PageRank

Ondřej Drbohlav, drbohlav@fel.cvut.cz

24. listopadu 2009

1997

Major Events of 1997

- Britain cedes Hong Kong (a British colony since 1841) back to China.
- Princess Diana dies in road accident in Paris, France.
- NASA's Mars Pathfinder probe lands on Mars.
- An unmanned Progress cargo-ship crashes into the Mir space station.
- Tiger Woods becomes the youngest golfer to win the Masters Tournament.
- Franklin Delano Roosevelt Memorial dedicated in Washington D.C.

(Podle `http://www.fun4birthdays.com`, odkaz č. 1 z prohledávače google.com) na dotaz 'Major events 1997'

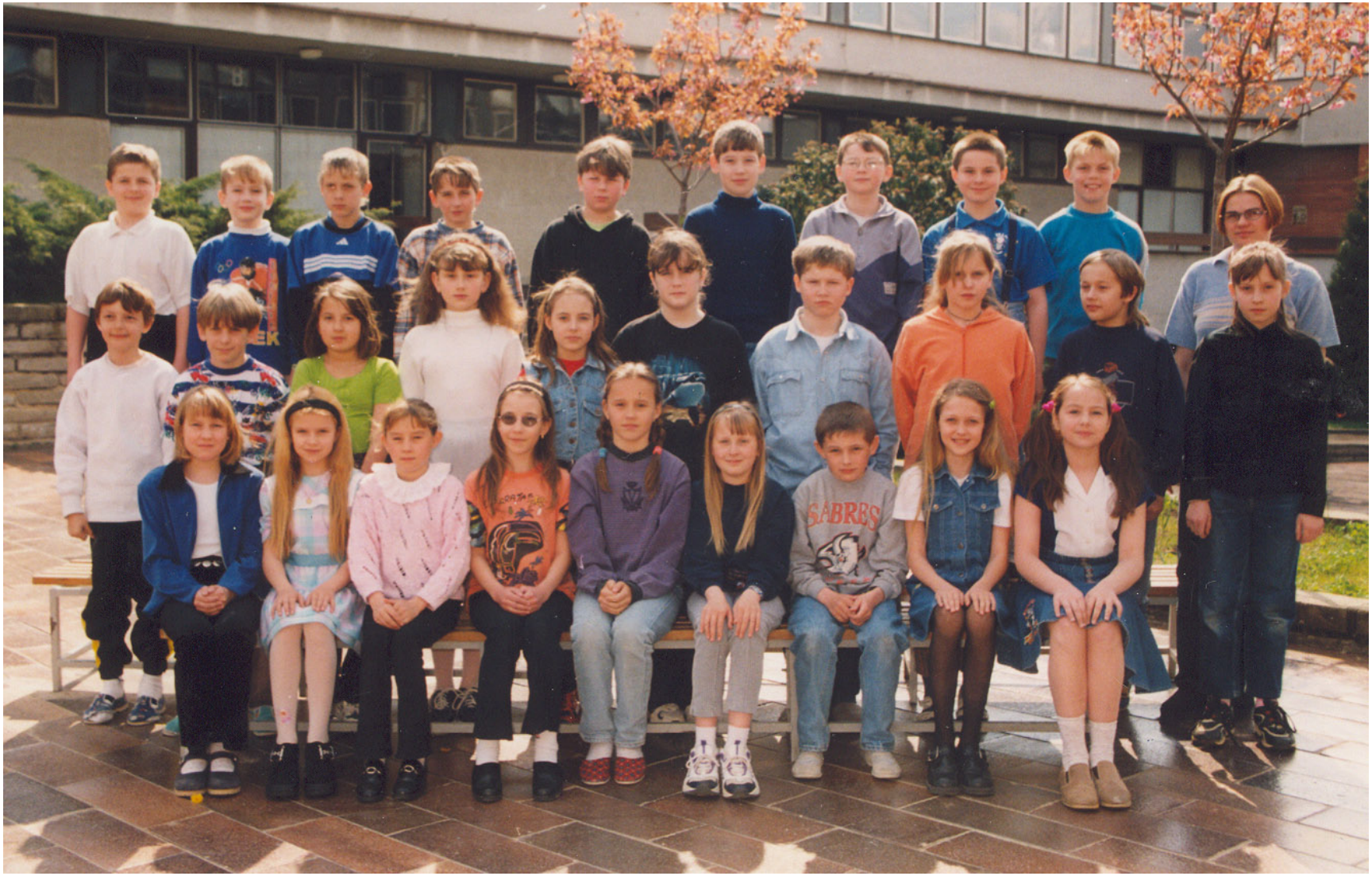
1997 – technika

Typické PC:

- 14-ti palcový monitor

- Procesor: Pentium, 1 jádro, 233MHz
- 32MB RAM
- 4.3GB harddisk
- floppy disk

1997 – budoucí studenti DZO



1997 – Internet (skoro tak, jak ho známe)

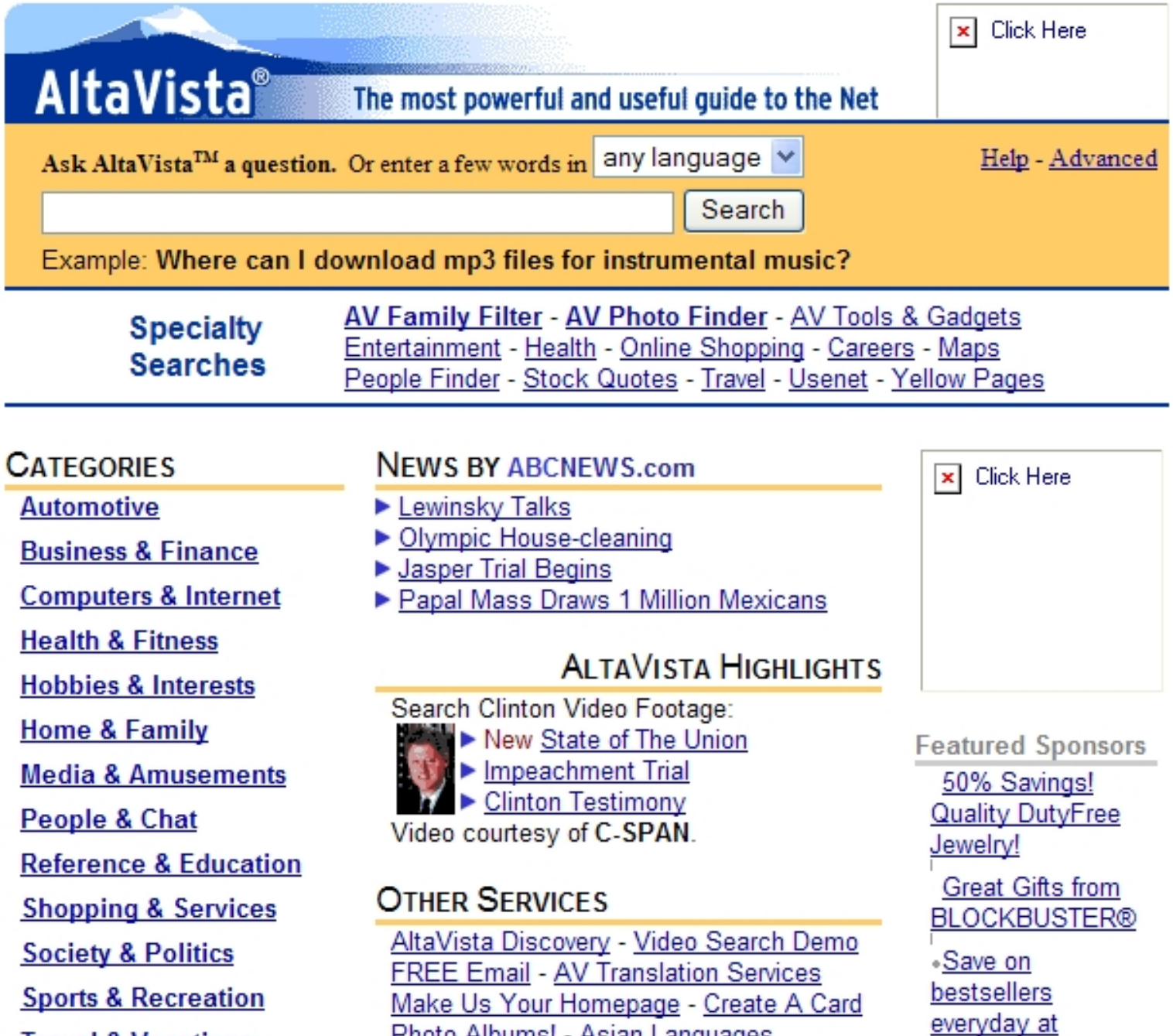
- Věk:

- první WWW prohlížeč Mosaic (1993)
- Netscape (1994)

Vyhledávací stroje:

- Altavista

Příklad: Altavista



Příklad: Hledání

Hledej: Apple

1. . . . byla jsem za Petrou a ta mě požádala o recept na skvělý babiččin koláč. Tady je: jedno **jablko**, dvě vejce, . . .
2. . . .

⋮

105. **Apple**. Chystáme se na výrobu iPodů. Až vyrostete, kupujte to!

⋮

217. **Jablko** patří mezi ovoce. Příbuzným druhem je hruška . . .

⋮

1997 - vlastnosti hledání

- velice často *nerelevantní* výsledky
- nic se nedá najít a nebo to zabere spoustu času
- v dokumentu se slova sice vyskytují, ale dokument sám je na poměrně obskurních stránkách

Příčiny?

- za všemi vyhledávacími stroji jsou stovky lidí, kteří procházejí rodící se internet. Odhadují důležitost stránek
- problém je v tom, že Internet rychle přerostl možnosti ručního procházení: jednak počtem stránek, jednak rychlostí obměny obsahu.

Co musí vyhledávací stroj umět?

- pracovat plně automaticky
- procházet Internet
- indexovat stránky (uchovávat jejich obsah ve formě vhodné k rychlému vyhledání při zadání klíčového slova)
- hodnotit důležitost stránek

Co musí vyhledávací stroj umět?

- pracovat plně automaticky
- procházet Internet
- indexovat stránky (uchovávat jejich obsah ve formě vhodné k rychlému vyhledání při zadání klíčového slova)
- hodnotit důležitost stránek

Jak procházet web a nezacyklit se?

Co musí vyhledávací stroj umět?

- pracovat plně automaticky
- procházet Internet
- indexovat stránky (uchovávat jejich obsah ve formě vhodné k rychlému vyhledání při zadání klíčového slova)
- hodnotit důležitost stránek

Jaké struktury dat zvolit pro indexování? Jak provádět efektivní vyhledávání dokumentů s výskytem klíčového slova?

Co musí vyhledávací stroj umět?

- pracovat plně automaticky
- procházet Internet
- indexovat stránky (uchovávat jejich obsah ve formě vhodné k rychlému vyhledání při zadání klíčového slova)
- hodnotit důležitost stránek

- Je to přesně to, co vyhledávače v roce 1997 neuměly.
- Přímo souvisí s pozorovaným problémem s nerelevantností výsledků hledání.
- Očekávané zlepšení výsledků vyhledávání je velké. Většina textu na internetu se obsahuje cca 10.000 slov; dnešní velikost internetu je cca 30 miliard stránek $\Rightarrow$ seřazení stránek podle důležitosti je opravdu potřebné!

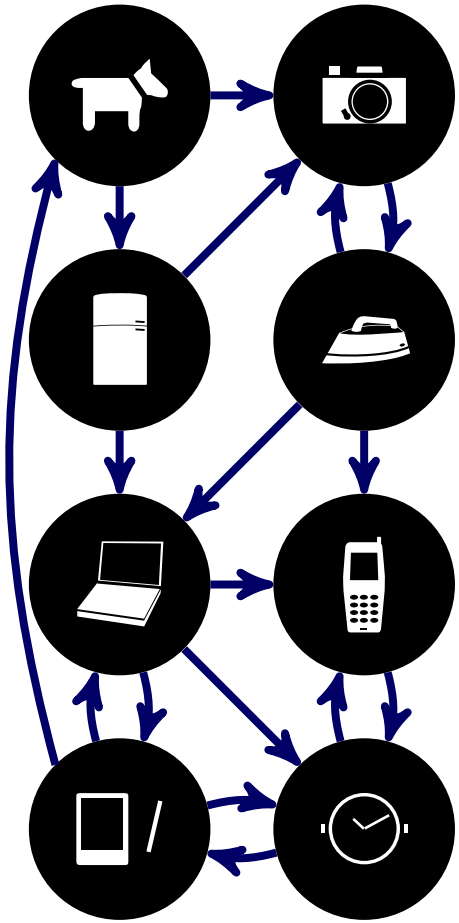
Jak automaticky zjistit důležitost stránek?

- velikost stránky?
- použití spisovného jazyka?
- kritéria založená na rozpoznávání smyslu textu?
- ...

Možné problémy:

- částečně by to mohlo fungovat, ale
- jednoduchá kritéria je "snadné" splnit: *Velikost*: dosáhnout je to podstatně jednodušší než např. ve světě firem
- *smysl textu*: extrémně těžké, neumí se to
- leccos je v zásadě snadné zfalšovat

Jak automaticky zjistit důležitost stránek?

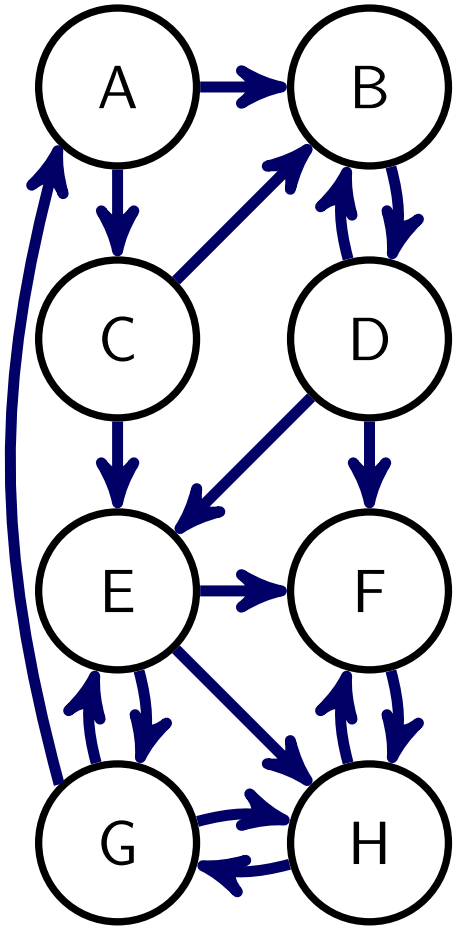


- web nejsou jen stránky, ale i odkazy mezi nimi
- mohl bych je použít pro spočítání důležitosti stránek?

