ADSO
10. Práce s grafy

(prezentace k učebnici)

Ivan Ryant

Agenda

- Základní pojmy
- Úloha abstraktních datových typů
- Posloupnosti a operace s nimi
- Vyhledávací datové struktury
- Vyhledávací posloupnost
- Algoritmy řazení
- Stromy
- Rozptýlené tabulky
- Prohledávání do hloubky a do šířky
- **Práce s grafy**
- Techniky návrhu efektivních algoritmů

Agenda

- **Práce s grafy**
  - Další pojmy z teorie grafů
  - Reprezentace grafu
  - Úlohy na grafech
  - Komponenty souvislosti
  - Minimální kostra
  - Topologické uspořádání
  - Hledání nejkratší cesty
    - Floydův algoritmus
    - Dijkstrův algoritmus
  - Shrnutí

Agenda

- Práce s grafy
  - **Další pojmy z teorie grafů**
  - Reprezentace grafu
  - Úlohy na grafech
  - Komponenty souvislosti
  - Minimální kostra
  - Topologické uspořádání
  - Hledání nejkratší cesty
    - Floydův algoritmus
    - Dijkstrův algoritmus
  - Shrnutí

Základní pojmy z teorie grafů (opakování)

- Graf
  - Vrcholy
    - Mohou mít přiřazené ohodnocení (jméno, váhu...)
  - Hrany
    - Mohou mít přiřazené ohodnocení (délku, cenu jízdného...)
  - Incidenční funkce
    - říká, se kterými vrcholy inciduje hrana, tj. každé hraně přiřazuje
      - uspořádanou dvojici vrcholů (orientovaný graf)
      - neuspořádanou dvojici vrcholů (neorientovaný graf)
- Cesta
  - je posloupnost vrcholů spojených hranami tak, že
    - první vrchol má následníka, poslední má předchůdce, ostatní mají obojí.
  - Žádná dvojice vrcholů se neopakuje.
- Cyklus
  - je posloupnost vrcholů spojených hranami tak, že
    - všechny vrcholy mají předchůdce i následníka.
  - Žádná dvojice vrcholů se neopakuje.
- Strom je souvislý acyklický neorientovaný graf.

Další pojmy z teorie grafů

- Grafy mohou být:
  - ohodnocené
    - kde vrchol nebo hrana mají hodnotu vyjadřující např. vzdálenost, dobu nebo cenu
  - souvislé (anglicky connected)
    - kde každé dva uzly jsou spojeny cestou
  - acyklické
    - neobsahují cykly (jako např. strom)

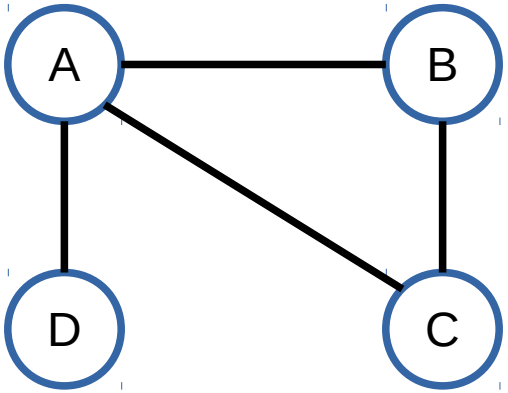
Agenda

- Práce s grafy
  - Další pojmy z teorie grafů
  - **Reprezentace grafu**
  - Úlohy na grafech
  - Komponenty souvislosti
  - Minimální kostra
  - Topologické uspořádání
  - Hledání nejkratší cesty
    - Floydův algoritmus
    - Dijkstrův algoritmus
  - Shrnutí

Reprezentace grafu

- matice sousednosti (uzlo-uzlová, anglicky: adjacency array)
  - pro hranově ohodnocené grafy, např. matice vzdáleností mezi uzly
- matice incidence (uzlo-hranová)
  - implementace typicky dvourozměrným polem
- seznam následníků
  - implementace dynamicky:
    - vrcholy např. seznamem, stromem nebo rozptýlenou tabulkou
    - hrany pomocí odkazů (referencí) nebo ukazatelů na vrcholy
  - implementace dvěma poli:
    - pole vrcholů
    - pole jejich následníků
- posloupnost hran (např. seznam dvojic vrcholů)

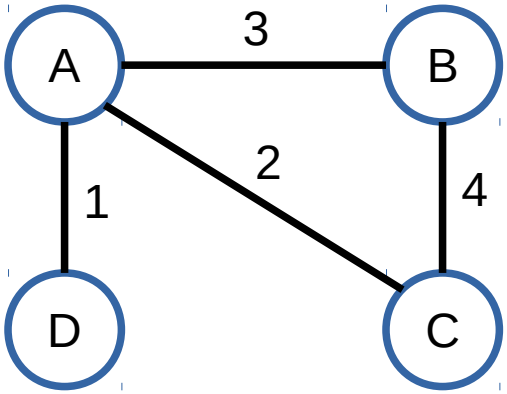
Reprezentace grafu



	A	B	C	D
A	False	True	True	True
B	True	False	True	False
C	True	True	False	False
D	True	False	False	False

