody jsou rovněž identifikovány řetězci, které musí být unikátní v rámci dané služby. Součástí rozhraní třídy **IService** je i schopnost publikovat seznam svých REST metod. Souvislost mezi REST metodami a HTTP metodou volání (GET, POST) neřešíme, v dalším kontextu je metoda bez přívlastku chápána jako členská funkce třídy.

Zpracování požadavku přes REST API probíhá tak, že server přeče URL a další data z HTTP protokolu a z nich dekóduje identifikátor služby, identifikátor REST metody a kolekci parametrů. Dle identifikátorů najde ve vnitřních záznamech požadovanou službu a na ní zavolá metodu implementující požadovanou REST metodu, přičemž metoda dostane kolekci parametrů jako vstupní argument a vrátí řetězec. Pro naše účely si představme kolekci parametrů jako mapu/slovník (konkrétní typ si upravte dle zvoleného jazyka), kde klíče jsou řetězce (názvy parametrů) a hodnoty jsou objekty typu **ParameterValue** (další detaily jsou pro naši úlohu nezajímavé).

Server je reprezentován objektem třídy **Server**. Tato třída má (mimo jiné) dvě podstatné metody, které by mohly být symbolicky zapsány přibližně takto (přesný zápis závisí na použitém jazyce):

```
public void add(IService service)  
private string dispatch(string serviceId, string methodId, ParametersCollection parameters)
```

Metoda **add** zaregistrouje novou službu v rámci serveru. Tato metoda může být relativně pomalá a volá se typicky jen při startu aplikace (např. když jsou načítány zásuvné moduly). Metoda **dispatch** najde a zavolá cílovou metodu příslušné služby a je volána výhradně z vnitřní implementace serveru při zpracování požadavku ze sítě. Metoda **dispatch** by neměla mít velký overhead.

1. Navrhněte, jak by měl vypadat interface **IService** a stručně (např. komentářem) vysvětlete význam jednotlivých metod (pokud není naprosto zřejmý z názvu metody).
2. Představte příklad konkrétní třídy implementující rozhraní **IService**. Zaměřte se zejména na realizaci části rozhraní zodpovědné za předání informací o nabízených metodách.
3. Stručně (případně v pseudokódu) popište, jak bude vypadat implementace metod **add** a **dispatch** třídy **Server**. Pokud tyto metody potřebují přistupovat ke členským proměnným třídy, popište tyto proměnné také. Připomeňme, že metoda **dispatch** by měla být rozumně efektivní.

Drobné chyby v syntaxi budou tolerovány, avšak celkový návrh a logika rozhraní by měly obecně odpovídat principům a zvyklostem zvoleného jazyka. V jazycích, které to umožňují, je povoleno rozumné použití reflexe, nicméně reflexe není nutně vyžadována.

interface IService {

 public ~~void~~ IList<string> AvailableMethods();
 ^{? chybno opatřeno?}

 public Dictionary<string, Func<ParametersCollection, string> AvailableMethods();

 public string RandomGetMethod(Par. Collection params)

}

```

interface IService {
    public List<string> GetAvailableMethods();
    public string Process(string methodId, Pa. Coll. Params);
}

```

→ to je třída „Factory“, kt. na základě mezin
vypočte, kt. bude zavolat

```

public class CarService : IService {
    private Dictionary<string, Func<Params, string> methods = new Dictionary<...>();
    public List<string> GetAvailableMethods() {
        return "methods": newP
    }
}

```

```

interface IService {
    public List<string> GetAvailableMethods();
    Get public string ServiceName; Service Name
}

```

```

public class CarService : IService {
    private string - serviceName; *
    public List<string> GetAvailableMethods() {
        methods. keys(). ToList(); ... nicí takto vlastní verze
        return the - methods;
    }
}

```

~~presne symbol~~

[Get]
 public string GetCar(Params params) {

return "Shiny car";

public string GetServiceName() {
 return - serviceName;

public CarService (string serviceName) {
 - serviceName = serviceName;