Úvahy:

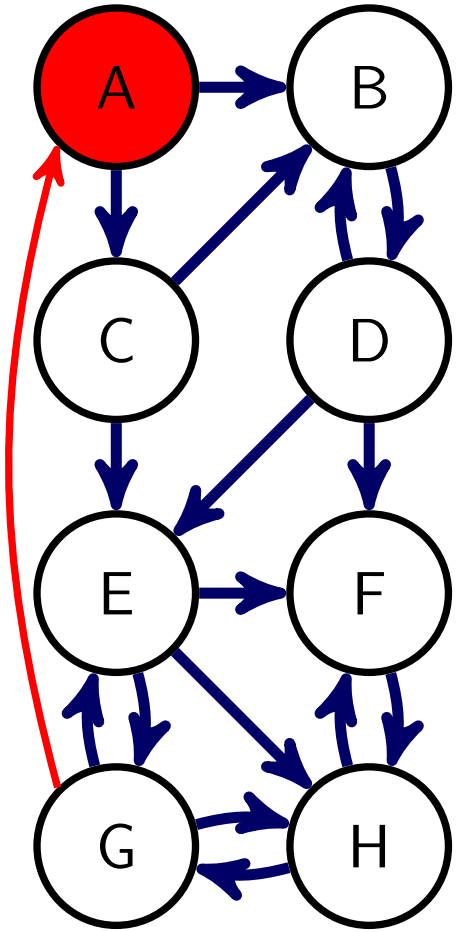
- pokud O doporučuje (odkazuje) mnoho lidí, O je věrohodná
- velká výhoda bude založit skóre stránky na odkazech **na** tuto stránku, tj. **zpětných** odkazech, protože je to něco, co vlastník stránky nemá v moci.

Nápad 1: počítání zpětných odkazů



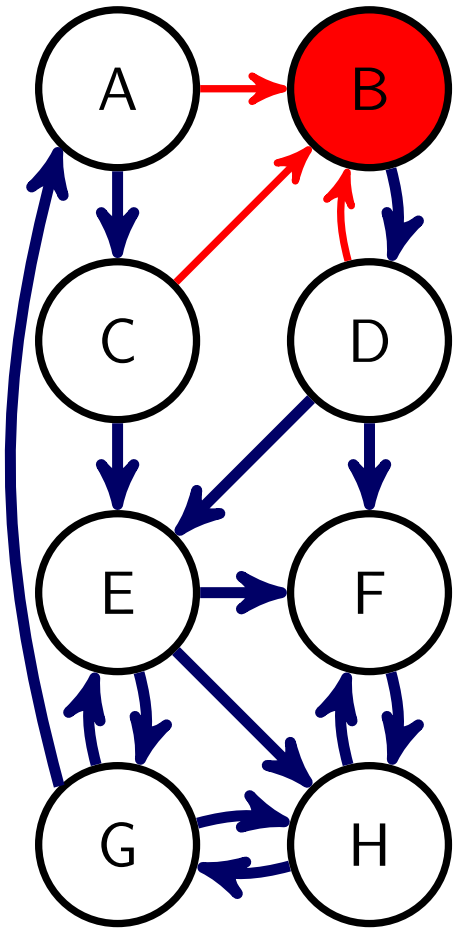
- začněme co nejjednoduššeji: co spočítat počet zpětných linků?

Nápad 1: počítání zpětných odkazů



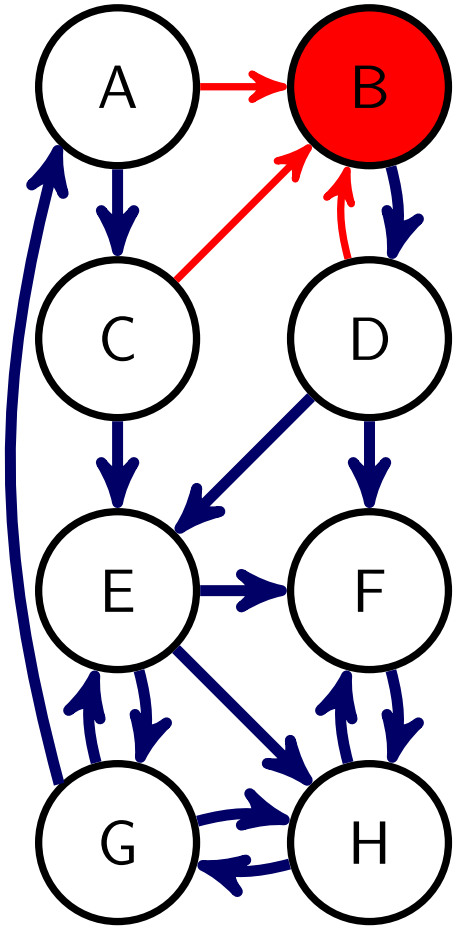
- začněme co nejjednoduššeji: co spočítat počet zpětných linků?
- $W_A = 1$

Nápad 1: počítání zpětných odkazů



- začněme co nejjednoduššeji: co spočítat počet zpětných linků?
- $W_A = 1$
- $W_B = 3$

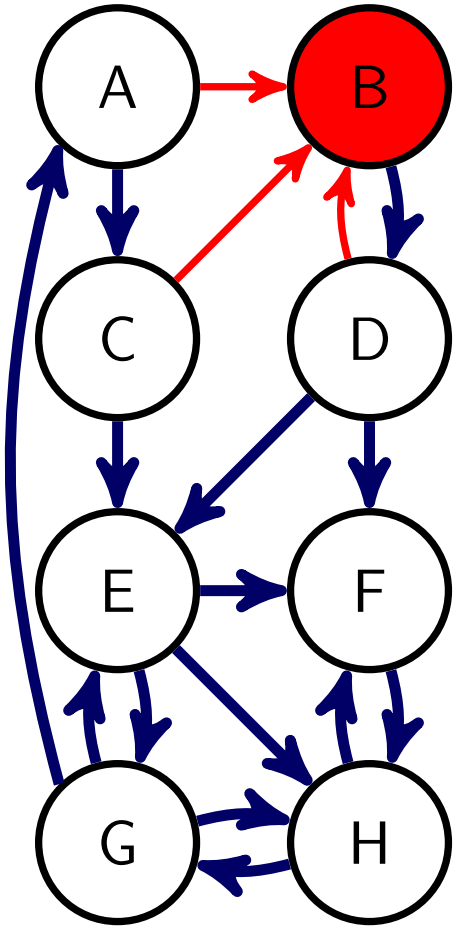
Nápad 1: počítání zpětných odkazů



- začněme co nejjednoduššeji: co spočítat počet zpětných linků?
- $W_A = 1$
- $W_B = 3$

  ...
- $W = [1, 3, 1, 1, 3, 3, 2, 3]$

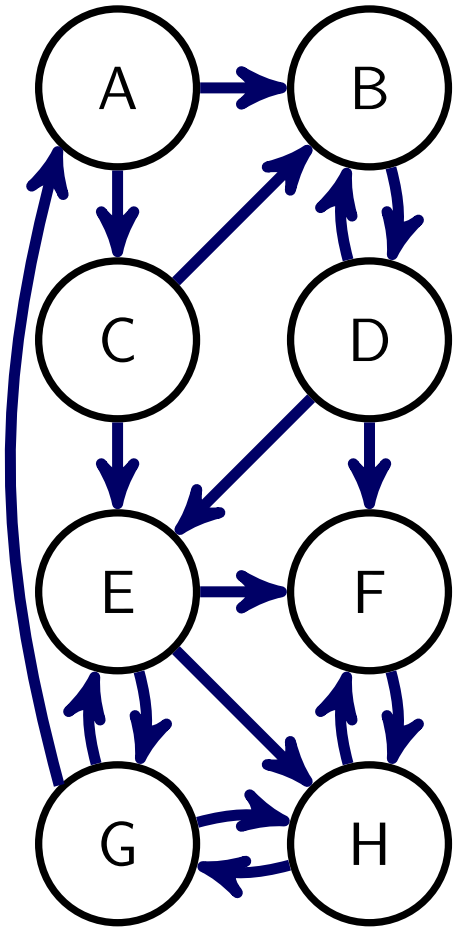
Nápad 1: počítání zpětných odkazů



- začněme co nejjednoduššeji: co spočítat počet zpětných linků?
- $W_A = 1$
- $W_B = 3$

  ...
- $W = [1, 3, 1, 1, 3, 3, 2, 3]$
- jako první přiblížení je to fajn, ale je samotný počet zpětných odkazů jednak nepopisuje v podstatě vůbec strukturu sítě, jednak je snadno manipulovatelný (návod: pan Zlobivý si koupí 10 domén a z každé z nich odkáže svou hlavní doménu)

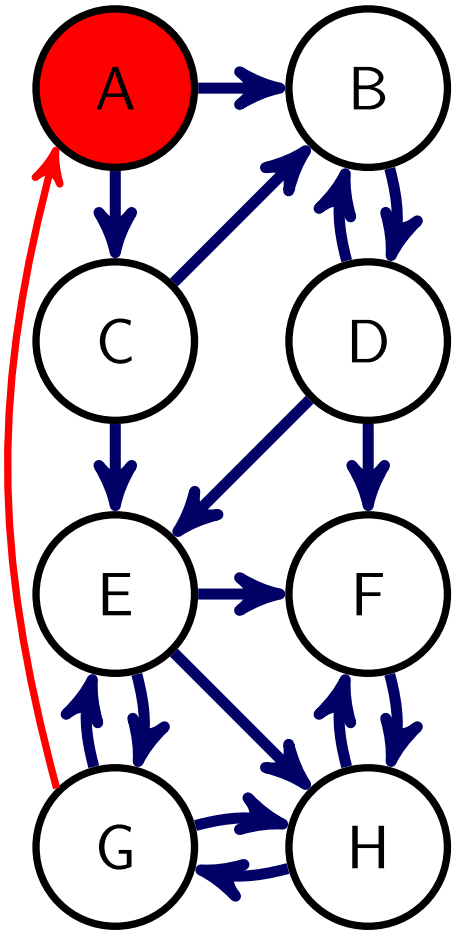
Můžeme počítání zpětných odkazů vylepšit?



Úvaha:

- Není jedno, kdo na mě dává odkaz. Odkaz od důležité stránky má větší váhu než víceméně náhodný odkaz.
- Zkusme tedy sčítat ne *počet* odkazů, ale jejich *váhy*:

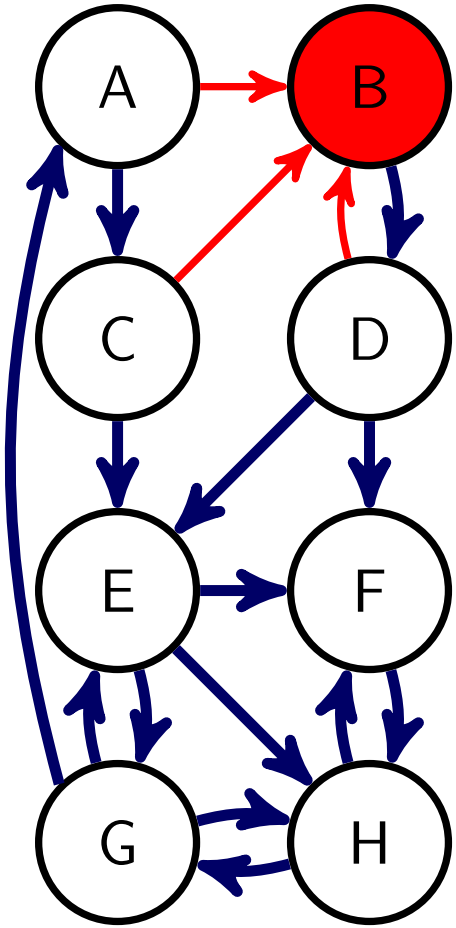
Můžeme počítání zpětných odkazů vylepšit?



Úvaha:

- Není jedno, kdo na mě dává odkaz. Odkaz od důležité stránky má větší váhu než víceméně náhodný odkaz.
- Zkusme tedy sčítat ne *počet* odkazů, ale jejich *váhy*:
- $W_A = W_G$

Můžeme počítání zpětných odkazů vylepšit?

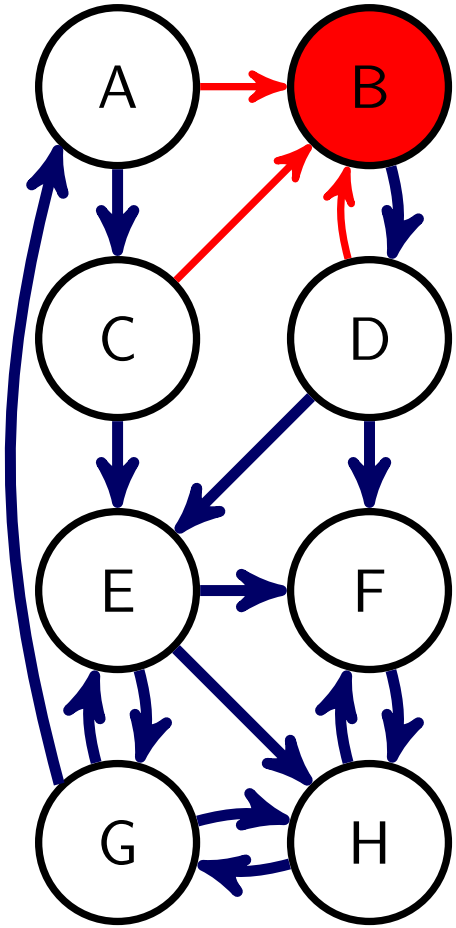


Úvaha:

- Není jedno, kdo na mě dává odkaz. Odkaz od důležité stránky má větší váhu než víceméně náhodný odkaz.
- Zkusme tedy sčítat ne *počet* odkazů, ale jejich *váhy*:
- $W_A = W_G$
- $W_B = W_A + W_C + W_D$

  ⋮

Můžeme počítání zpětných odkazů vylepšit?

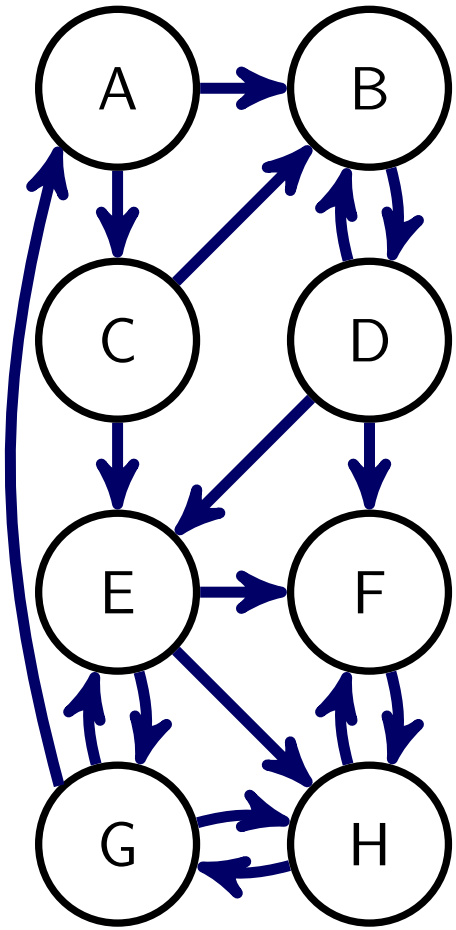


Úvaha:

- Není jedno, kdo na mě dává odkaz. Odkaz od důležité stránky má větší váhu než víceméně náhodný odkaz.
- Zkusme tedy sčítat ne *počet* odkazů, ale jejich *váhy*:
- $W_A = W_G$
- $W_B = W_A + W_C + W_D$

  ⋮
- To vypadá rozumně, navíc  má trochu ztíženou práci: musí myslet na to, že svým 'pomocným' doménám musí zajistit kredibilitu.

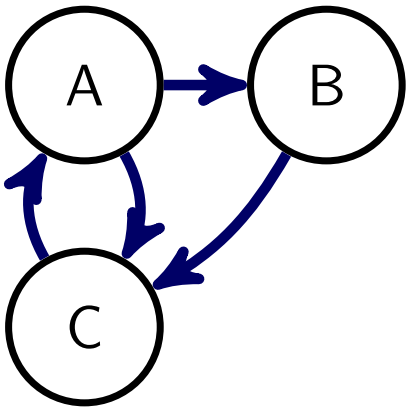
Můžeme počítání zpětných odkazů vylepšit?