Matice sousednosti

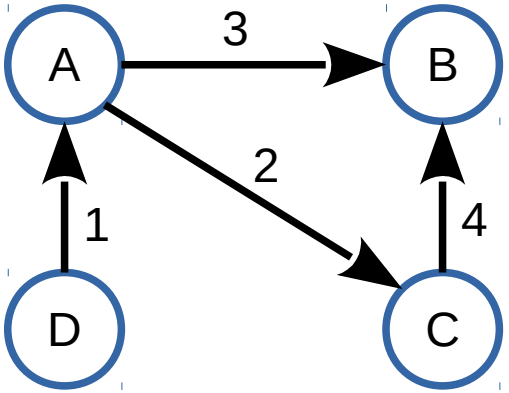
Reprezentace grafu



	A	B	C	D
A	nil	3	2	1
B	3	nil	4	nil
C	2	4	nil	nil
D	1	nil	nil	nil

Matice sousednosti hranově ohodnoceného grafu

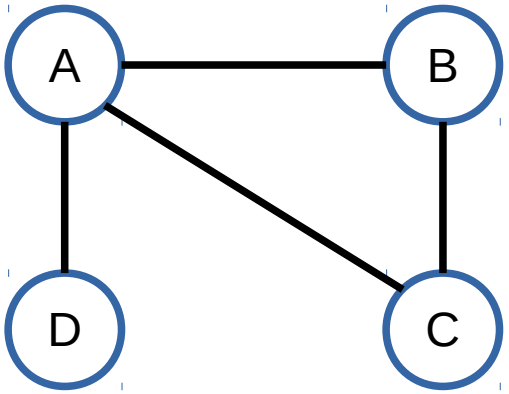
Reprezentace grafu



	A	B	C	D
A	nil	3	2	nil
B	nil	nil	nil	nil
C	nil	4	nil	nil
D	1	nil	nil	nil

Matice sousednosti hranově ohodnoceného orientovaného grafu

Reprezentace grafu

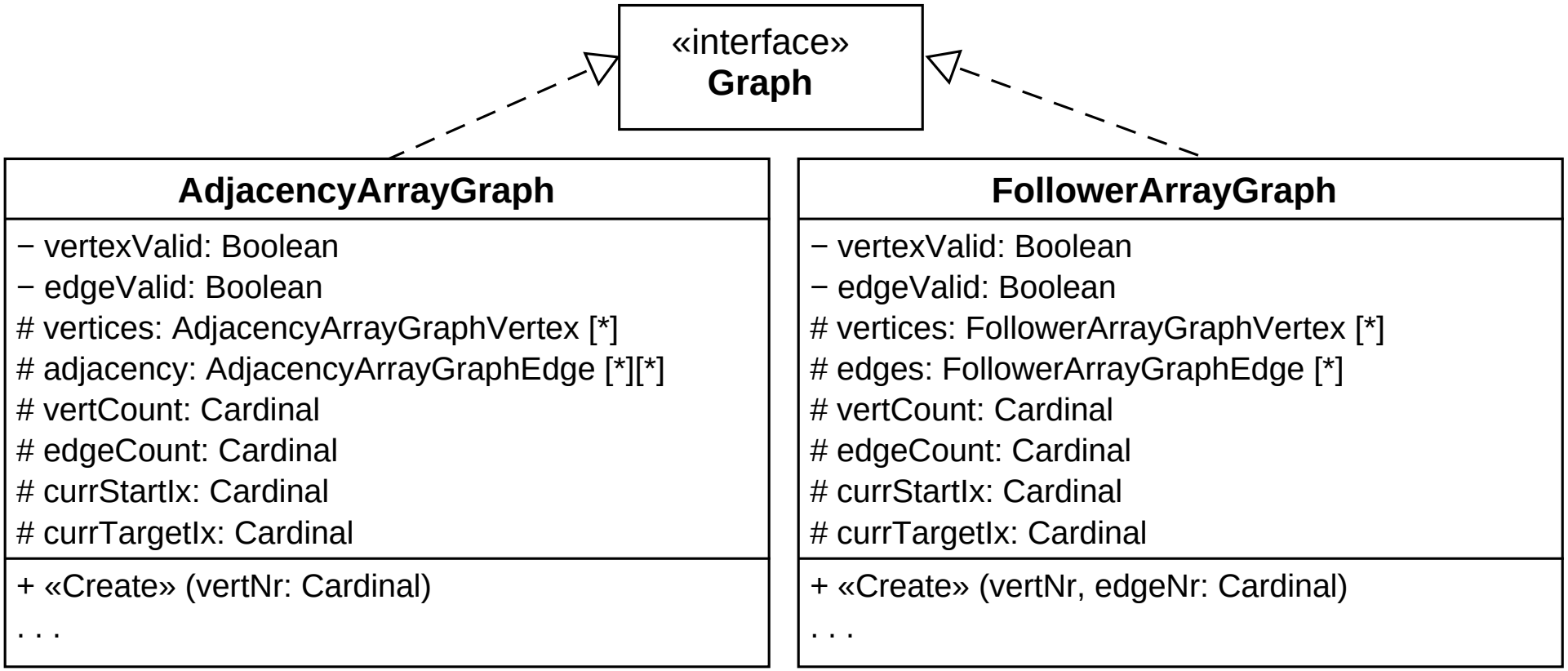


Index:	0	1	2	3
Obsah:	A, 0	B, 3	C, 5	D, 7

Index:	0	1	2	3	4	5	6	7
Obsah:	1	2	3	0	2	0	1	0

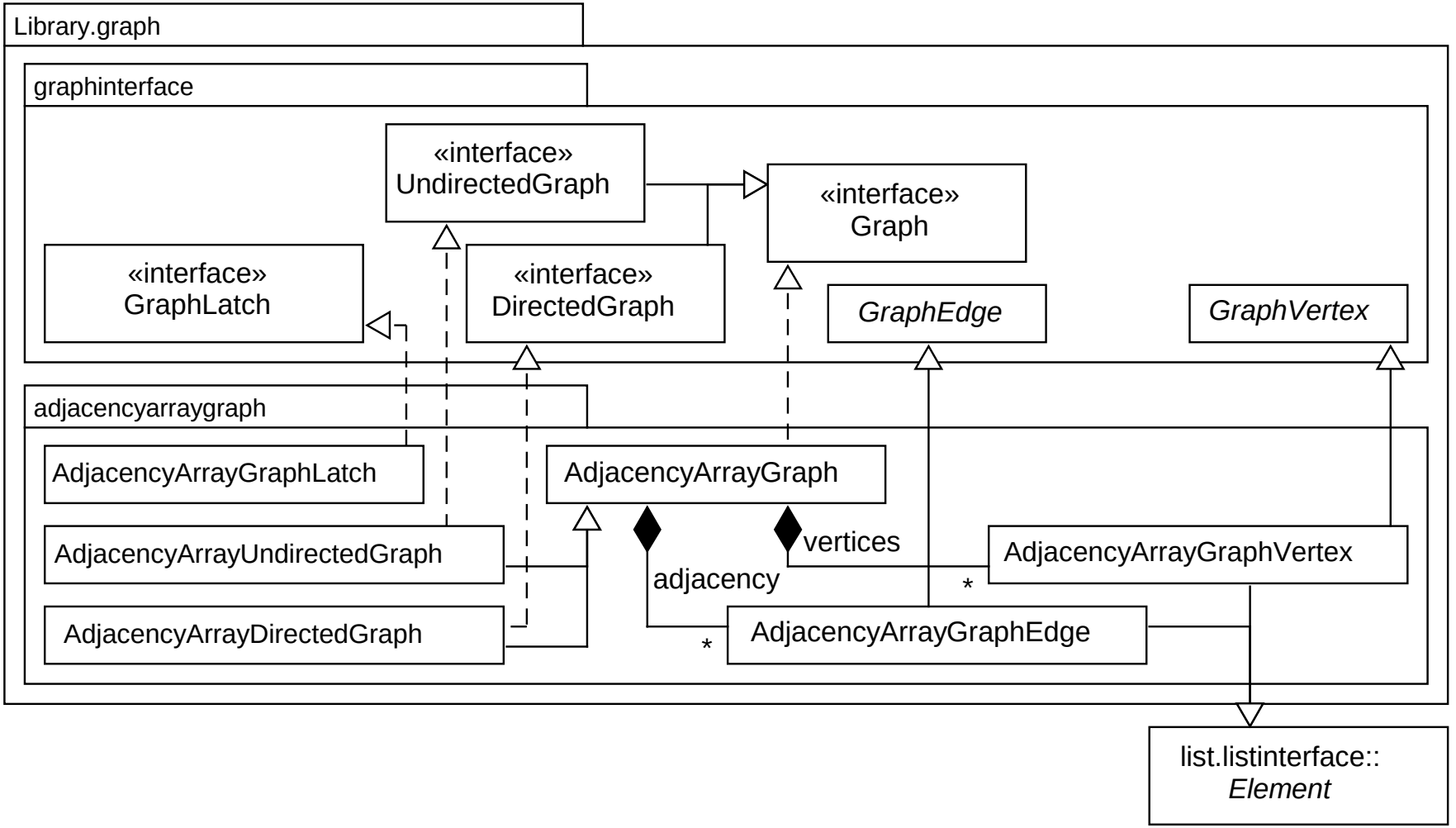
Seznam následníků