}

* private readonly Dictionary<string, Func<ParamsColl., string>> = new Dictionary<string, Func<ParamsColl., string>>();
 methods

```

    "GetCar": GetCar,
    "BuyCar": BuyCar,
    :
}
    
```

tu budem registrat k metodám
 Normálne by som využíval reflexiu a hľadanie
 public methody, ktoré sú označené
 alebo [Get("názov")]
 alebo [Post("názov")]

a je toto vytvoriť tento zoznam,
 takže by sa generoval skôr

main niekde základ
 don't slovíť

public void add(IService service)

Ste var name = service.GetName();
 1.) kontroly, či už ju obsahuje
 a ak už vytvoril užívateľa, že
 už tam je.

"CarService" → ("GetCar", GetCar)
 "BuyCar" → ("BuyCar", BuyCar)
 "ShopService" → ("IsOpened", ...)

2) Services[{"name": "CarService"}]
 kontroly, či tento názov má metódu

3) Príďme do slovníka
 dict.Add("name", service.GetAllMethods());

public string dispatch(string serviceId, string methodName, ParamsColl params)

```

    try {
      var f = dict[serviceId][methodName];
    }
    catch (NotFoundException e) {
      ...
    }
  
```

return f(params);
 ↓
 toto si nie som istý
 ale vyslo sa rotačne
 ale nijako sa to dešte

3 body



Kód studenta 26

4 Souborový systém (3 body)

Tato otázka obsahuje zjednodušený popis souborového systému FAT. Pokud chcete, můžete v odpovědi uvažovat i skutečný souborový systém FAT (pokud ano, výslově to napište).

Oddíl naformátovaný (zjednodušeným) souborovým systémem FAT12/16/32 je rozdelený na 3 části:

1. boot sektor (nultý logický sektor oddílu),
2. tzv. tabulka FAT, viz dále (1. až N. sektor oddílu),
3. datové sektory (N+1. sektor až poslední sektor oddílu), které obsahují samotná data souborů (nebo adresářů, nicméně jelikož ty se z pohledu alokace sektorů od souborů v souborovém systému neliší, tak se jimi dále nebudeme zabývat zvláště). Pro jednoduchost předpokládejme, že alokační jednotka souborového systému je právě jeden sektor disku. Sektoru disku mají jednu z obvyklých velikostí – označme ji jako X bytů.

Každý datový sektor oddílu je přiřazený právě jednomu souboru, nebo je označený jako volný. Každý soubor na disku zabírá určité množství celých sektorů, přičemž ale tyto sektory nemusí být konkrétnímu souboru přiděleny kontinuálně (soubory mohou být na disku fragmentované). V adresářovém záznamu pro konkrétní soubor je zapsáno číslo datového sektoru (číslovány od 1), který obsahuje data prvních X bytů souboru (pro soubory s velikostí menší nebo rovnou X bytů je to zároveň poslední sektor souboru). Pro soubory s velikostí větší než X bytů nalezneme informaci o dalších sektorech přidělených souboru ve FAT tabulce oddílu – pro každý soubor (který zabírá např. M sektorů) obsahuje FAT tabulka „zakódovanou“ obdobu jednosměrně vázaného seznamu posloupnosti čísel 2. až M. sektoru souboru. Přesný obsah FAT tabulky je následující:

Pro každý datový sektor obsahuje FAT tabulka právě jeden Z bitový záznam ($Z = 12$ bitů pro FAT12, 16 bitů pro FAT16, 32 bitů pro FAT32) – tento záznam je **bezznaménkové Z bitové celé číslo uložené v pořadí little endian**. V tabulce FAT je tedy právě tolik záznamů, kolik oddíl obsahuje datových sektorů, a žádné jiné informace FAT tabulka neobsahuje (FAT tabulka je tedy fakticky jen pole celých čísel). Záznam na offsetu 0 v prvním sektoru FAT tabulky obsahuje informaci o 1. datovém sektoru oddílu, hned za ním (bez paddingu) následuje záznam pro 2. datový sektor oddílu, pak záznam pro 3. datový sektor oddílu, atd. Pokud záznam pro datový sektor A obsahuje číslo 0, tak je tento datový sektor volný a není přidělen žádnému souboru. Pokud záznam pro datový sektor A obsahuje číslo B, tak je datový sektor A přidělený nějakému souboru, a po X bytech dat toho souboru uložených v datovém sektoru A je následujících X bytů souboru uložených v datovém sektoru B (kde pro fragmentované soubory nemusí být B rovno A+1, a obecně může být B být větší i menší než A). Pokud záznam pro datový sektor A obsahuje maximální hodnotu Z bitového čísla, pak je sektor A posledním sektorem nějakého souboru (M. sektorem přiděleným souboru o velikosti M sektorů). Pokud tedy např. data souboru A.TXT budou ležet v datových sektorech 10, 7, 8, 15, tak v adresářovém záznamu pro A.TXT bude zapsáno číslo 10 a ve FAT tabulce bude v záznamu pro sektor 10 uloženo číslo 7, v záznamu pro sektor 7 číslo 8, v záznamu pro sektor 8 číslo 15, a v záznamu pro sektor 15 maximální hodnota Z bitového čísla.

1. Jaká může být obvyklá hodnota X? Pokud je velikost oddílu právě 1 GiB, je vhodné tento oddíl naformátovat souborovým systémem FAT16 nebo FAT32? Vysvětlete proč.
2. V takovém 1 GiB oddílu máme uloženo 100 000 souborů, kde každý má velikost 1 KiB. Pokud nyní chceme přečíst obsah všech těchto souborů, bylo by pro typický disk vhodné načíst si předem celou FAT tabulku do paměti? Nebo by bylo vhodnější před čtením každého ze souborů znova načíst do paměti pouze ty sektory FAT tabulk, o kterých zjistíme, že obsahují informace potřebné pro právě čtený soubor? Vysvětlete proč.
3. Předpokládejte, že v „globální proměnné“ (resp. statické proměnné nějaké třídy) **fat** typu pole bytů máme načtené veškeré záznamy celé FAT tabulky oddílu naformátovaného souborovým systémem FAT12 – jeden záznam FAT tabulky má velikost 12 bitů, tedy každé 3 byty pole **fat** obsahují právě 2 záznamy o 2 datových sektorech (záznam pro sektor s nižším číslem je vždy v 1. bytu a ve spodních 4 bitech 2. bytu; záznam pro sektor s vyšším číslem je ve vyšších 4 bitech 2. bytu a ve 3. bytu). V jazyce C# nebo Java nebo C++ naprogramujte proceduru, která jako svůj jediný argument dostane číslo prvního datového sektoru přiděleného nějakému souboru, a má na standardní výstup vypsat čísla všech datových sektorů tomuto souboru přiřazených (dle informací v proměnné **fat**).

1.) 4 kilobyty ✓

$2^8 \dots 256$

$2^{10} \dots 1024$

$2^{12} \dots 4096$

$2^{12} \dots$

→ musí to byt' dostatočne male', aby následující rekapitulaci množstva
a kde prázdný disk by sa tránil ako plný

→ a ideálne dosť "veľká" aby pri načítaní písal celý súbor a nie veľa malých

16iga → FAT 82 alebo FAT 16?

→ chcem aby sme mohli využiť väčší priestor

→ t.z. keď "ešte nie sme išli", takže radšej väčší záber my $\Rightarrow z=32$

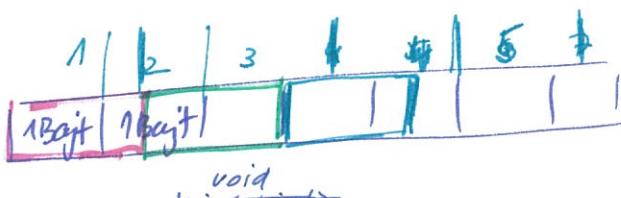
~~z=32~~

$$16iB = 16 \cdot 2^{10} kB = 2^{10} \cdot 2^{10} kB = \frac{2^{20} kB}{2^2 kB} = 2^{18} \text{ možných záberov}$$