Úvaha:

- máme ale problém, uvažme jednoduchou síť

Můžeme počítání zpětných odkazů vylepšit?



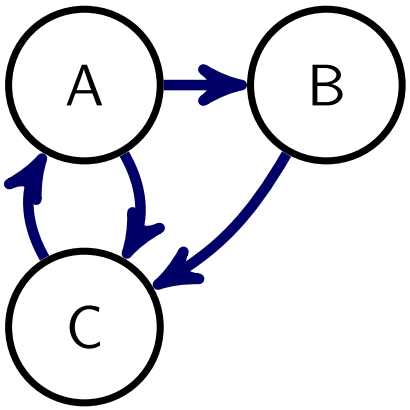
Úvaha:

- máme ale problém, uvažme jednoduchou síť

$$W_A = W_C$$
$$W_B = W_A$$
$$W_C = W_A + W_B$$

- to není možné splnit

Můžeme počítání zpětných odkazů vylepšit?



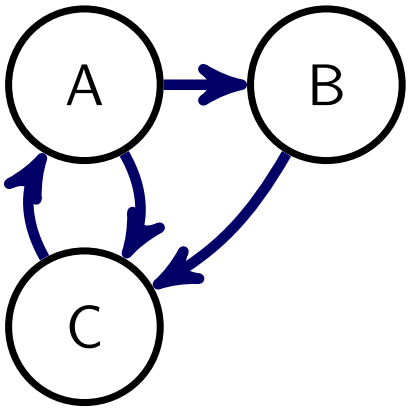
Úvaha:

- máme ale problém, uvažme jednoduchou síť

$$W_A = W_C$$
$$W_B = W_A$$
$$W_C = W_A + W_B$$

- to není možné splnit
- možná není správné, že stránky mohou hlasovat pro libovolný počet ostatních stránek
- co kdybychom tedy každému dali jen jeden hlas, který může spravedlivě dělit mezi všechny stránky, které doporučuje?

Můžeme počítání zpětných odkazů vylepšit?

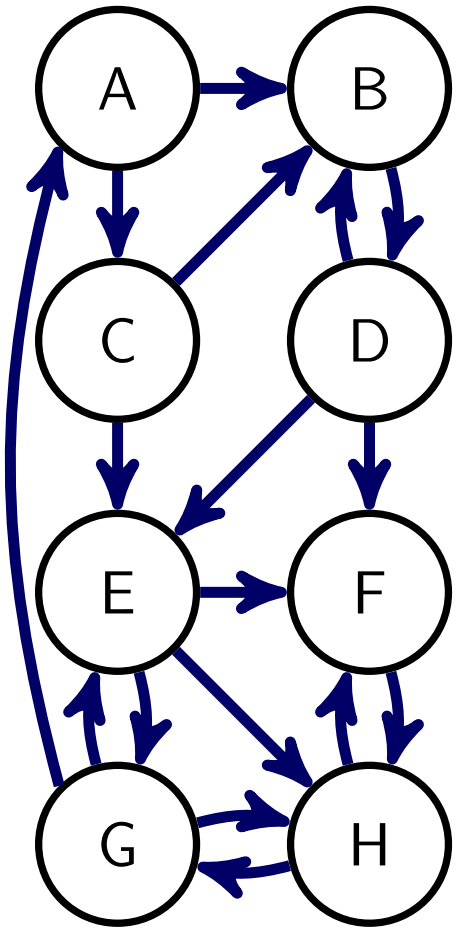


- označme n_A počet odkazů vedoucích z webu A
- $n_A = 2$, $n_B = 1$, $n_C = 1$
- s použitím poměrného hlasování tedy bude

$$
\begin{aligned}
W_A &= \frac{W_C}{n_C} = W_C \\
W_B &= \frac{W_A}{n_A} = \frac{1}{2} W_A \\
W_C &= \frac{W_A}{n_A} + \frac{W_B}{n_B} = \frac{1}{2} W_A + W_B
\end{aligned}
$$

- to je konzistetní!

Můžeme počítání zpětných odkazů vylepšit?

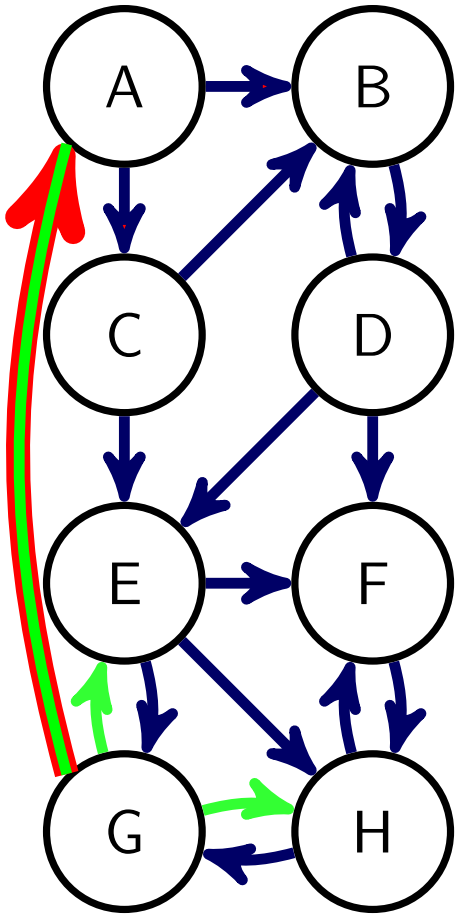


- Zkusme hodnotit důležitost webů takto:

$$W_k = \sum_{b \in \{\rightarrow k\}} \frac{W_b}{n_b},$$

kde $\{\rightarrow k\}$ je množina webů, které odkazují na web k.

Můžeme počítání zpětných odkazů vylepšit?



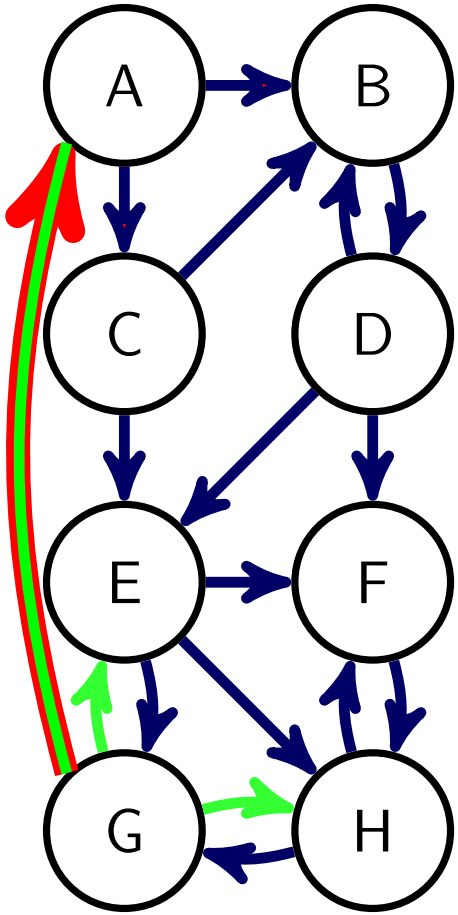
- Zkusme hodnotit důležitost webů takto:

$$W_k = \sum_{b \in \{\rightarrow k\}} \frac{W_b}{n_b},$$

kde $\{\rightarrow k\}$ je množina webů, které odkazují na web k.

- Příklad: $W_A = \frac{W_G}{3}$

Můžeme počítání zpětných odkazů vylepšit?

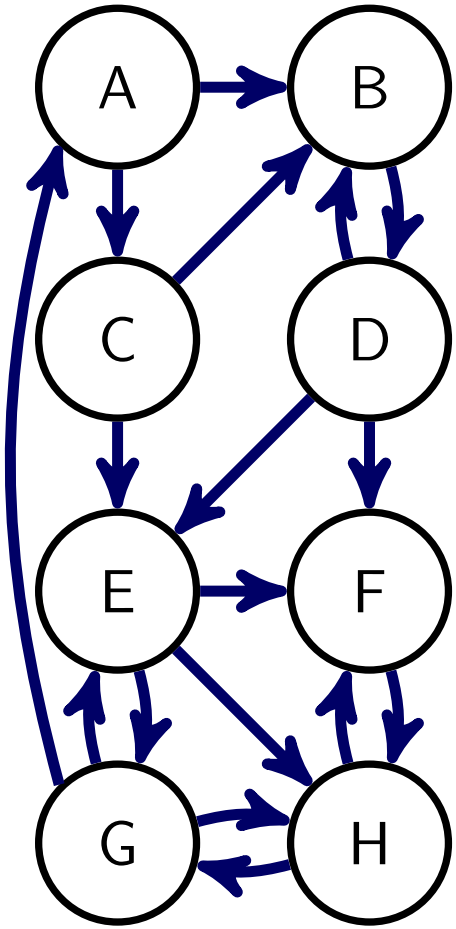


- Zkusme hodnotit důležitost webů takto:

$$W_k = \sum_{b \in \{\rightarrow k\}} \frac{W_b}{n_b},$$

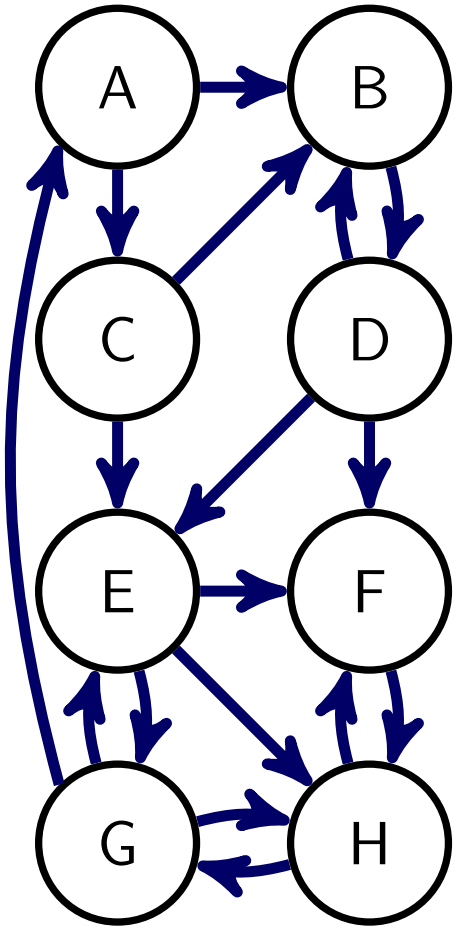
  kde $\{\rightarrow k\}$ je množina webů, které odkazují na web k.
- Příklad: $W_A = \frac{W_G}{3}$
- Jak takovou soustavu rovnic vyřešit?
- Jde to vůbec? (pro obecný web)?
- Je řešení jen jedno, nebo víc?

Nápad 2: Náhodná procházka

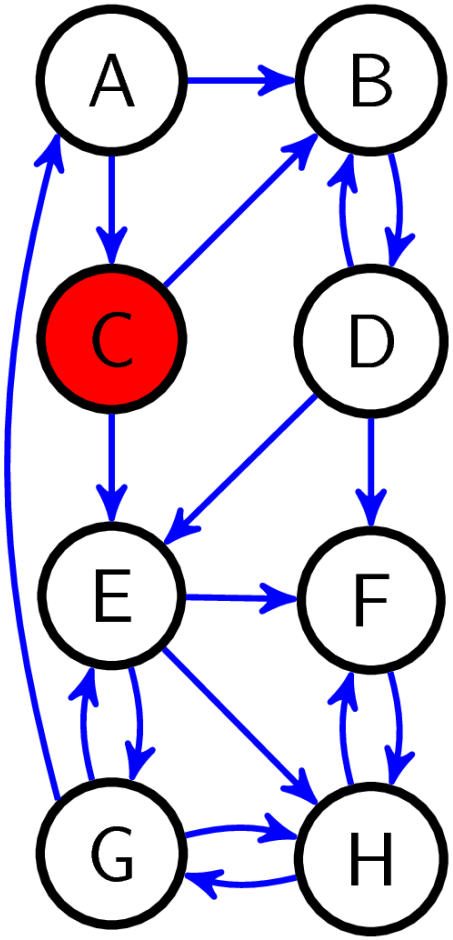


- co kdybychom nemuseli počítat skoro nic?
- máme strukturu webu k dispozici – jak bychom toho mohli využít
- nechme robota, aby procházel web následujícím způsobem:
  - začni na náhodné stránce
  - na této stránce je X odkazů. Náhodně se rozhodni pro jeden z nich
  - opakuj
  - počítej, kolikrát se dostaneš na jednotlivé stránky
- to celkem dává smysl, a vystihuje to konektivitu sítě

Nápad 2: Náhodná procházka

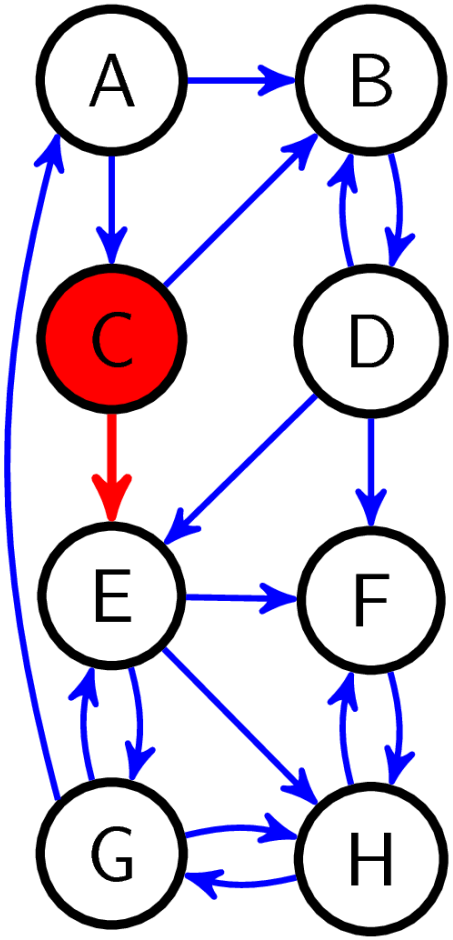


Nápad 2: Náhodná procházka



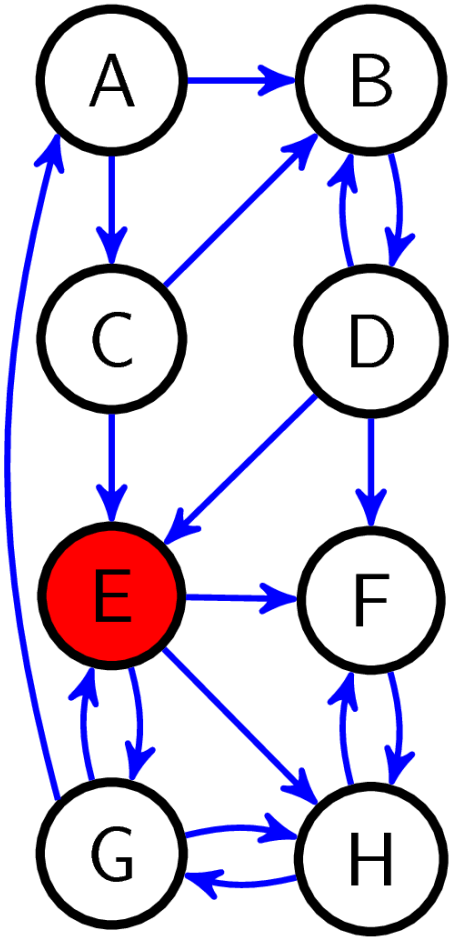
- začnu na C; 2 možná pokračování

Nápad 2: Náhodná procházka



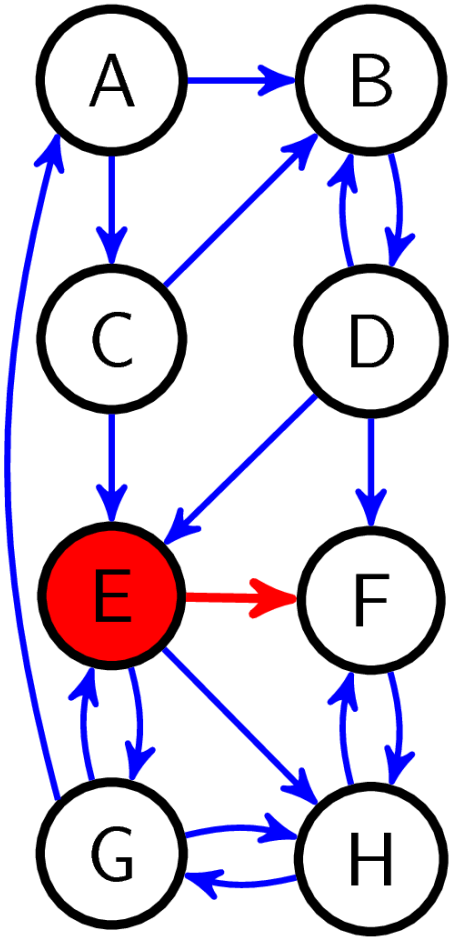
- začnu na C; 2 možná pokračování
- náhodně vyberu