Reprezentace grafu



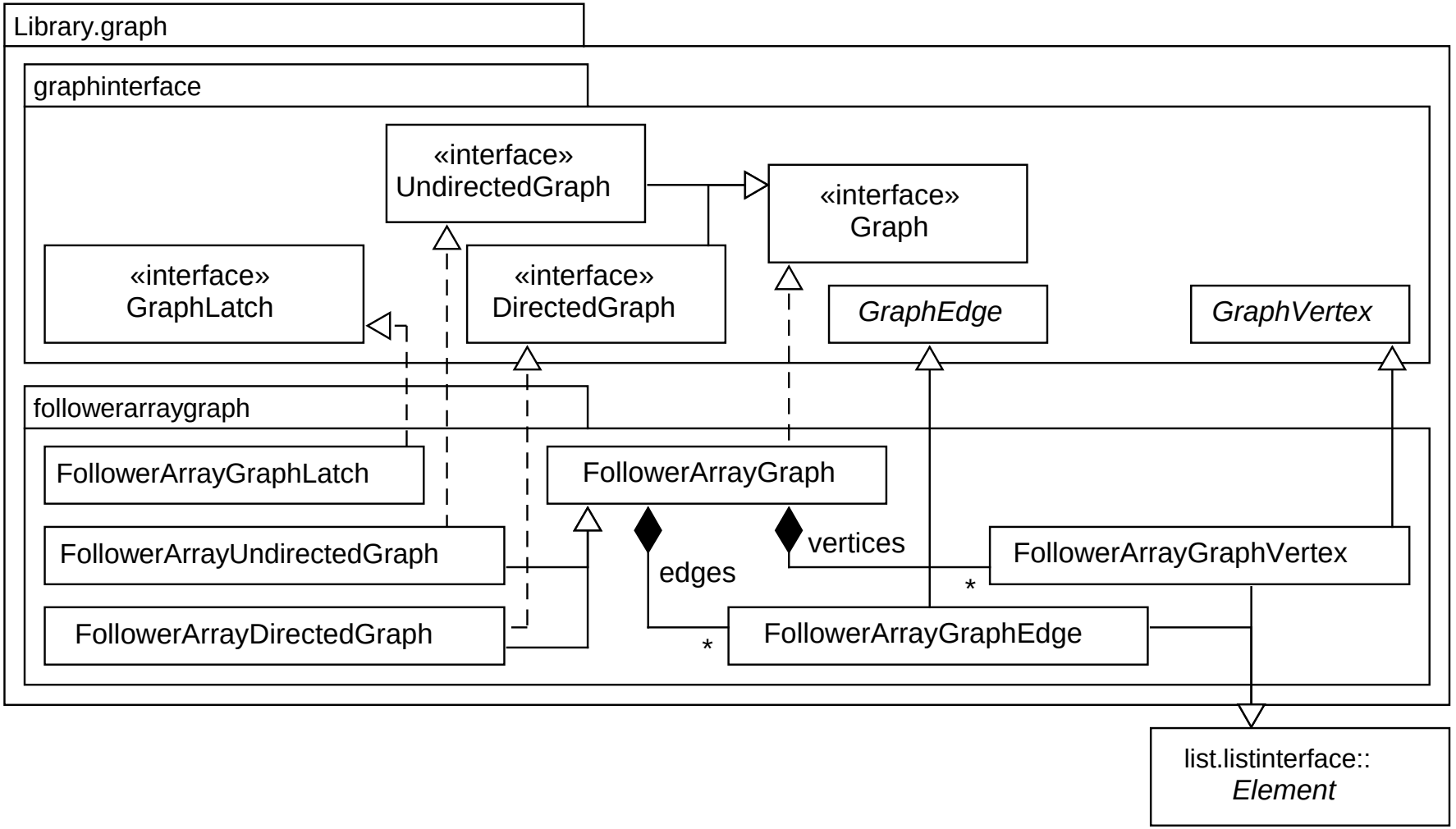
Návrh vnitřní reprezentace grafu

Reprezentace grafu



Návrh objektů, které reprezentují graf maticí sousednosti

Reprezentace grafu

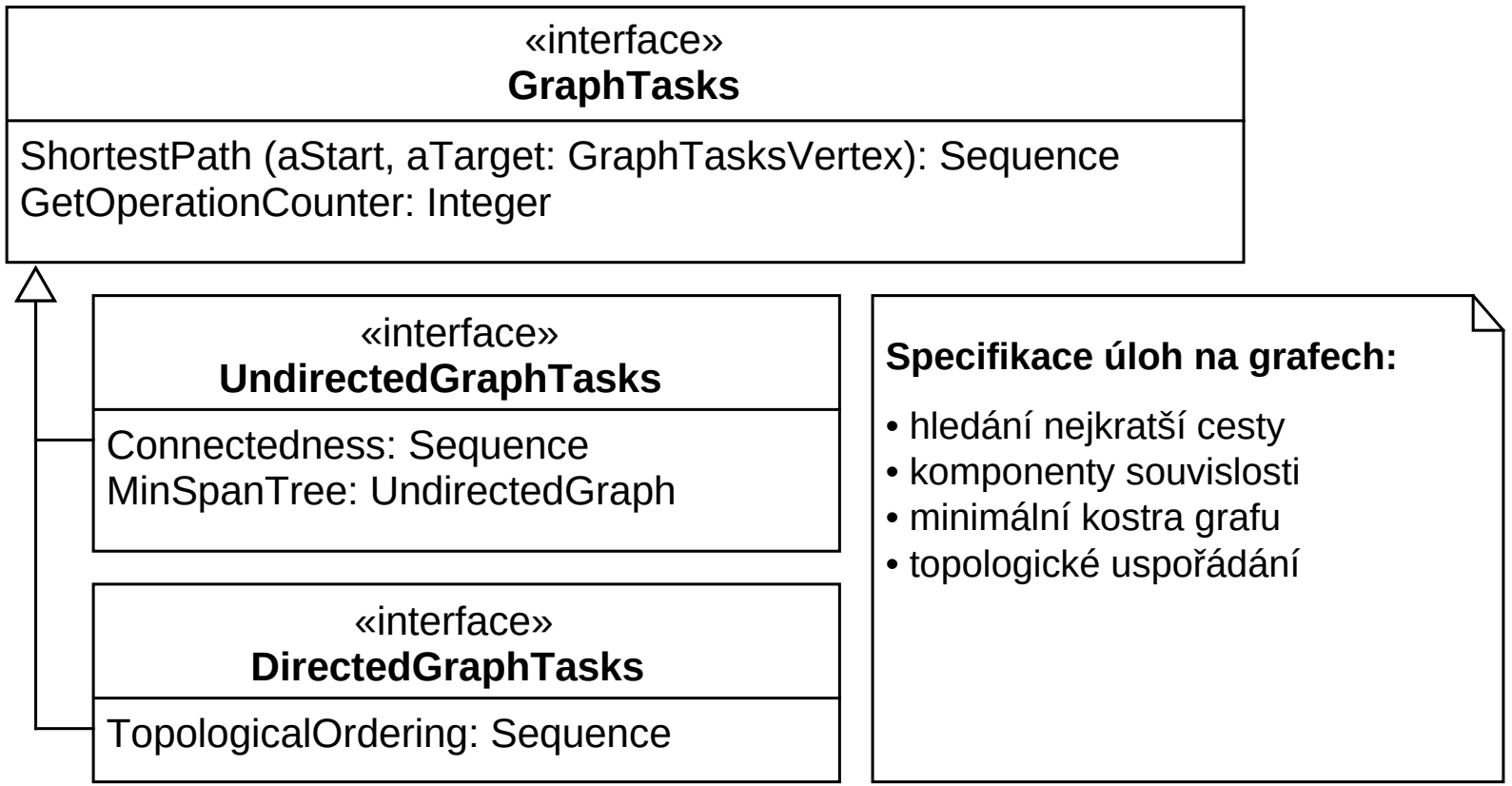


Návrh objektů, které reprezentují graf seznamem následníků

Agenda

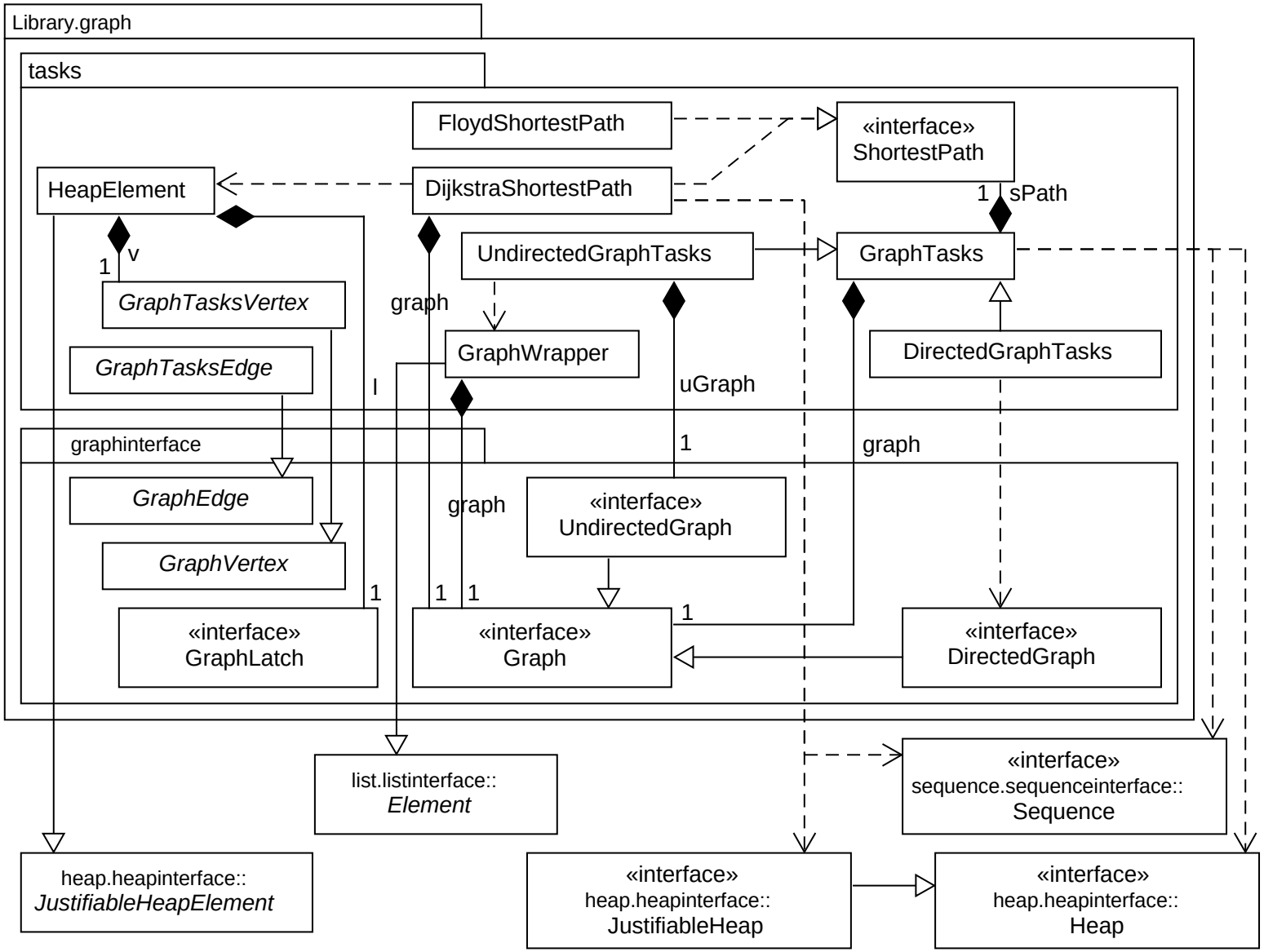
- Práce s grafy
  - Další pojmy z teorie grafů
  - Reprezentace grafu
  - **Úlohy na grafech**
  - Komponenty souvislosti
  - Minimální kostra
  - Topologické uspořádání
  - Hledání nejkratší cesty
    - Floydův algoritmus
    - Dijkstrův algoritmus
  - Shrnutí