\Rightarrow t.z. že by nám FAT 16 nemal stačiť ✓

2.) $10^5 kB = 10^2 MB = 100 MB$? ✓

Ta' tabuľka je "male". Pretože pre disk je lep' 0 naj' aj vieme, že
je enviroment uložený. Pretože som uvedla najprv načítanie až počom
už kedyči dalsie informacie sú disku položky na data a nie na tabuľka dát...
A hľadame by sme ju získali 100 000 kast s dátami, čo nie je malo



public int GetAllSectors (int firstSectorId) {

~~Console.WriteLine(firstSectorId);~~

~~if (firstSectorId == 0) {~~

~~firstSectorId += 1;~~

~~firstSectorId / 2~~

~~firstSectorId / 2~~

~~((2 * firstSectorId) + 1) / 2~~

* ~~int zero = 0;~~
~~int nextSector = GetNextSectorId (zero + data);~~

LIBK bytez tap

~~probabilny~~
je tiež

int GetNextSectorId (int data) {

return data;

}

if (nextSector == (int) (1 << 12))) {
 Console.WriteLine(nextSector);
 return;

GetAllSectors (nextSector);

new 'zavŕšené' rozvinutá tail recursion → pro veľké súborov
⇒ Stack overflow

```

int data = 0;
if (firstSectorId % 2 == 0) {
    data += (fat[fatSectorId / 2] << 4);
    data += (fat[firstSectorId / 2 + 1] >> 4);
} else {
    data += 2 * (fat[firstSectorId / 2] << 4);
    data += fat[firstSectorId / 2 + 1];
}
int prevBajt = sectorId / 3 + (long * 4); (sectorId / 3 + (long * 4));
int dunkyBajt = prevBajt + 1;
int data = 0;
if (Lavy) {
    data += fat[prevBajt] << 4;
    data += fat[dunkyBajt] >> 4;
} else {
    data += (fat[prevBajt] << 4) << 4;
    data += fat[dunkyBajt];
}

```

1,2...0
 1,2,3
 2,3,4
 3,4,5
 4,5,6
 5,6,7
 6,7,8
 7,8,9
 8,9,10

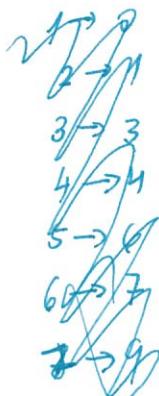
3,4,...2,3
 5,6,...6

bool Lavy = (sectorId % 2 == 0);
 if (!Lavy) sectorId --;
 if (Lavy) sectorId++; prevBajt++;

Väistävä arviontulku ja little-endian

tulos tuloks postilleday int
 by sa mal rivevine aloitit.
 na zacíatok by bolo vročne
 kontrolovat, ū je dvojicu
 little endian
 op "real" třeba jy

1	2	3	4	5	6	7
9	1	2	3	4	5	6



$$3 - b$$



Kód studenta 26



2 Databáze (3 body)

- Načrtněte, jak byste v databázové tabulce reprezentovali binární vztah M:N. Lze z definice tabulky $T(FK_1, FK_2)$, vzniklé převodem binárního vztahu, určit jeho původní kardinalitu (1:1, 1:N, M:N)? Vysvětlete.
 - Uvažujte transakce $T_1: W(A) R(B) W(C)$ a $T_2: R(A) W(B) R(C)$. Je rozvrh $S: W_1(A) R_2(A) R_1(B) W_2(B) W_1(C) W_2(C)$ konfliktově serializovatelný (conflict serializable)? Vysvětlete proč a pokud není, navrhněte úpravu, aby byl.

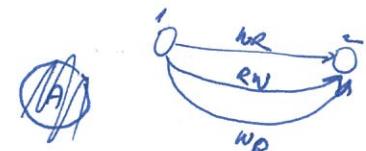
$$R_2(c)$$

K. K. R. B.

T_1	T_2	T_1	T_2
$w(A)$	$R(A)$		$w(A)$
$R(B)$	$w(B)$		$R(A)$
$w(C)$	$R(C)$		$R(B)$

wystąpią się gry konfliktowej
współczesności

WR, RW, WW



10

10

⇒ Přijí pracuje výšky před analyzou. Takže je to ekvivalentně
Tomuto se všem my rozumíme ...