Nápad 2: Náhodná procházka



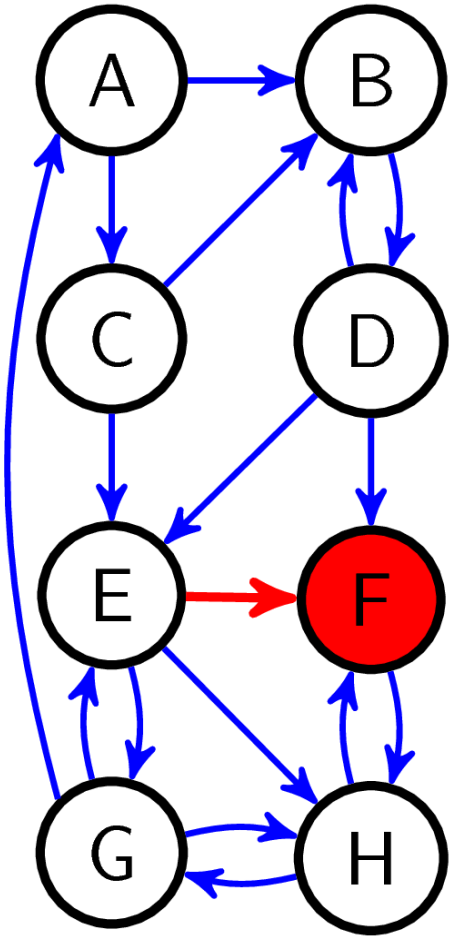
- začnu na C; 2 možná pokračování
- náhodně vyberu
- dostanu se do E; 3 cesty

Nápad 2: Náhodná procházka



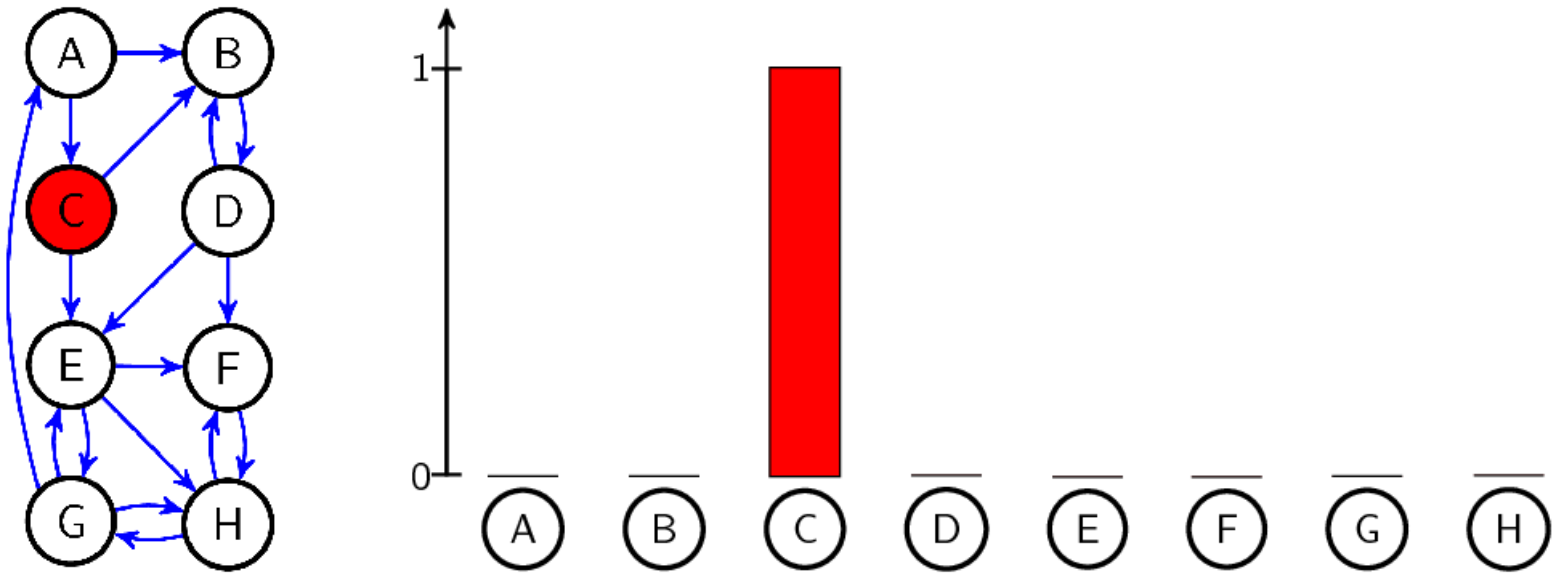
- začnu na C; 2 možná pokračování
- náhodně vyberu
- dostanu se do E; 3 cesty
- náhodně vyberu

Nápad 2: Náhodná procházka

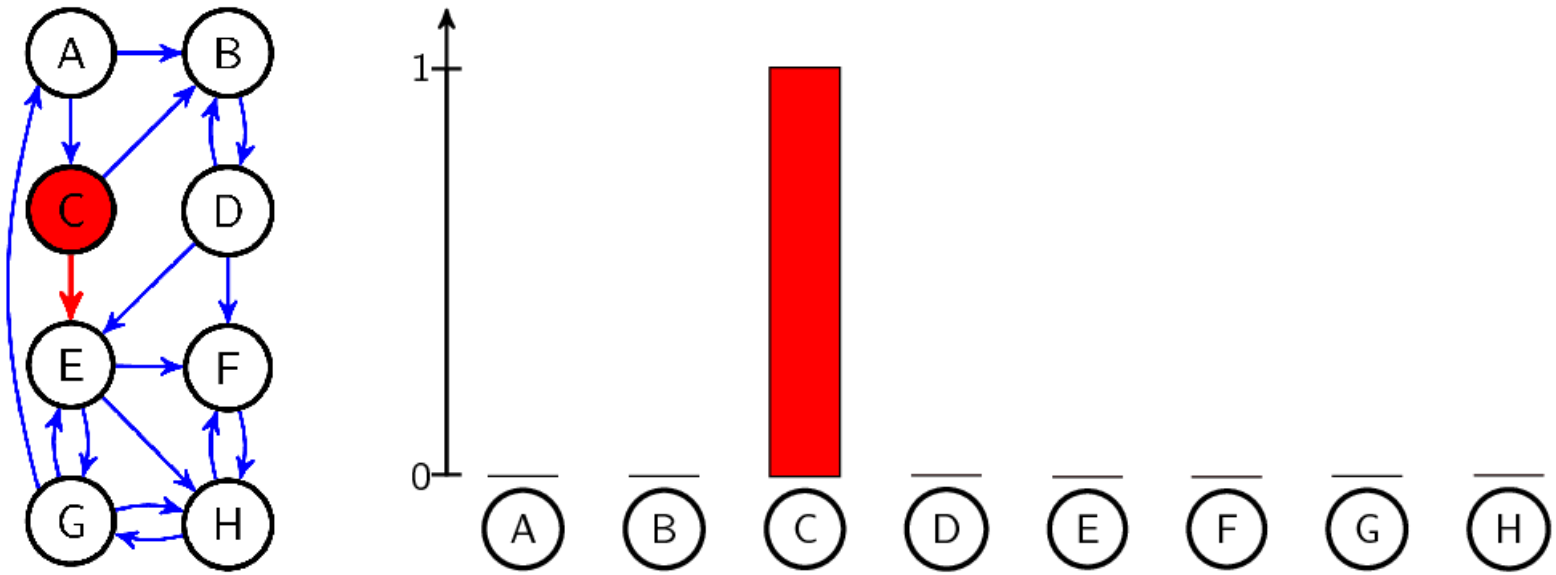


- začnu na C; 2 možná pokračování
- náhodně vyberu
- dostanu se do E; 3 cesty
- náhodně vyberu
- dostanu se do F
- . . . a tak dále

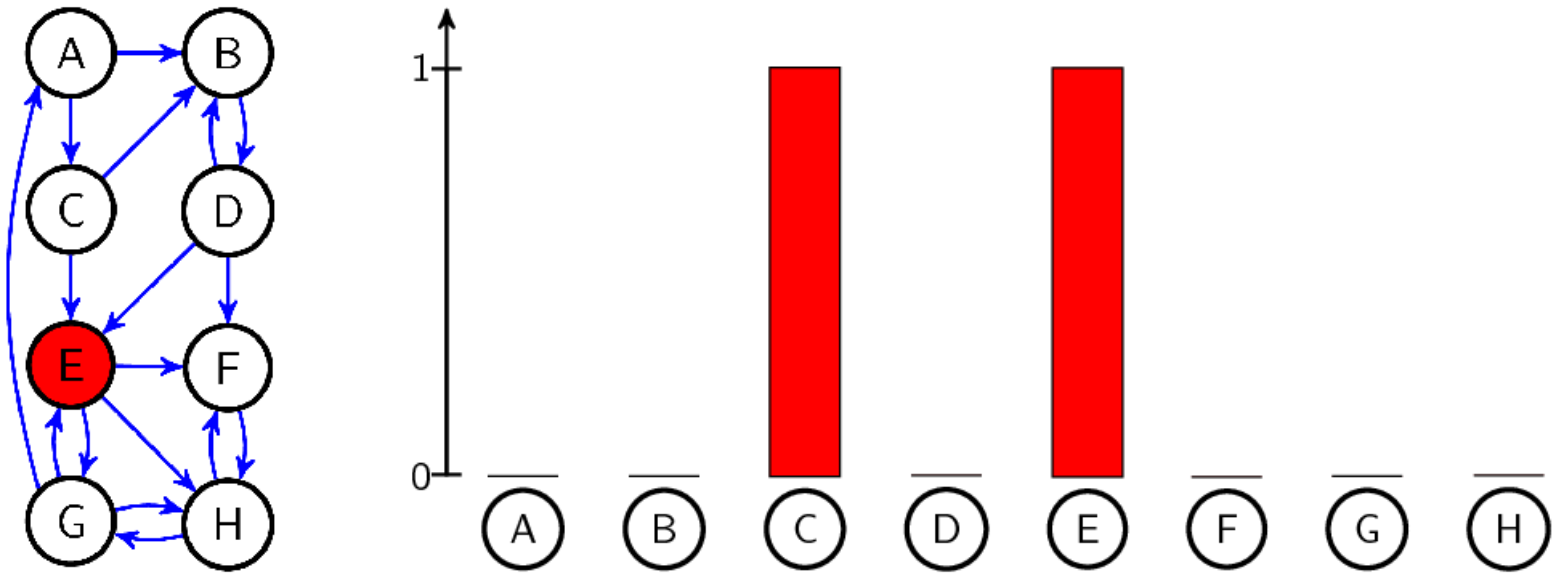
Nápad 2: Náhodná procházka, počítání průchodů



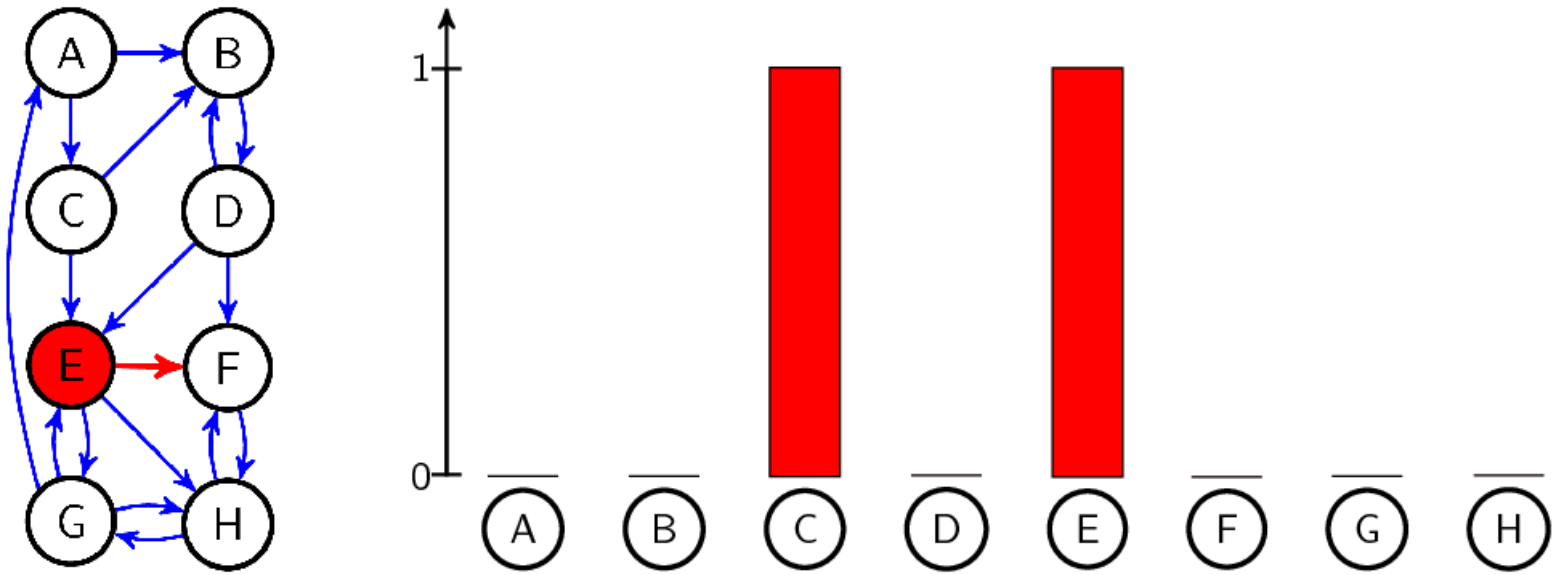
Nápad 2: Náhodná procházka, počítání průchodů