Úlohy na grafech



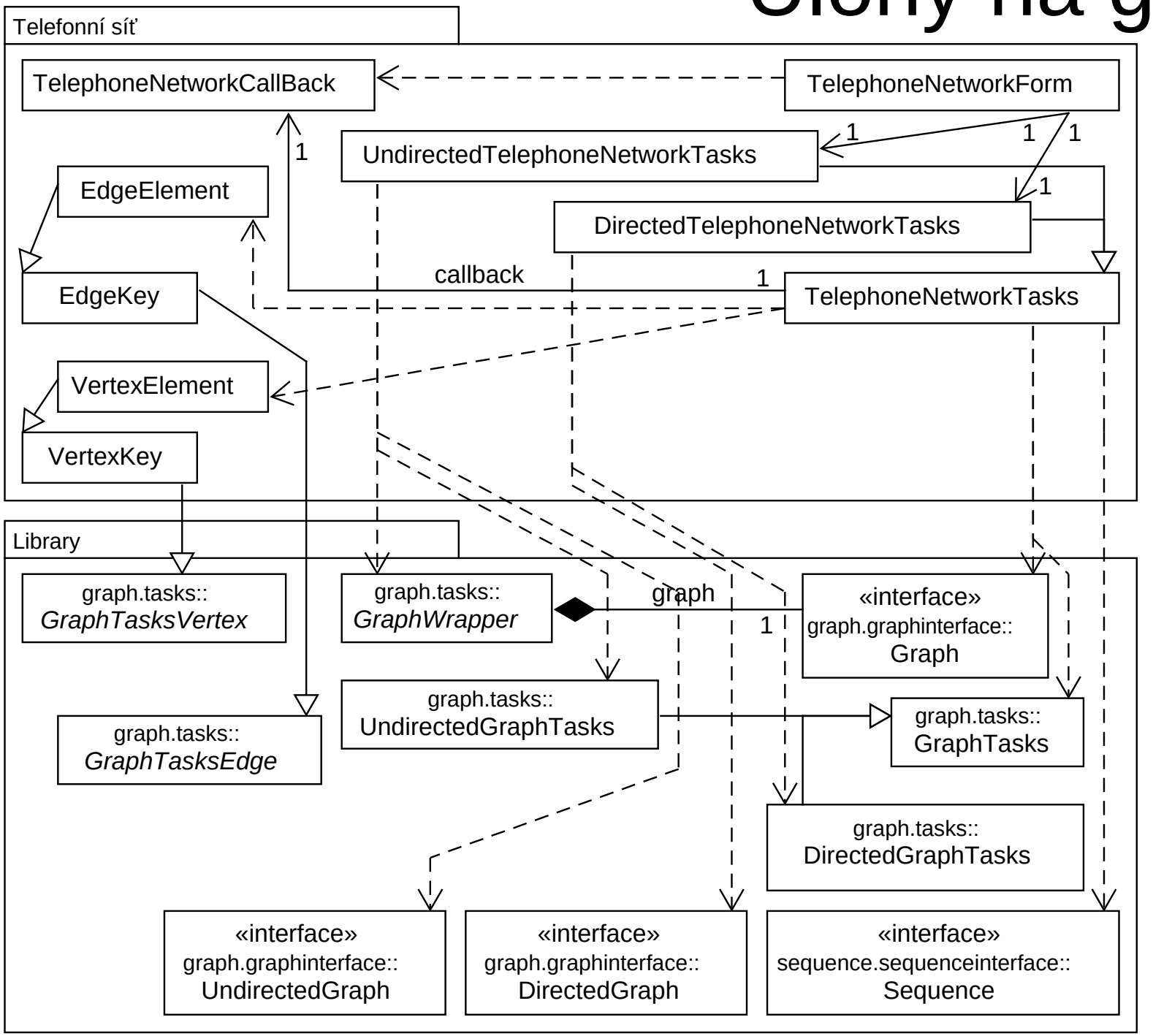
Rozhraní grafových úloh

Úlohy na grafech



Knihovna objektů pro úlohy na grafech

Úlohy na grafech



Aplikace „Telefonní síť“

Agenda

- Práce s grafy
  - Další pojmy z teorie grafů
  - Reprezentace grafu
  - Úlohy na grafech
  - **Komponenty souvislosti**
  - Minimální kostra
  - Topologické uspořádání
  - Hledání nejkratší cesty
    - Floydův algoritmus
    - Dijkstrův algoritmus
  - Shrnutí

Komponenty souvislosti

- Algoritmus
  - Každou komponentu můžeme „zmapovat“ průchodem do hloubky nebo do šířky
    - Označovat navštívené vrcholy, abychom nezabloudili v cyklech!
  - Když vyčerpáme všechny vrcholy grafu, je jasné, že graf je souvislý.
  - V opačném případě (když graf je nesouvislý) můžeme postupně „zmapovat“ jednu komponentu po druhé.

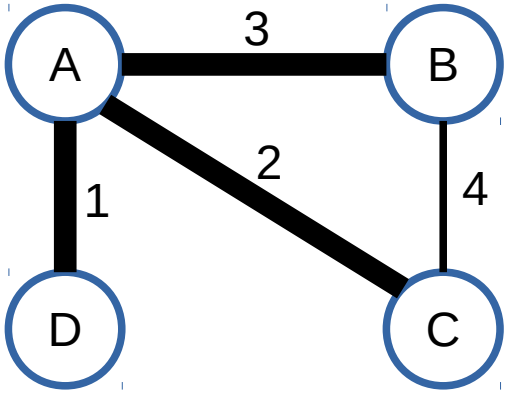
Komponenty souvislosti

- Časová složitost
  - Musíme navštívit každý vrchol grafu, některé vrcholy pak navštíví-me dokonce víckrát. Z každého vrcholu totiž hledáme dosud nena-vštívené následovníky, kam vedou hrany. Abychom zjistili, kteří následovníci ještě nebyli navštíveni, musíme bohužel prohlédnout i vrcholy již navštívené – tím se výpočetní složitost nepříjemně zvětšuje. Takže
    - Musíme projít všech *V* vrcholů grafu.
    - Z každého vrcholu pak musíme prohlédnout všechny jeho následovníky, kterých může být také až *V*.
    - V tom případě vyjde odhad časové složitosti kvadratický vzhledem k *V*.
  - Uvažujme i hrany. Při hledání následovníků procházíme každou hranu v grafu celkově jen jednou (případně jednou tam a jednou zpátky).
    - Časová složitost vyjde lineárně úměrná (*V* + *E*),
      kde *V* je počet vrcholů a *E* je počet hran.

Agenda

- Práce s grafy
  - Další pojmy z teorie grafů
  - Reprezentace grafu
  - Úlohy na grafech
  - Komponenty souvislosti
  - **Minimální kostra**
  - Topologické uspořádání
  - Hledání nejkratší cesty
    - Floydův algoritmus
    - Dijkstrův algoritmus
  - Shrnutí

Minimální kostra



- Algoritmus
  - Napřed odebereme všechny hrany z grafu a ponecháme jen izolované vrcholy. Co vrchol, to komponenta souvislosti.
  - Hrany seřadíme podle hodnot od nejmenší do největší.
  - Pak postupně vracíme hrany do grafu (začínáme od nejmenší):
    - Hranu vrátíme jen pod podmínkou, že spojí dvě nesouvislé komponenty do jedné souvislé komponenty.
  - V okamžiku, kdy se všechny vrcholy propojí do jedné komponenty, výpočet skončí.
    - Vyčerpáme-li všechny hrany, a přece se nám nepodaří vytvořit jednu komponentu, znamená to, že graf zřejmě původně nebyl souvislý (získáme minimální kostru nesouvislého grafu).
  - Na obrázku je znázorněn příklad minimální kostry grafu. Hrany, které tvoří minimální kostru, jsou tlusté.

Minimální kostra

- Hladový (greedy) algoritmus
  - Přímočarý způsob nalezení minima (nebo naopak maxima)
  - Tím, že hrany s malými hodnotami přidáváme napřed, dosáhneme toho, že graf propojíme hranami s nejmenšími hodnotami tak, že součet hodnot všech přidaných hran bude minimální možný.