T_1
 WLA)
 R(B)
 W(C)

⇒ fiktionsverarbeitung

H:N

~~H:míšloř má N~~

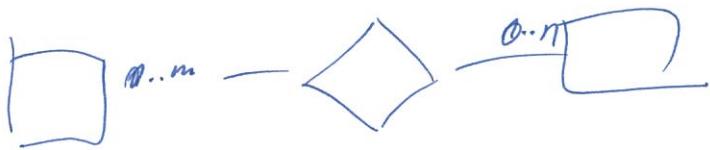
kedyž místel má mnoho studentů

kedyž student má mnoho místel

⇒ vytvořím tabulkou, kde sa každý druhý v relaci dám zápisem

#	STUDENT	UČITEL
1	1	1
2		2
3	1	3
:	1	2
i	2	2
	3	
	:	

⇒ tabuľka reprezentuje



↪ bude kľúč?

+ (Fk_1, Fk_2)

1:1 nie, na to by stačil jeden kľúč

1:N nie, to by zaradiež na základe toho pravého určilo to ľave'

✓ ⇒ je to H:N

protože $L \xrightarrow{\text{nevyrobne}} P$ ani $P \xrightarrow{\text{nevyrobne}} L$

tak je potrebný samostatný kľúč

BM



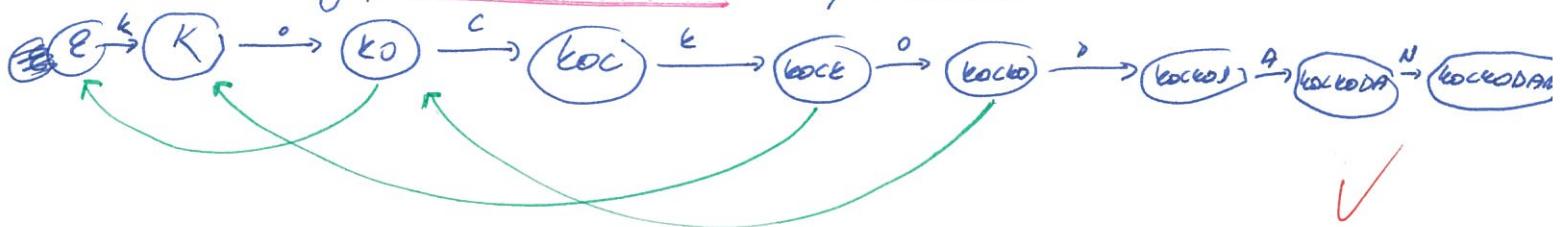
Kód studenta 26

1 Algoritmy a datové struktury: vyhledávání v textu (3 body)

- Definujte vyhledávací automat algoritmu KMP (Knuth-Morris-Pratt). Jak vypadají jeho stavy, dopředné a zpětné hrany?
- Nakreslete vyhledávací automat pro slovo KOCKODAN.
- Jakou časovou složitost má konstrukce automatu a jakou jeho použití k vyhledávání všech výskytů slova v textu?

1.) Stavy jsou uspořádány na 'jednu' část slova
dopředné hrany všech daných významových písmen dom. tedy má následovat
~~(K, KO, KOE, KOEO, KOEOC, KOEOCD, KOEOCDN)~~
zpětné hrany naše posunů do dříve uvedeného prefixu slova
dejte ji až zadního fixu od konca ✓

Kde mi m' hrany, tak vezm' do E! (Pre prehlednost)



- Automat se konstruuje samoběžným KMP. Teda najdete tyto řešení sloužící algoritmu:

Ten přechádku celý všechny relace (dejte n) a když je náčítané znaku se nádi automatem (toto je konstantní počet, když se má v tom posavit).

Složitost algoritmu je $O(n+m)$ tedy vratane konstrukce automatu
dejte slova \notin (vzorec)
Takže automat akonstrukuje $O(m)$ ✓

BRUTE FORCE $O(n \cdot m)$