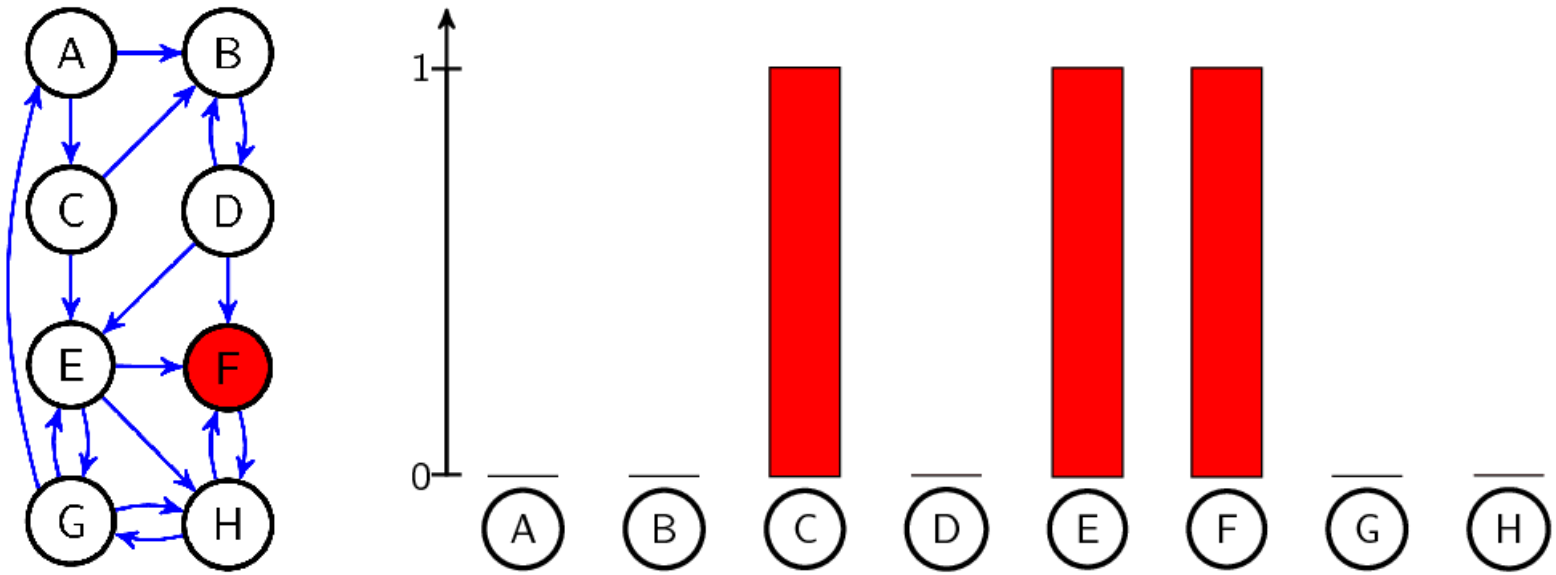
Nápad 2: Náhodná procházka, počítání průchodů



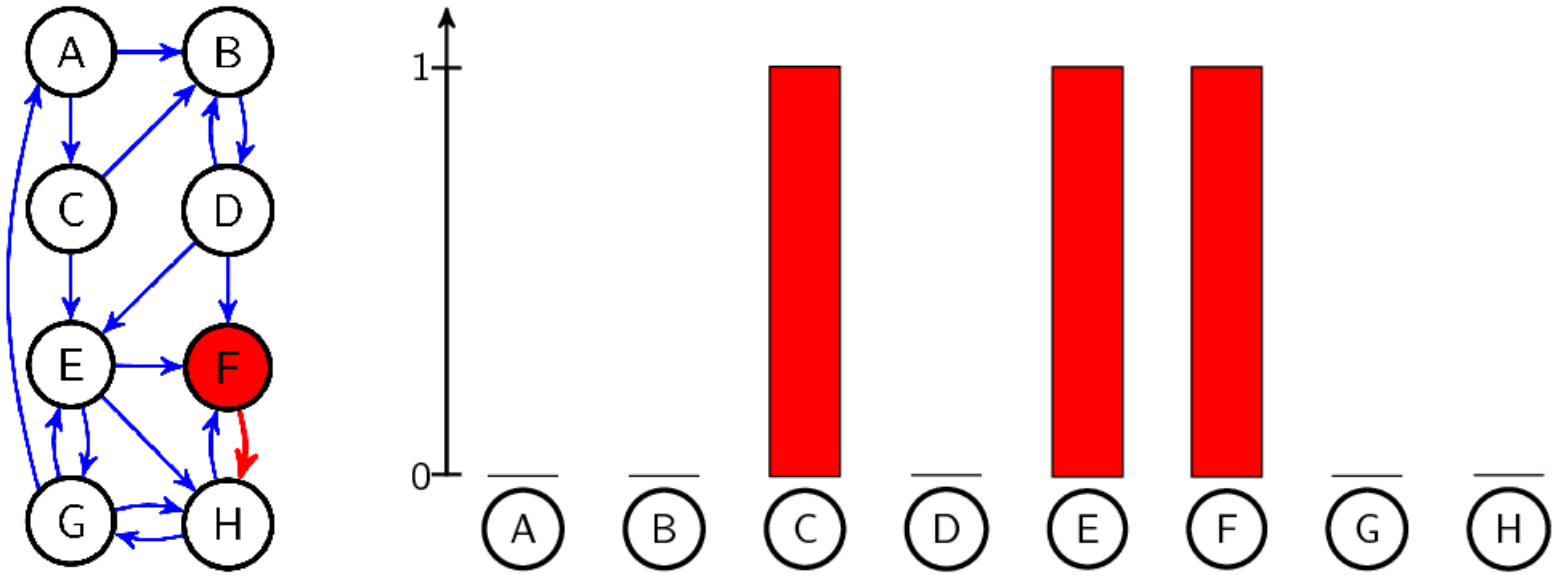
Nápad 2: Náhodná procházka, počítání průchodů



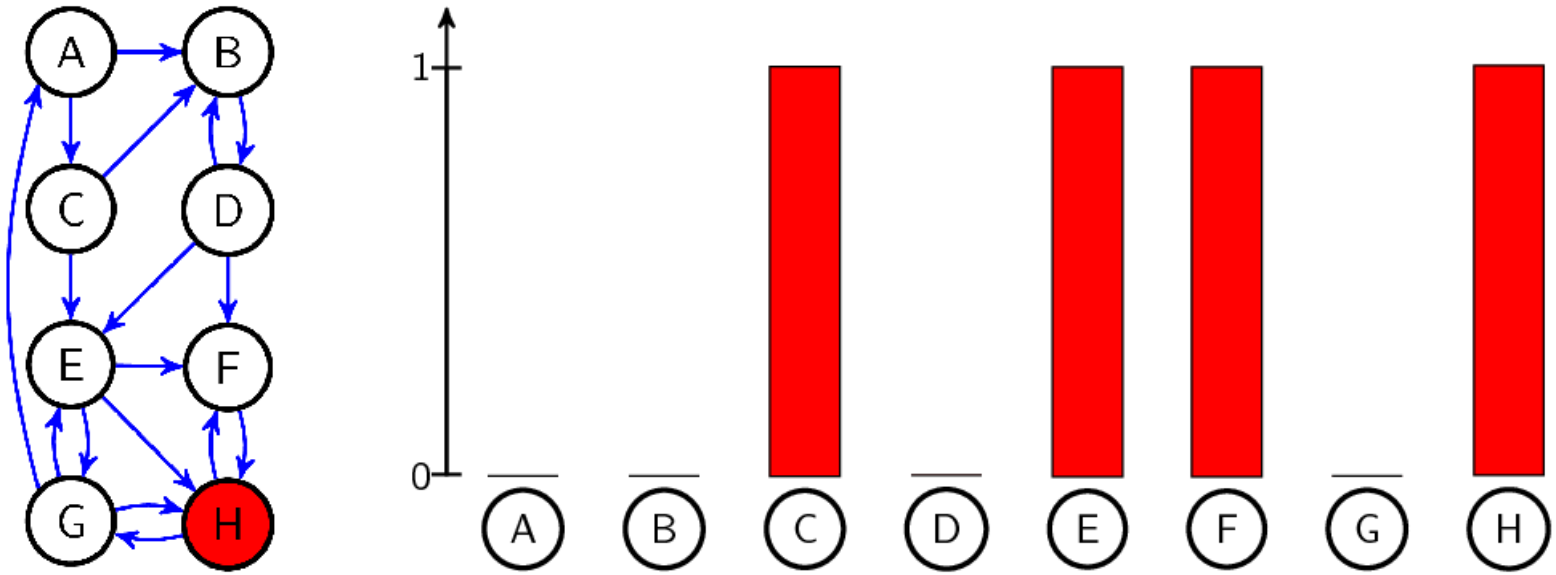
Nápad 2: Náhodná procházka, počítání průchodů



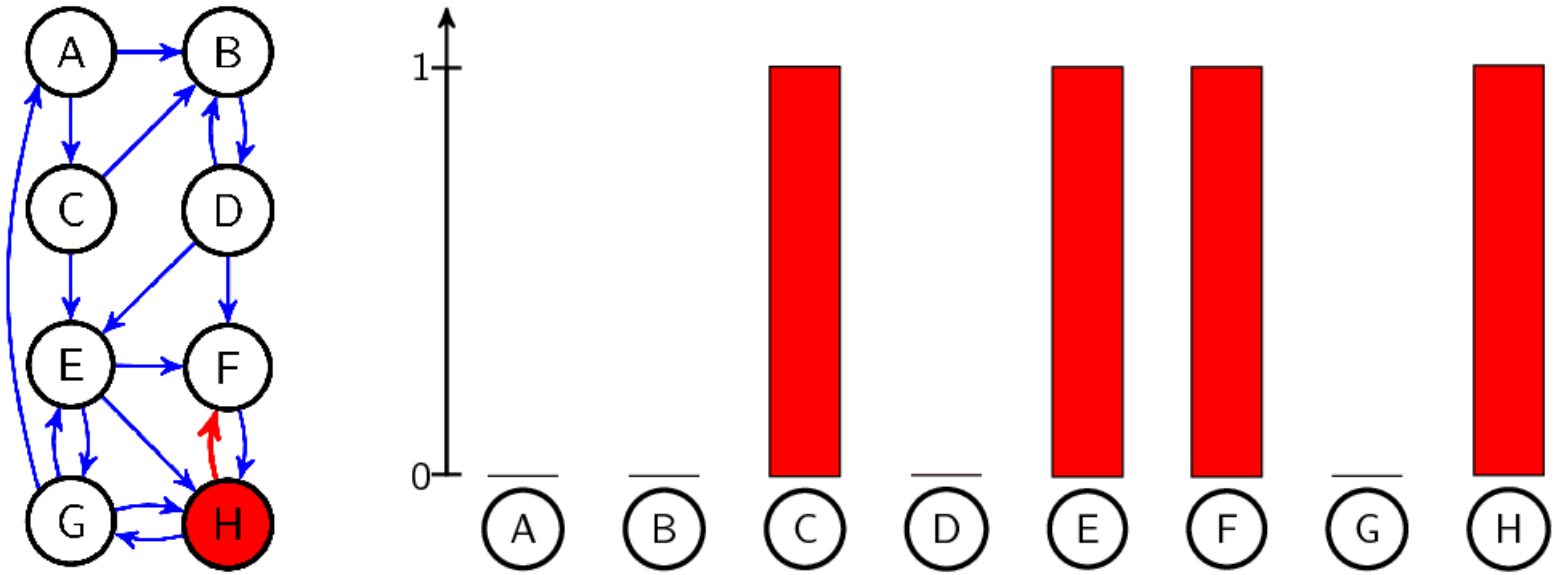
Nápad 2: Náhodná procházka, počítání průchodů



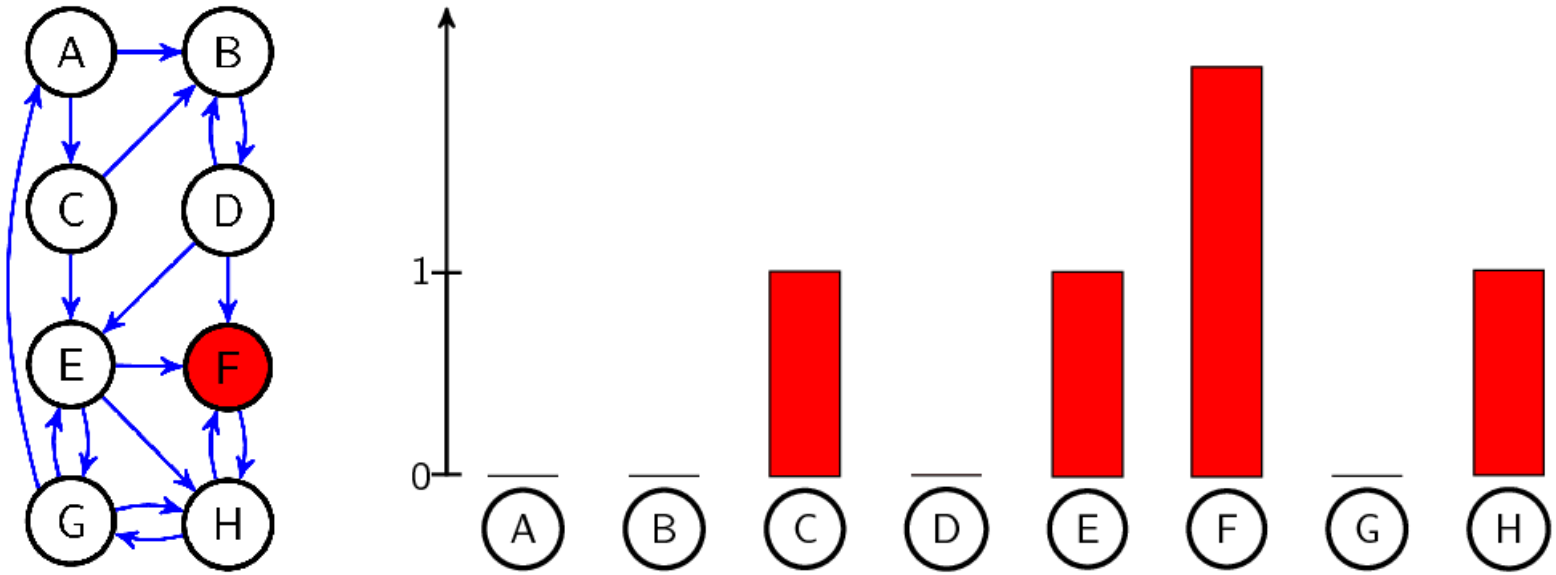
Nápad 2: Náhodná procházka, počítání průchodů