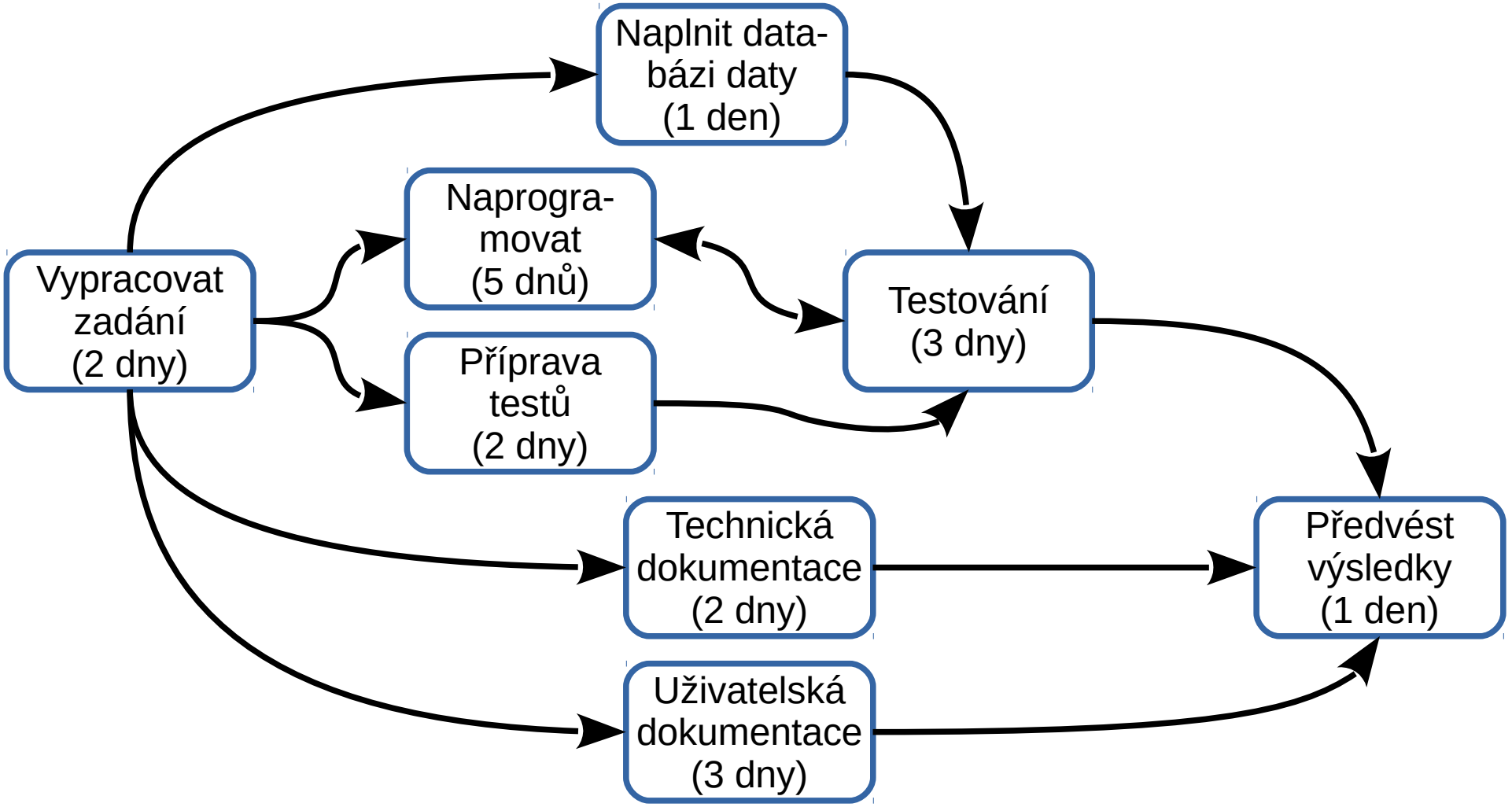
Minimální kostra

- Časová složitost
  - Do kostry postupně vkládáme hranu za hranou, celkem nejvýše E hran.
    - Každou přidávanou hranu musíme napřed vyjmout z haldy – to vyžaduje až $\log_2 E$ kroků.
    - Potom se pokusíme hranu vložit do grafu.
      - Kostru ovšem netvoří všechny hrany původního grafu, ale jen ty, které propojují vrcholy, aniž by vznikl cyklus.
      - Takových hran je $V - 1$ (V je počet vrcholů). Ve $V - 1$ případech tedy musíme přečíslovat vrcholy v jedné komponentě grafu. Počet vrcholů v komponentě může být až $V - 1$.
  - Celková časová složitost tedy nebude horší než
    - lineárně logaritmicky závislá na počtu hran
    - kvadratická vzhledem k počtu vrcholů

Agenda

- Práce s grafy
  - Další pojmy z teorie grafů
  - Reprezentace grafu
  - Úlohy na grafech
  - Komponenty souvislosti
  - Minimální kostra
  - **Topologické uspořádání**
  - Hledání nejkratší cesty
    - Floydův algoritmus
    - Dijkstrův algoritmus
  - Shrnutí

Topologické uspořádání

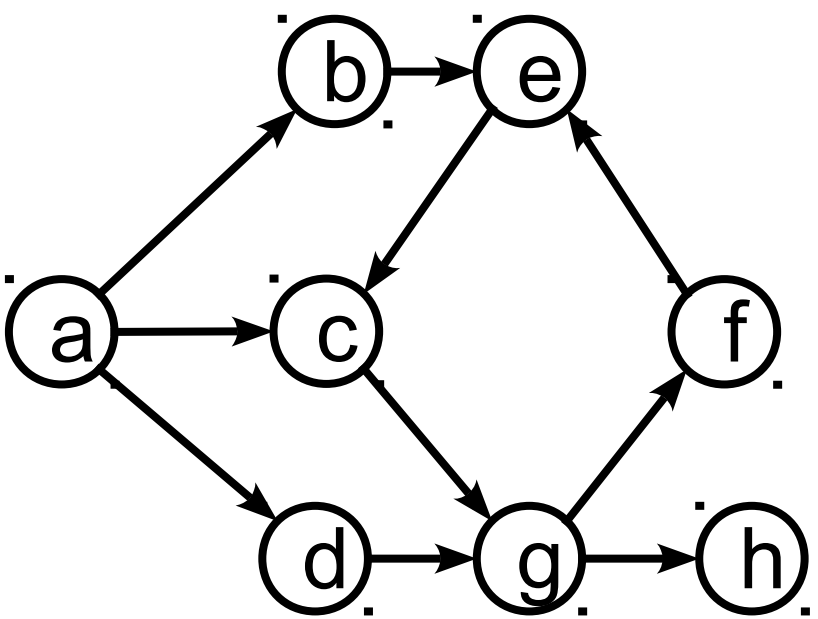


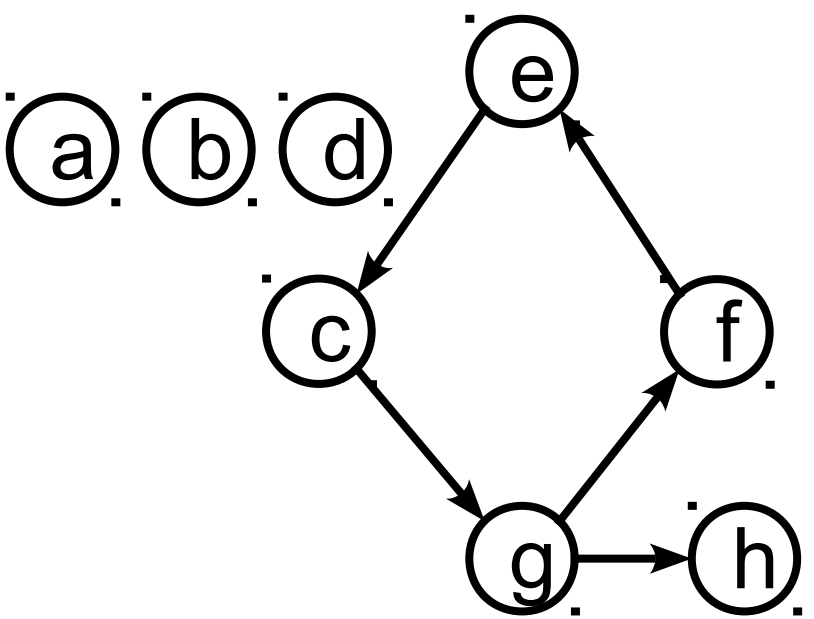
Příklad grafu, který zobrazuje návaznost činností na softwarovém projektu.

Topologické uspořádání

- Algoritmus
  - Napřed vybereme libovolný vrchol, do kterého nevede žádná hrana (tj. činnost, které nic nepředchází).
  - Tento vrchol přesuneme z grafu do posloupnosti
    - a z grafu také odstraníme všechny hrany, které z vrcholu vedly.
  - Pak vybereme další vrchol, do kterého nevede hrana
    - přesuneme jej do posloupnosti atd.
  - Výpočet skončí
    - Úspěšně: všechny vrcholy jsou přesunuty do posloupnosti.
    - Neúspěšně: v grafu zbudou vrcholy, do kterých vede hrana.
      - Odkud tam vede? – Nemůže vést odjinud než právě z některého z těchto vrcholů. Hrany mezi nimi totiž tvoří **cyklus.**

Topologické uspořádání





Grafy, které obsahují cyklus, nelze topologicky uspořádat.

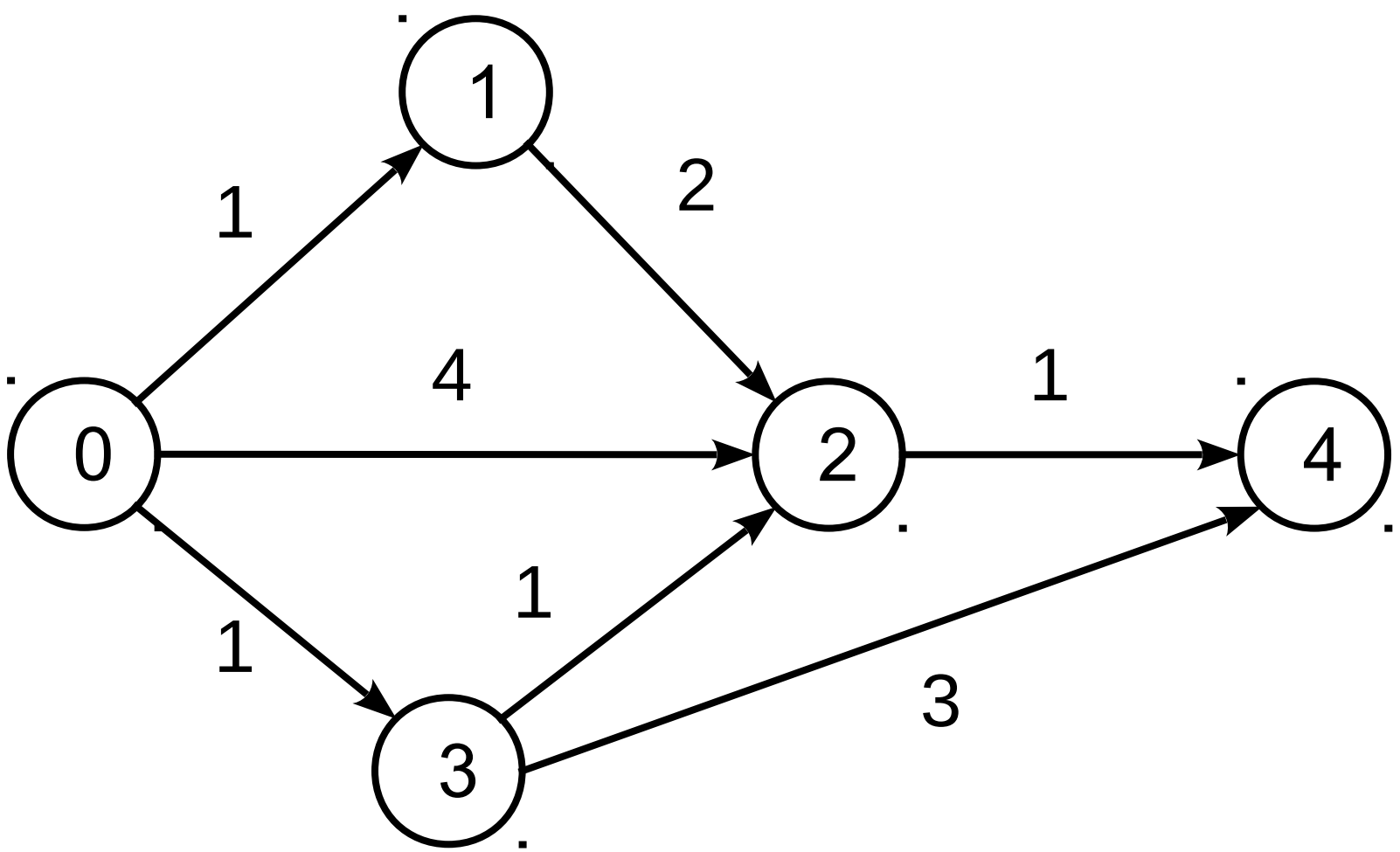
Topologické uspořádání

- Časová složitost
  - Do výsledné posloupnosti postupně vkládáme vrchol za vrcholem, celkem *V* vrcholů.
    - Při vložení každého vrcholu do posloupnosti ještě musíme projít všechny jeho následovníky a přepočítat jim počty předchůdců, příp. je zařadit mezi vrcholy bez předchůdců.
    - Následovníků každého vrcholu může být nejvýše *V*.
  - Pak časovou složitost můžeme odhadnout jako **kvadratickou** vzhledem k počtu vrcholů.
  - Uvažujeme-li také hrany:
    - Každá hrana grafu se prochází jen jednou.
    - Odhad můžeme zpřesnit na úměrný (*V* + *E*).

Agenda

- Práce s grafy
  - Další pojmy z teorie grafů
  - Reprezentace grafu
  - Úlohy na grafech
  - Komponenty souvislosti
  - Minimální kostra
  - Topologické uspořádání
  - **Hledání nejkratší cesty**
    - Floydův algoritmus
    - Dijkstrův algoritmus
  - Shrnutí

Hledání nejkratší cesty

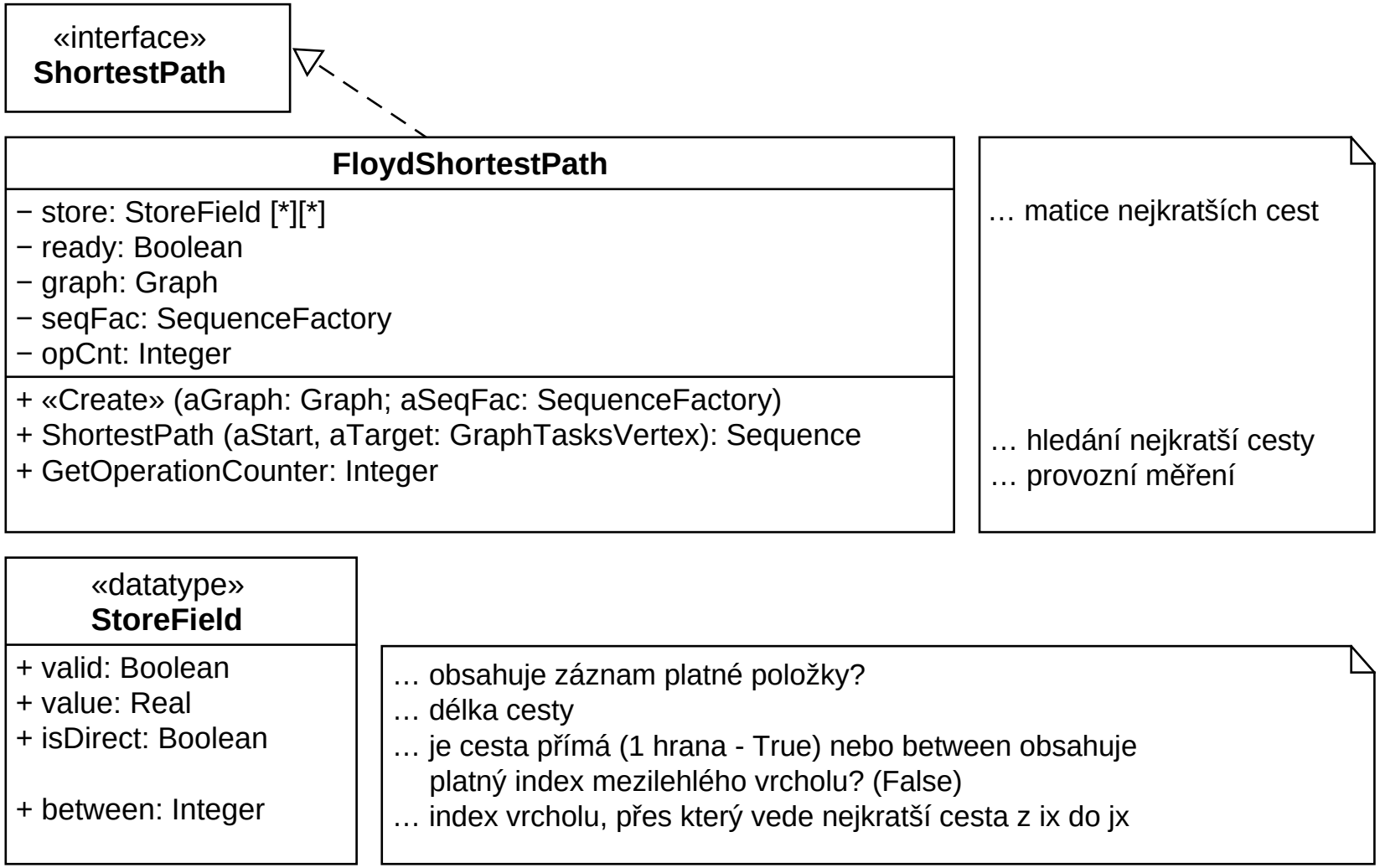


Příklad grafu, kde hledáme nejkratší cesty

Agenda

- Práce s grafy
  - Další pojmy z teorie grafů
  - Reprezentace grafu
  - Úlohy na grafech
  - Komponenty souvislosti
  - Minimální kostra
  - Topologické uspořádání
  - **Hledání nejkratší cesty**
    - **Floydův algoritmus**
    - Dijkstrův algoritmus
  - Shrnutí