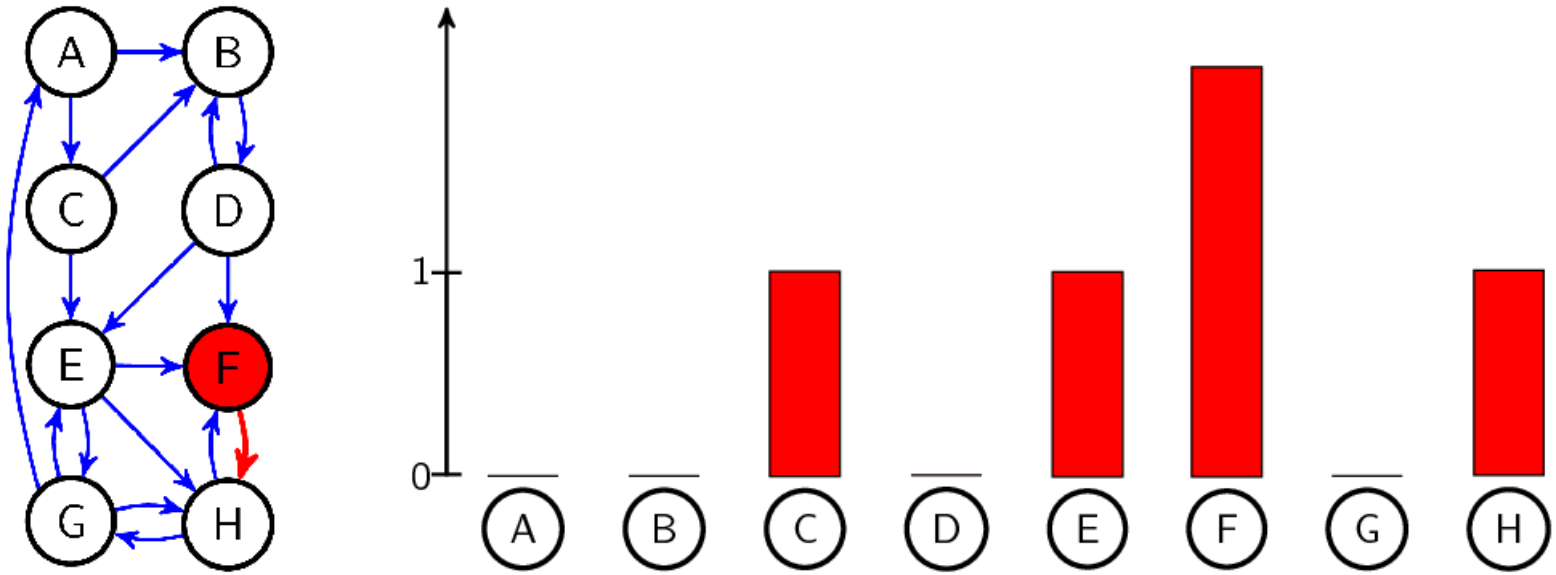
Nápad 2: Náhodná procházka, počítání průchodů



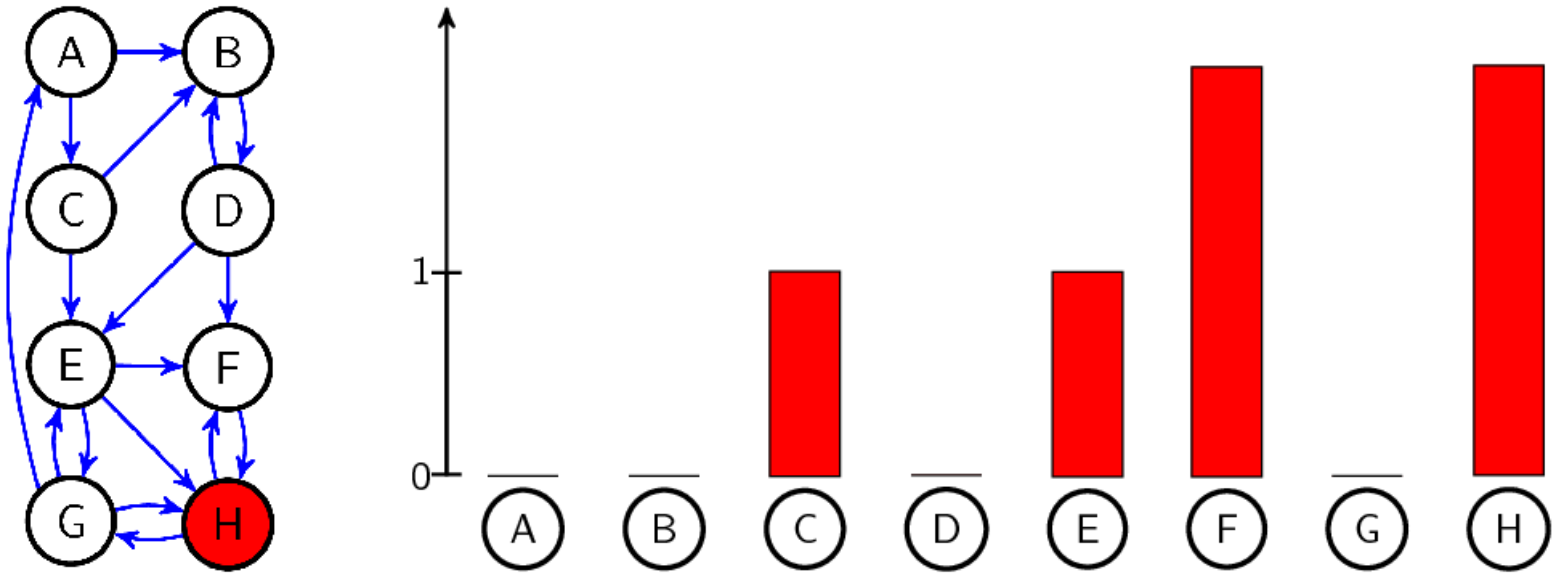
Nápad 2: Náhodná procházka, počítání průchodů



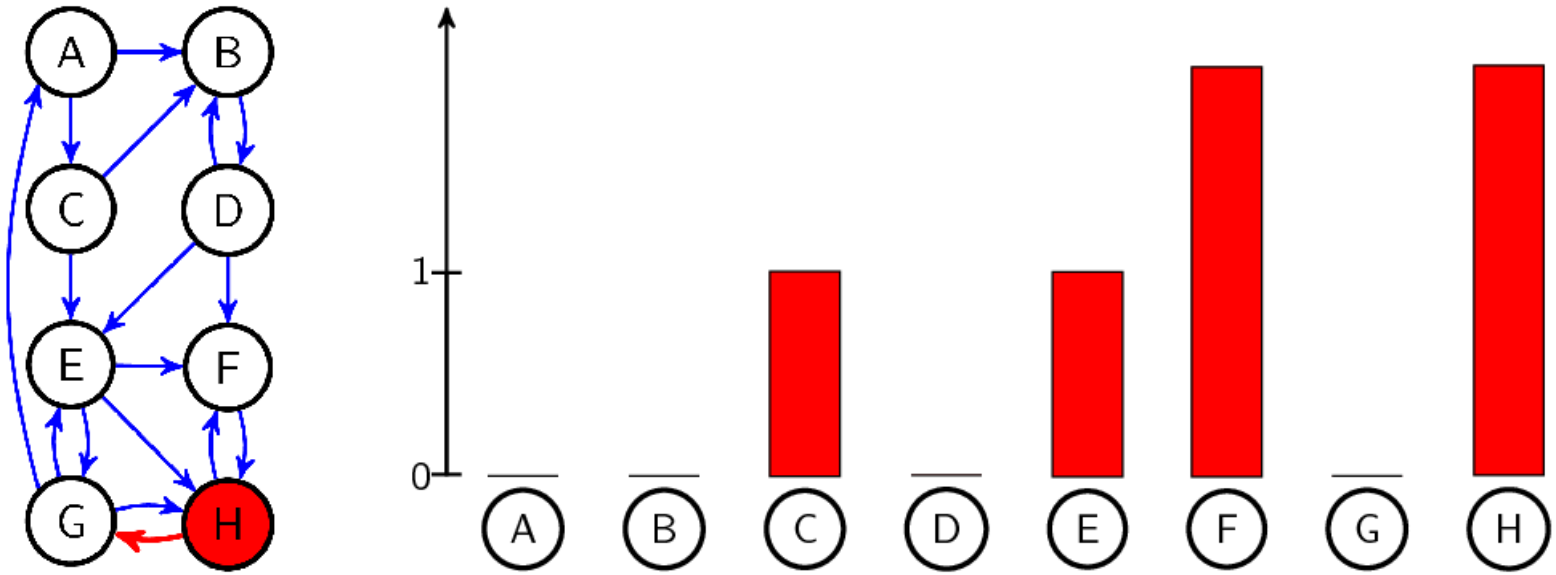
Nápad 2: Náhodná procházka, počítání průchodů



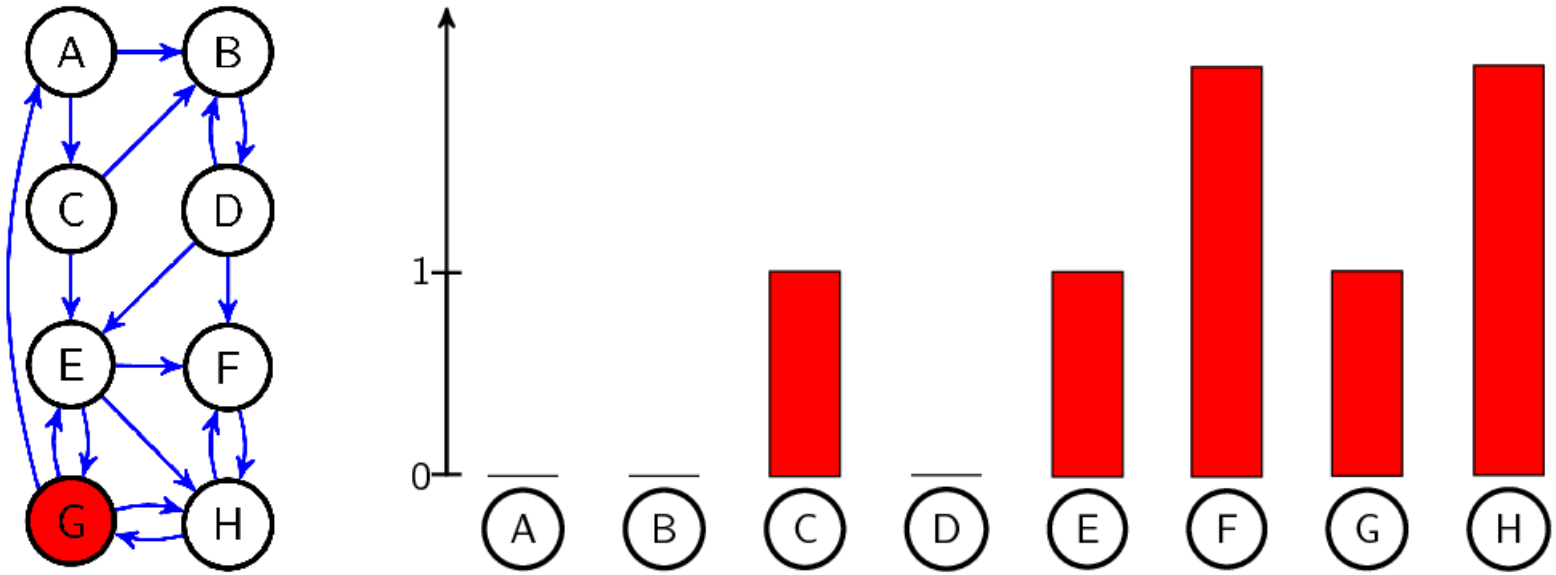
Nápad 2: Náhodná procházka, počítání průchodů



Nápad 2: Náhodná procházka, počítání průchodů



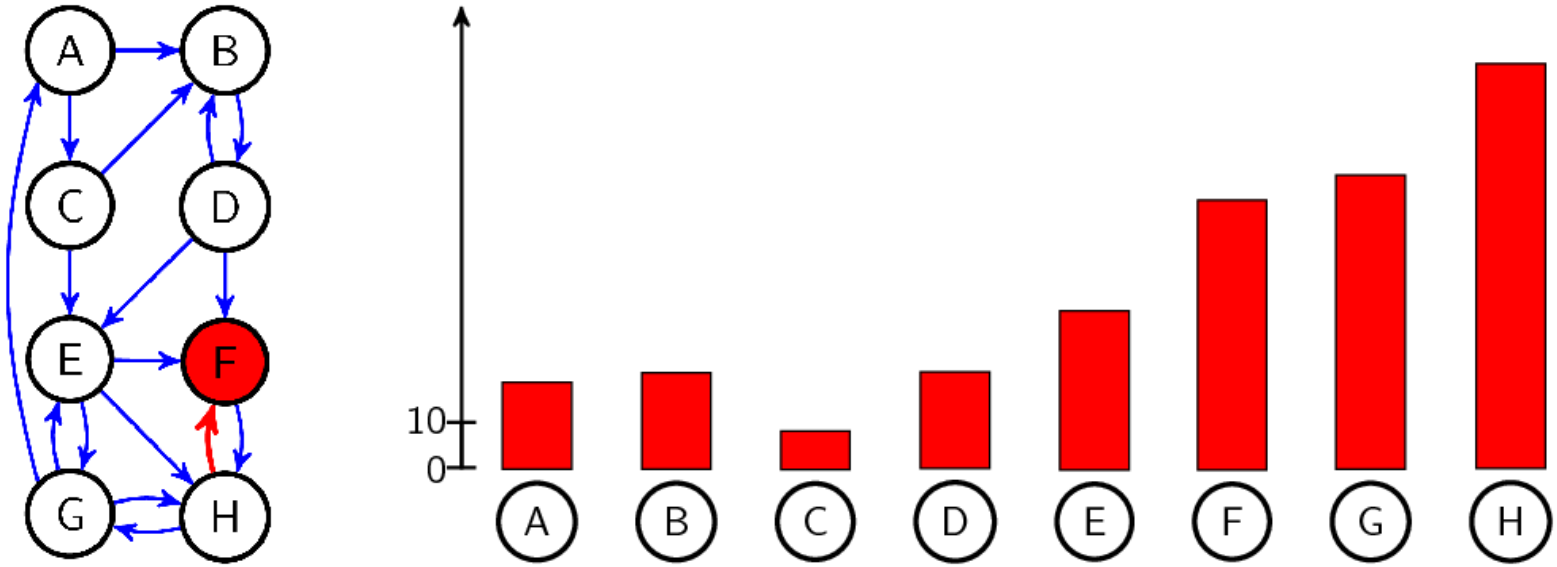
Nápad 2: Náhodná procházka, počítání průchodů



Nápad 2: Náhodná procházka, počítání průchodů

Nápad 2: Náhodná procházka, počítání průchodů

300 iterací



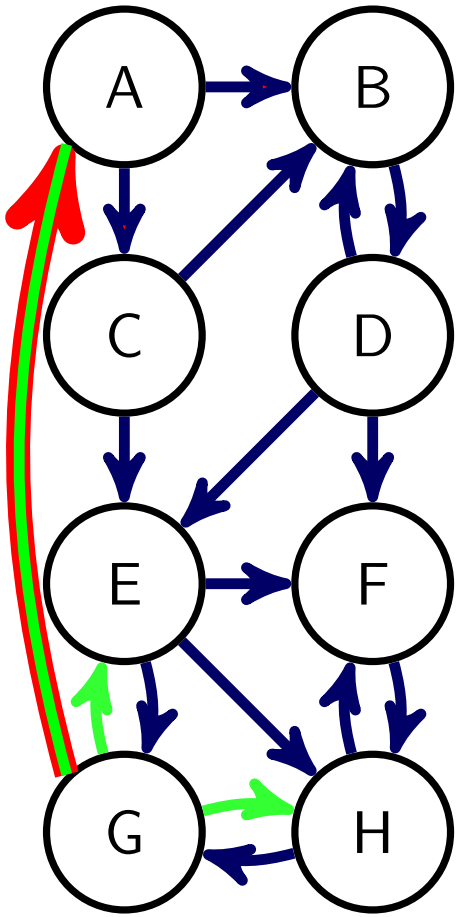
$$T = (6.0, 6.7, 2.7, 6.7, 11.0, 18.7, 20.3, 28.0)$$

Nápad 2: Náhodná procházka, počítání průchodů

$$T = (6.0, 6.7, 2.7, 6.7, 11.0, 18.7, 20.3, 28.0)$$

- dobře, z náhodné procházky dostaneme vždy počty průchodů
- co ta čísla jsou, je jasné. Mohu nějak vysvětlit, jak jednotlivá T souvisejí?