Floydův algoritmus



Třída `floydShortestPath`

Floydův algoritmus

	0	1	2	3	4
0	0	1	4	1	–
1	–	0	2	–	–
2	–	–	0	–	1
3	–	–	1	0	3
4	–	–	–	–	0

na počátku

kx = 0: beze změny

	0	1	2	3	4
0	0	1	3	1	–
1	–	0	2	–	–
2	–	–	0	–	1
3	–	–	1	0	3
4	–	–	–	–	0

kx = 1

	0	1	2	3	4
0	0	1	3	1	4
1	–	0	2	–	3
2	–	–	0	–	1
3	–	–	1	0	2
4	–	–	–	–	0

kx = 2

	0	1	2	3	4
0	0	1	2	1	3
1	–	0	2	–	3
2	–	–	0	–	1
3	–	–	1	0	2
4	–	–	–	–	0

kx = 3

kx = 4: beze změny

Příklad výpočtu nejkratší cesty Floydovým algoritmem

Floydův algoritmus

- Algoritmus
  - Algoritmus pracuje s dvojrozměrným polem `store`, ve kterém si pro každou dvojici vrcholů pamatuje délku cesty. Jak výpočet postupuje, nachází čím dál tím kratší cesty. Na konci výpočtu pole `store` obsahuje délky nejkratších cest.
  - Na počátku označí všechny údaje o cestách v poli `store` jako neplatné (`valid = False`).
  - Pak projde všechny hrany a jejich hodnoty („délky“) nastaví do pole `store`: každá hrana spojuje dva vrcholy, čísla vrcholů slouží jako indexy do pole `store`.
    - Do položky `value` uloží hodnotu hrany.
    - Nastaví `valid = True` (v položce `value` uložena platná hodnota).
    - Nastraví `isDirect = True` (hrana propojuje vrcholy přímo).
  - Tímto způsobem vytvoříme v poli `store` *matici sousednosti.*
  - Potom se zkouší, jestli by se daly propojit vrcholy ix a jx nepřímo – cestou složenou ze dvou hran:
    - z vrcholu s číslem `ix` do vrcholu s číslem 0 a z vrcholu s číslem 0 do vrcholu s číslem `jx`.
      - Pokud se podaří propojit dosud nepropojené vrcholy nebo aspoň zkrátit dosud nejkratší známou cestu, zapamatuje se celková „délka“ cesty v příslušném prvku pole `store [ix, jx]`. Tím jsme do pole `store` dostali údaje o nejkratších cestách složených z jedné nebo dvou hran, kde mezilehlý vrchol má číslo 0.
    - Dál se postupně vybírají další vrcholy s číslem 1, 2 atd. (číslo vrcholu se uloží do proměnné kx) a zkouší se, zda by se dala najít nějaká ještě kratší cesta
      - složená z 2–4 hran s mezilehlým vrcholem 1
      - potom z 2–8 hran s mezilehlým vrcholem 2
      - potom z 2–16 hran s mezilehlým vrcholem 3 atd.
    - Délka každé takto nalezené cesty se uchovává v příslušném prvku pole `store` a použije se při dalším průchodu cyklem k hledání ještě kratších cest s využitím dalšího mezilehlého uzlu `kx`.
  - Po postupném vyzkoušení všech možných vrcholů `kx` bude pole `store` obsahovat nejkratší cesty.

Floydův algoritmus

- Poznámky
  - Dynamické programování
    - Obtížně řešitelná úloha (najdi všechny nejkratší cesty) se rozdělí na posloupnost začínající triviální úlohou (najdi přímé cesty), pokračuje úlohami postupně těžšími a těžšími (najdi nejkratší cesty obsahující mezilehlé uzly 0 až `kx`), až nakonec dospěje k řešení původně požadované úlohy. Přitom krok od snazší k těžší úloze je usnadněn tím, že v poli `store` máme uložené všechny dosavadní výsledky – jednou spočítané cesty se už nikdy nepočítají znovu.
    - Takováto strategie založená na rozdělení obtížné úlohy na snadné iterativní kroky a na opakovaném využití jednou spočítaných dílčích výsledků, se nazývá **dynamické programování.**
  - POZOR na záporně ohodnocené hrany v cyklech
    - Kdyby hodnota celého cyklu byla záporná, potom každým průchodem takového cyklu by se cesta zkrátila. Čím víckrát bychom záporným cyklem prošli, tím kratší cestu bychom našli – v tom případě celá úloha o hledání nejkratší cesty zřejmě ztrácí smysl.

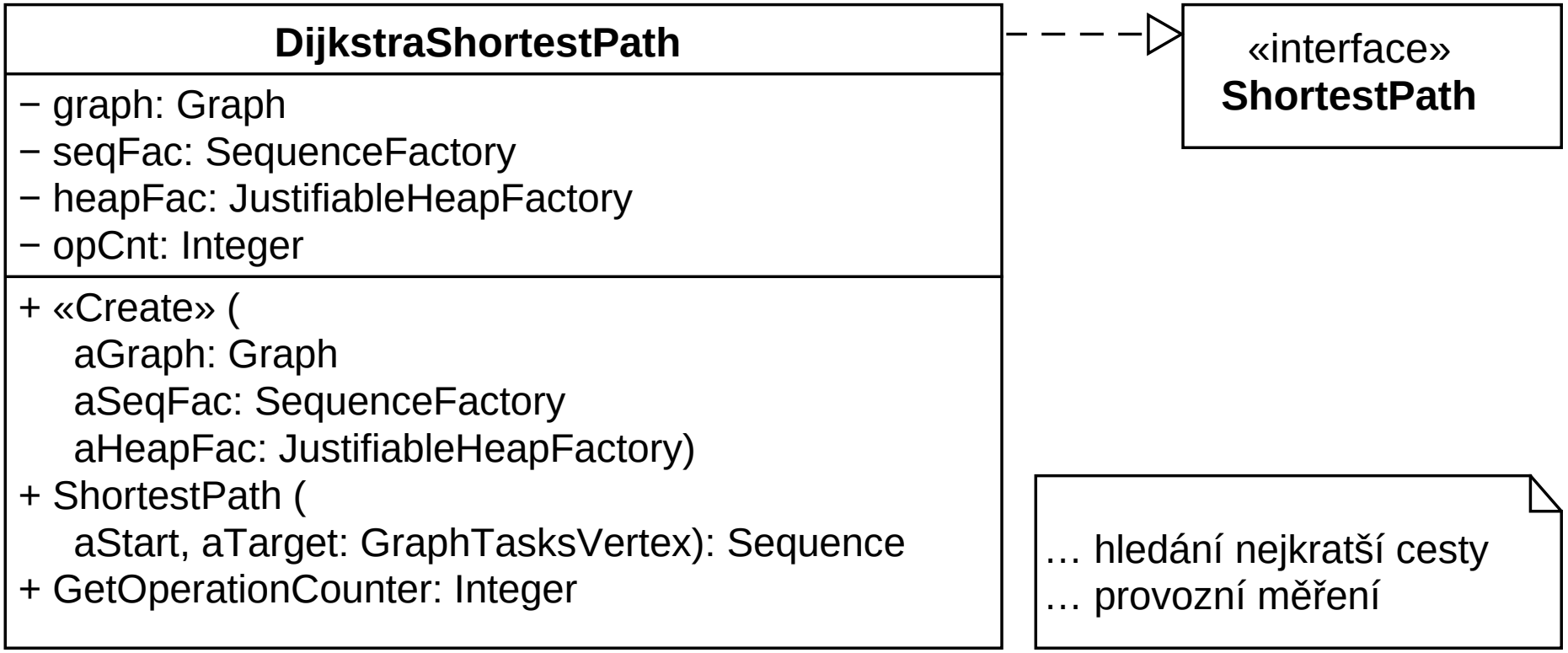
Floydův algoritmus

- Časová a paměťová složitost
  - V programu zřetelně vidíme tři vnořené cykly. Proto má Floydův algoritmus **kubickou** časovou složitost
    - tj. všechny nejkratší cesty mezi dvojicemi V vrcholů najde v V^3 krocích výpočtu.
  - Tím, že si musí pamatovat matici nejkratších cest, je náročný i paměťově, a to **kvadraticky:**
    - Matice má V^2 položek.

Agenda

- Práce s grafy
  - Další pojmy z teorie grafů
  - Reprezentace grafu
  - Úlohy na grafech
  - Komponenty souvislosti
  - Minimální kostra
  - Topologické uspořádání
  - **Hledání nejkratší cesty**
    - Floydův algoritmus
    - **Dijkstrův algoritmus**
  - Shrnutí