Nápad 2: Náhodná procházka, počítání průchodů



$T = (6.0, 6.7, 2.7, 6.7, 11.0, 18.7, 20.3, 28.0)$

- dobře, z náhodné procházky dostaneme vždy počty průchodů
- co ta čísla jsou, je jasné. Mohu nějak vysvětlit, jak jednotlivá T souvisejí?
- ano: pokud jsem například strávil na A čas T_A, mohlo se to stát jen proto, že jsem tam odněkud přišel, v tomto případě z G. Z G se chodí do A v 1/3 případů.
- $T_A = \frac{T_G}{3}$
- hmmm, to je zajímavé!

Naše dva nápady: Ekvivalence!

Nápad 1: Poměrné hlasování

- odvozeno na základě velmi jednoduchých úvah
- máme podmínky na to, jak mají spolu váhy W_k jednotlivých webů souviset
- zatím jsme nevěděli, jak tyto podmínky splnit a tak najít řešení

Nápad 2: Náhodná procházka

- přirozená úvaha při absenci dalších znalostí
- dokážeme modelovat problém, získat časy strávené na jednotlivých stránkách T_k
- teprve po krátké analýze se ukázalo, že $T_k = W_k$ a tedy Nápad 2 řeší soustavu rovnic z Nápadu 1!

Systém $W_k = \sum P_{kb} W_b$

Chceme, aby platilo

$$W_k = \sum_{b \in \{\to k\}} \frac{W_b}{n_b}$$

Systém $W_k = \sum P_{kb} W_b$

Chceme, aby platilo

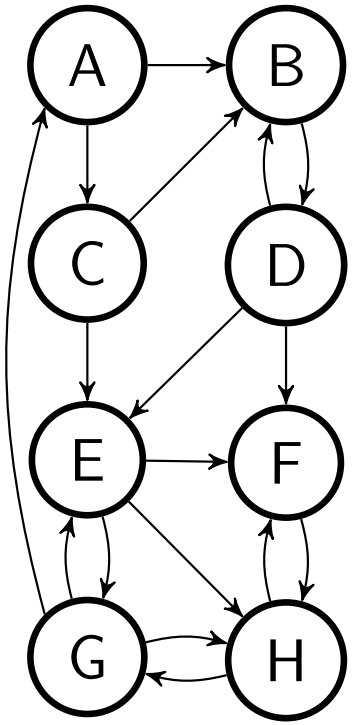
$$W_k = \sum_{b \in \{\to k\}} \frac{W_b}{n_b} = \sum_b P_{kb} W_b \,,$$

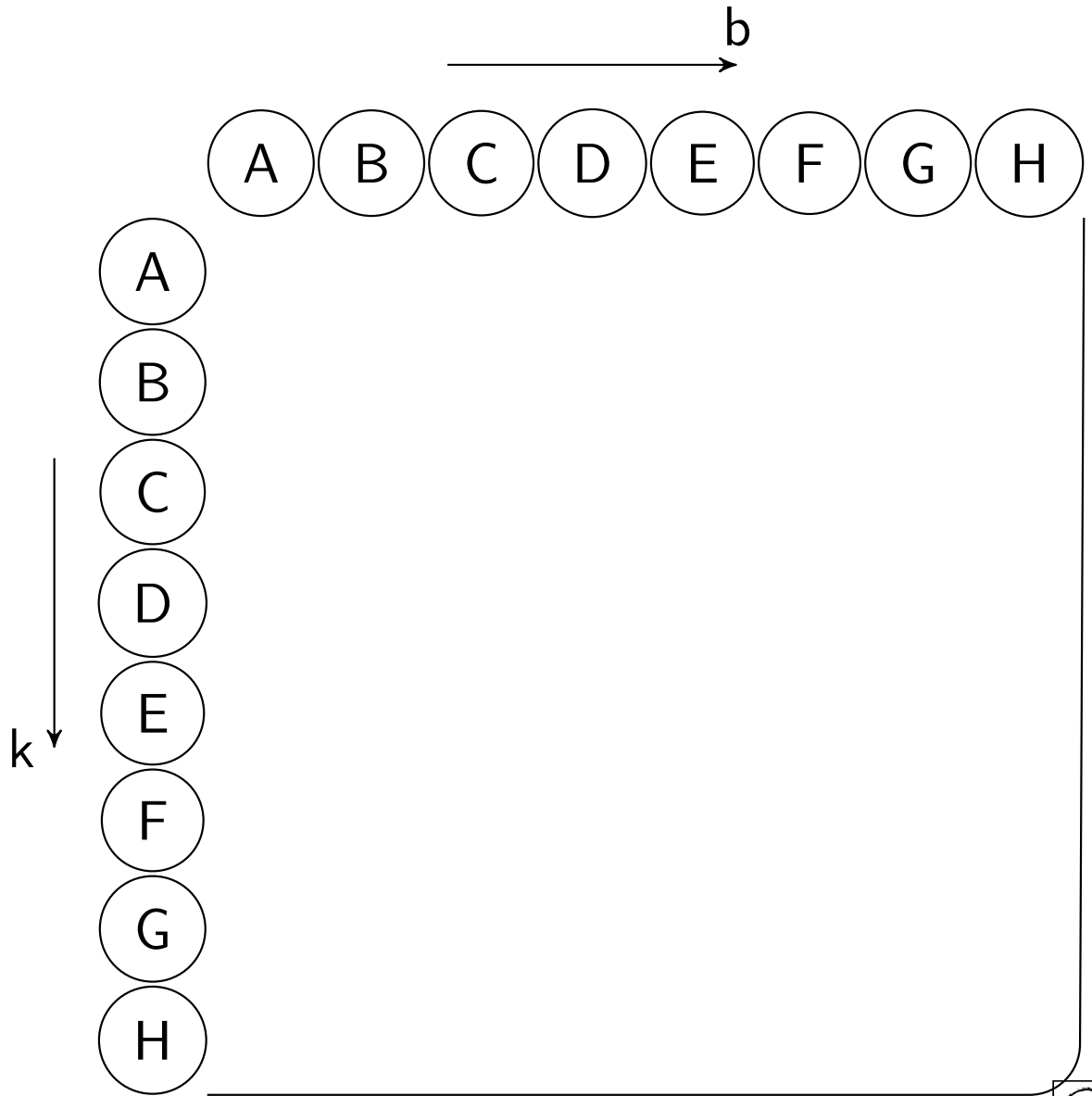
což je ekvivaletní maticovému zápisu

$$\mathbf{W} = \mathbf{P}\mathbf{W} \,.$$

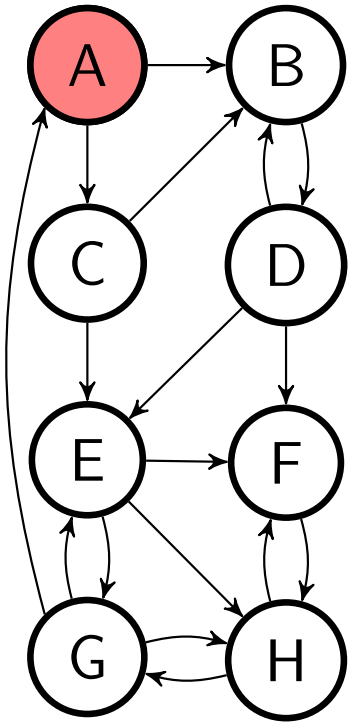
- P_{kb} je nenulové tam, kde existuje odkaz $b \to k$.
- je rovno $1/$(počet odkazů z b)

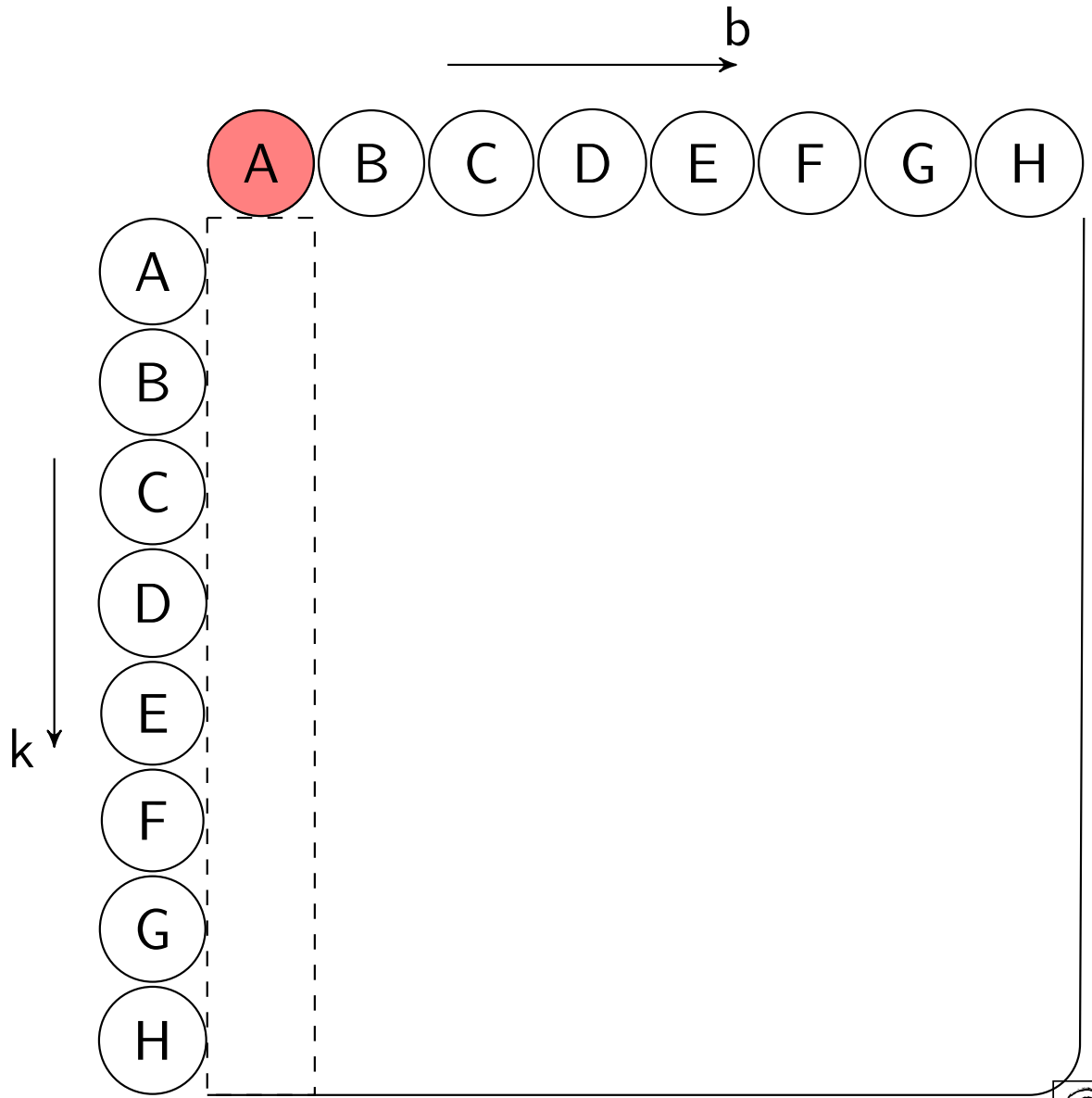
Konstrukce matice **P**, matice sousednosti



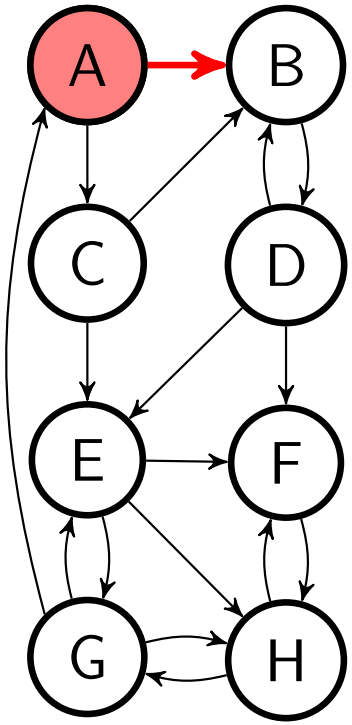


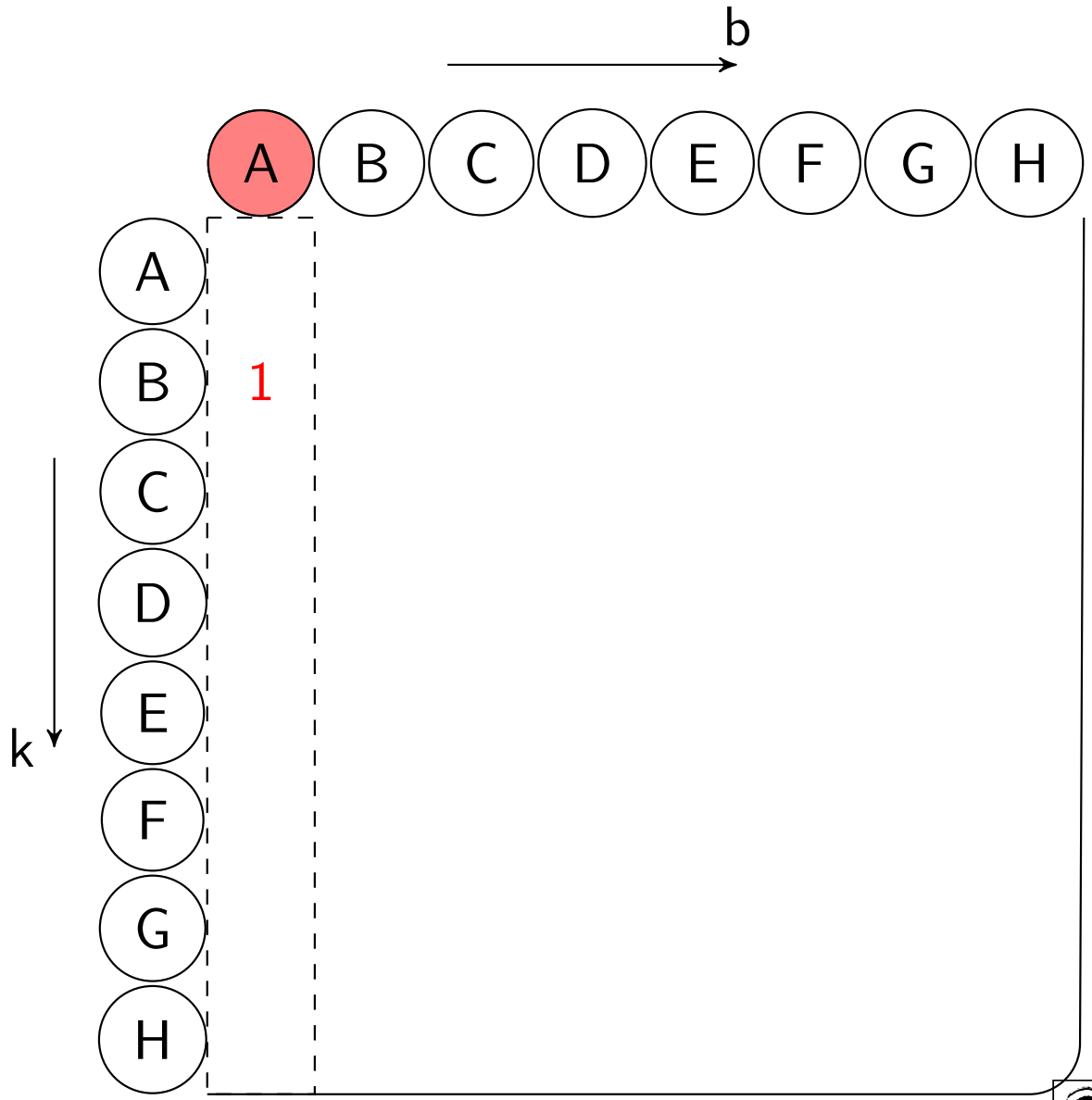
Konstrukce matice **P**, matice sousednosti