Dijkstrův algoritmus



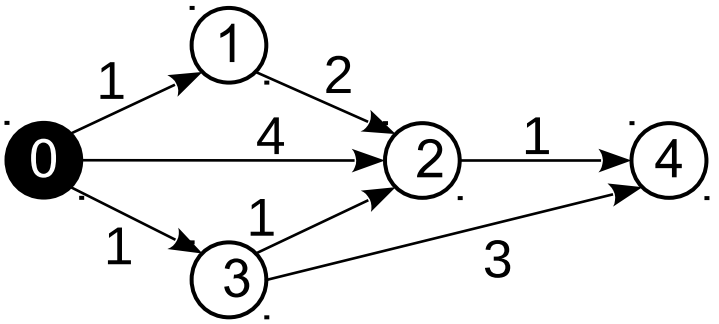
Třída `DijkstraShortestPath`

Dijkstrův algoritmus

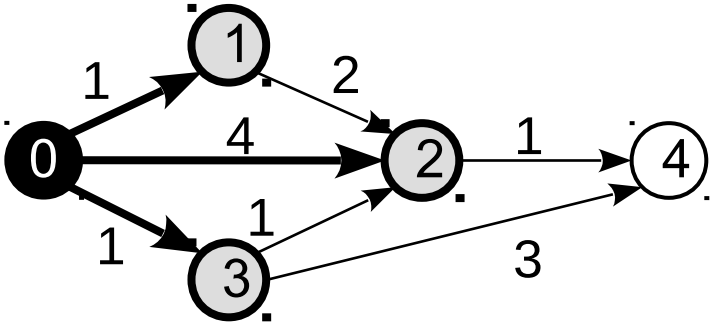
- Algoritmus (je založen na principu prohledávání do šířky, „vlnou“).
  - Napřed nastaví všechny vrcholy do stavu `Unreached` (nedosažené).
  - Výchozí vrchol nastaví jako běžný, vzdálenost od sebe samého je `0`, stav `Completed` (dokončený).
  - Potom cyklem prohledává graf, až buď dosáhne cílového vrcholu, anebo vyčerpá všechny dosažitelné vrcholy:
    - Nejprve zkusí, zda poslední dosažený dokončený vrchol je hledaný cílový vrchol.
      - Je-li tomu tak, ukončí prohledávání.
      - Nebylo-li dosaženo cílového vrcholu, nastaví všem následovníkům běžného uzlu nejkratší dosud známou vzdálenost od uzlu výchozího.
        - Nebyl-li následovník dosud navštíven (je ve stavu `Unreached`), vzdálenost se spočítá prostě jako součet délky hrany a vzdálenosti běžného uzlu od uzlu výchozího. Stav následovníka nastaví na navštívený (`Visited`).
        - Byl-li následovník už navštíven a jeho dosud známá vzdálenost je delší než vzdálenost právě spočtená, je zřejmě cesta přes běžný vrchol kratší.
          - Algoritmus uloží její délku do `pathLength`.
          - Vrchol už byl vložen do pomocné haldy, halda se teď musí přerovnat metodou Justify.
        - Vrchol nově navštívený se musí přidat do haldy.
    - Po prozkoumání a případné opravě všech následovníků algoritmus přejde z dosud běžného vrcholu do jednoho z vrcholů navštívených, ale dosud nedokončených.
      - Ze všech navštívených vrcholů se pokusí vybrat ten, do kterého vede nejkratší cesta z výchozího vrcholu. K tomu slouží halda.
        - Je-li halda prázdná, algoritmus již prozkoumal všechny dostupné vrcholy v grafu, ale cílový vrchol nenašel. V grafu neexistuje cesta z výchozího do cílového vrcholu. Algoritmus skončí a jako výsledek vrátí prázdnou cestu, která neobsahuje žádný vrchol.
        - Není-li halda prázdná, algoritmus nastaví vrchol vybraný z haldy jako běžný a jeho stav na dokončený (`Completed`).
    - Cyklus se opakuje pro další a další běžné vrcholy.
    - Po ukončení cyklu algoritmus naplní výslednou posloupnost vrcholy, které leží na nejkratší cestě.

Dijkstrův algoritmus

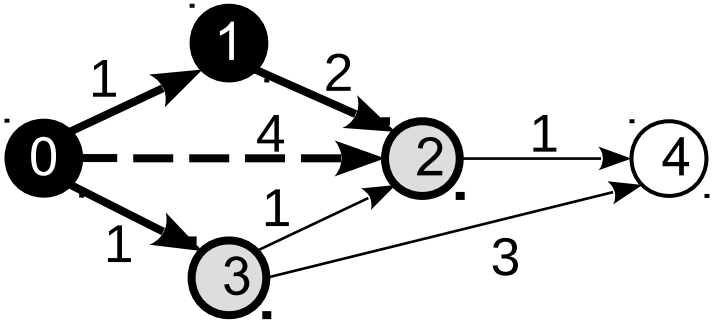
(a)



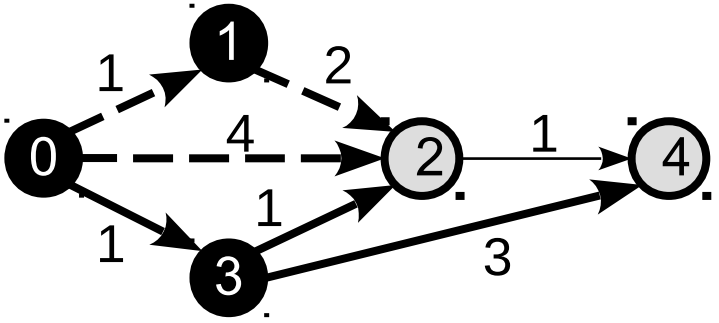
(b)



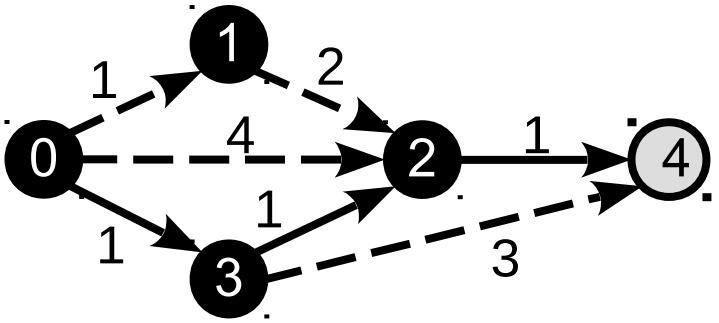
(c)



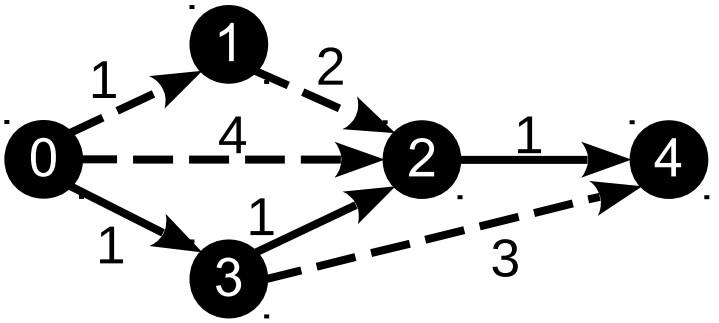
(d)



(e)



(f)



Ukázka Dijkstrova algoritmu

Dijkstrův algoritmus

- Časová složitost
  - V krajně nepříznivém případě musíme navštívit všechny vrcholy grafu, některé i víckrát.
    - Každý vrchol grafu, který leží na nejkratší cestě, musíme dokončit. V krajním případě musíme dokončit všech V vrcholů.
    - Dokončený vrchol vybíráme z prioritní fronty – haldy. Na to potřebujeme $\log_2 V$ kroků.
    - Z každého z nejvýše V dokončených vrcholů pak musíme prohlédnout i všechny jeho následovníky, kterých může být také až V.
  - Odhad časové složitosti vzhledem k počtu vrcholů je $V^2 \log V$.
  - Vezmeme-li v úvahu počet hran E, můžeme využít i toho, že každou hranou projdeme nejvýše jednou. Pak vyjde časová složitost nanejvýš přímo úměrná $(V + E) \log_2 V$.

Agenda

- Práce s grafy
  - Další pojmy z teorie grafů
  - Reprezentace grafu
  - Úlohy na grafech
  - Komponenty souvislosti
  - Minimální kostra
  - Topologické uspořádání
  - Hledání nejkratší cesty
    - Floydův algoritmus
    - Dijkstrův algoritmus
  - **Shrnutí**

Shrnutí

- **Vnitřní reprezentaci grafu** volíme na míru algoritmu, kterým hodláme řešit úlohu. Např.:
  - *k Floydovu algoritmu* se hodí **matice sousednosti**
  - *k Dijkstrovu algoritmu* **seznam následníků**
  - k uložení grafu do souboru zase spíš **posloupnost hran**
- Je třeba rozlišovat mezi grafy **orientovanými** a **neorientovanými**. Např.:
  - *Topologicky uspořádáváme* graf orientovaný
  - *minimální kostru* hledáme pro graf neorientovaný
  - *nejkratší cestu* můžeme hledat jak na grafech orientovaných, tak na neorientovaných.
- Úlohy, které se řeší na grafech:
  - zjišťování různých **vlastností grafu** – např.: Je graf strom? Jsou v grafu cykly? Existuje cesta mezi zadanými vrcholy? Je graf souvislý?
  - **optimalizační úlohy,** např. hledání nejkratší cesty, dopravní problém. Optimalizace bývají časově nebo paměťově náročné, ale
    - **hladové (greedy)** algoritmy jsou mimořádně efektivní
      - Např. *Kruskalův algoritmus* na hledání minimální kostry grafu.

Konec

Tato prezentace patří k učebnici *Algoritmy a datové struktury objektově* od Ivana Ryanta. Obsahuje texty a obrázky z této učebnice.

RYANT, Ivan. *Algoritmy a datové struktury objektově.* Praha: Ivan Ryant, 2017. ISBN 978-80-270-1660-0