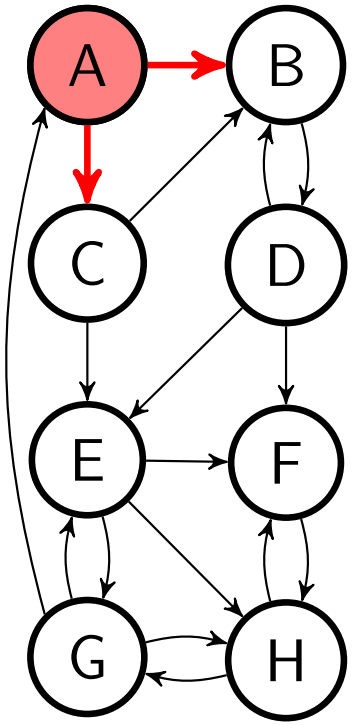


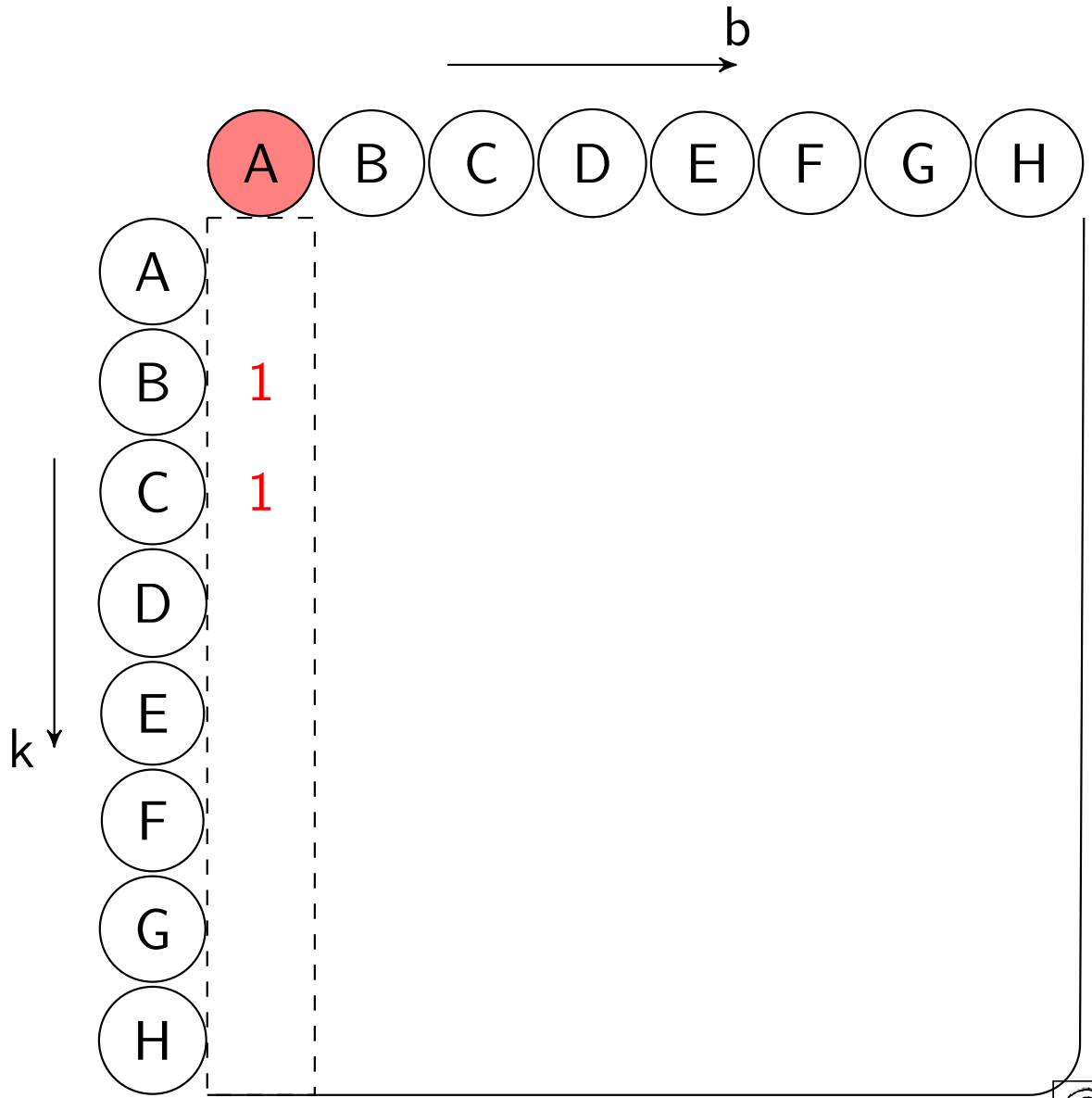
Konstrukce matice **P**, matice sousednosti



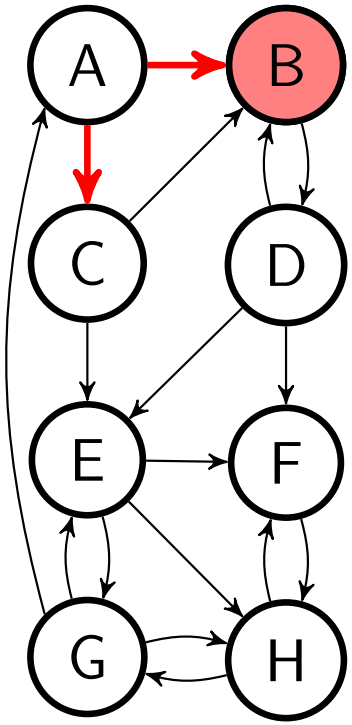


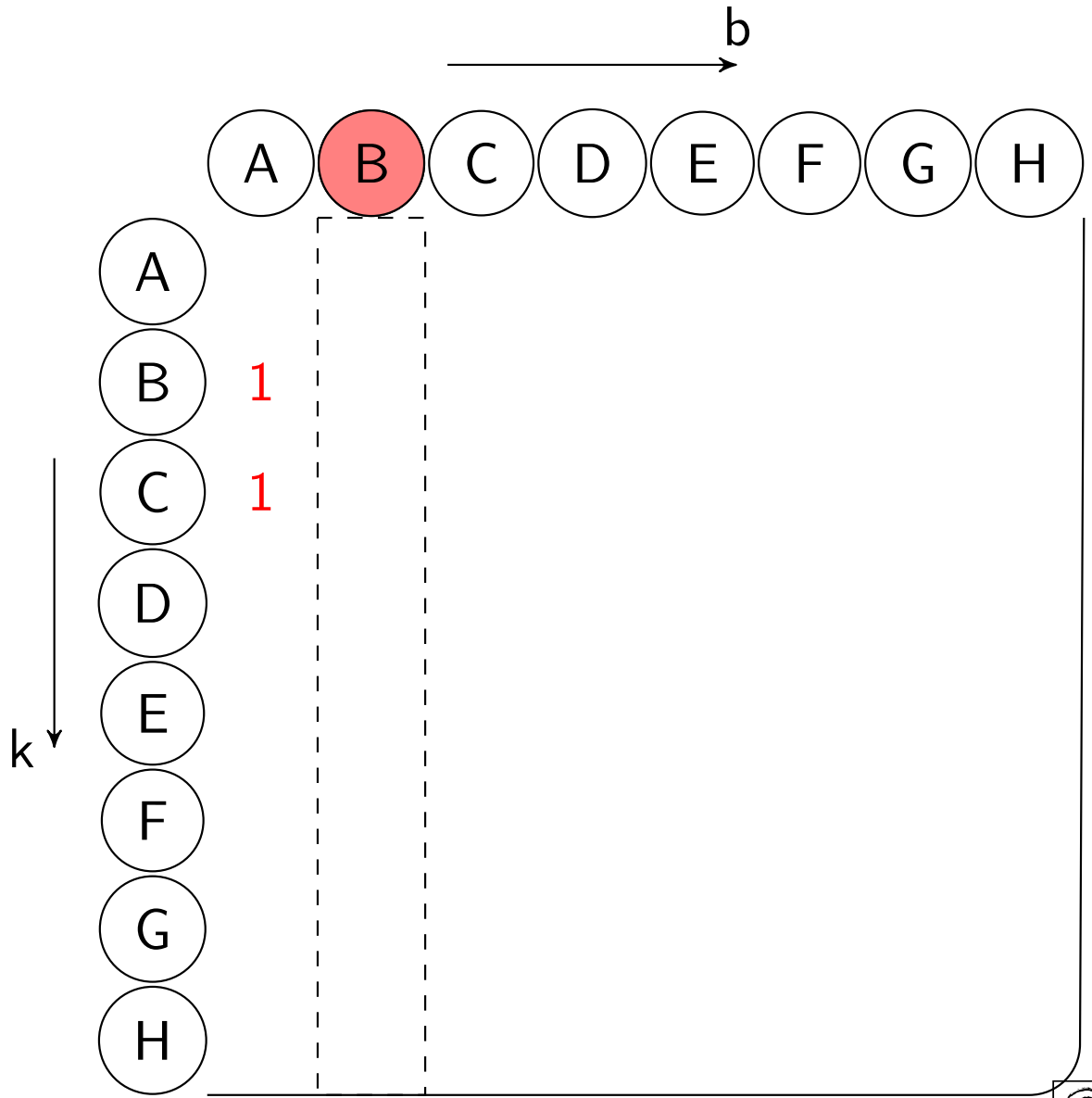
Konstrukce matice **P**, matice sousednosti



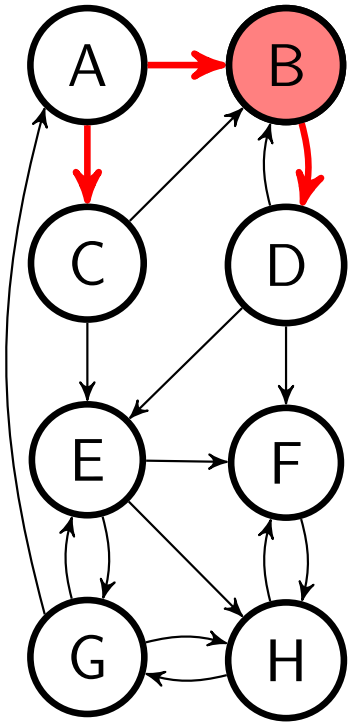


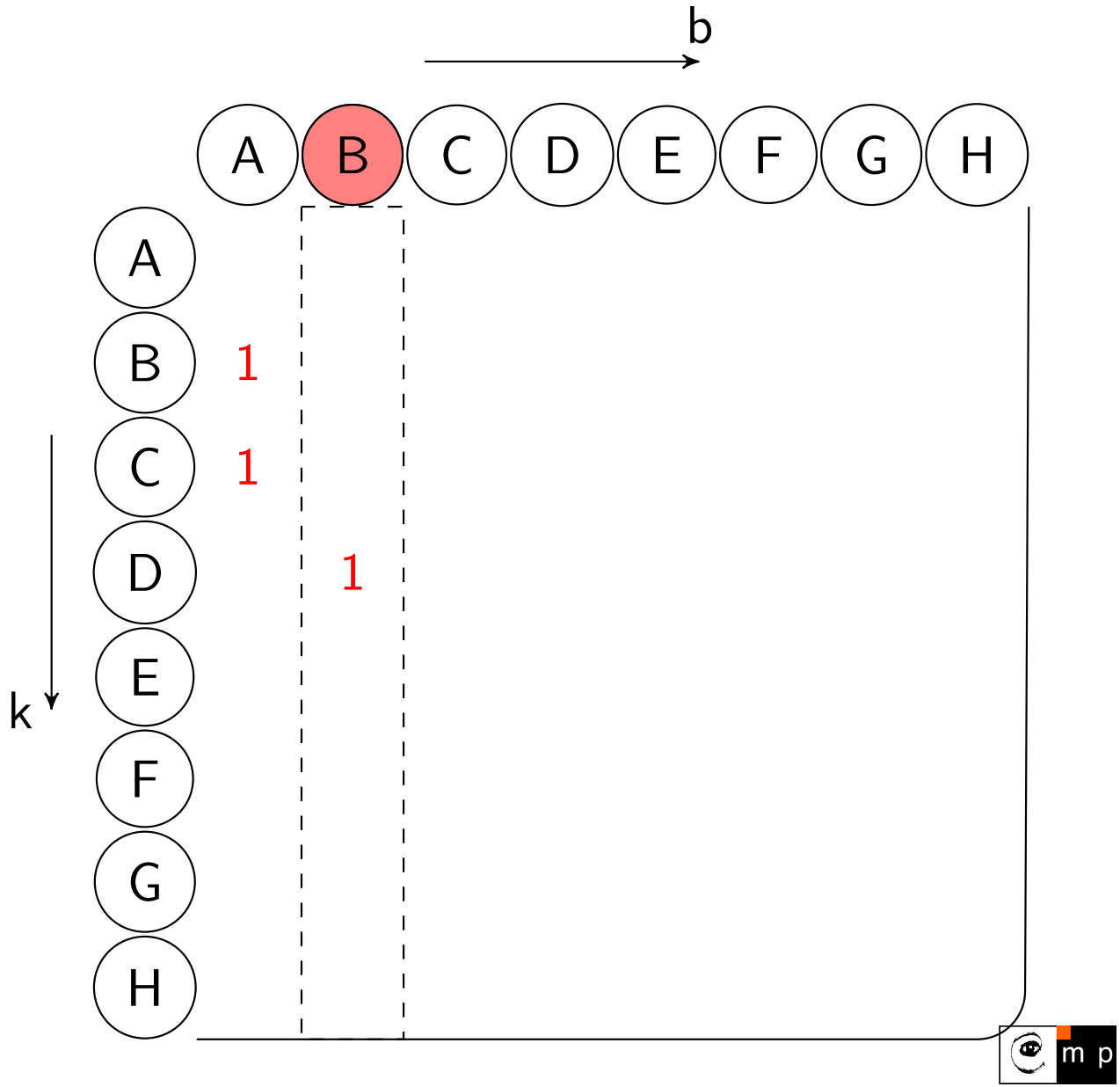
Konstrukce matice **P**, matice sousednosti



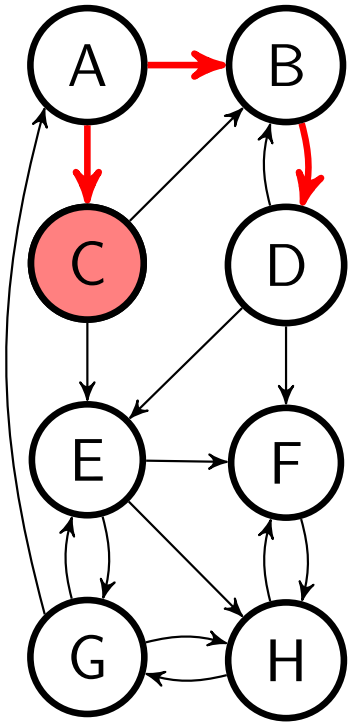


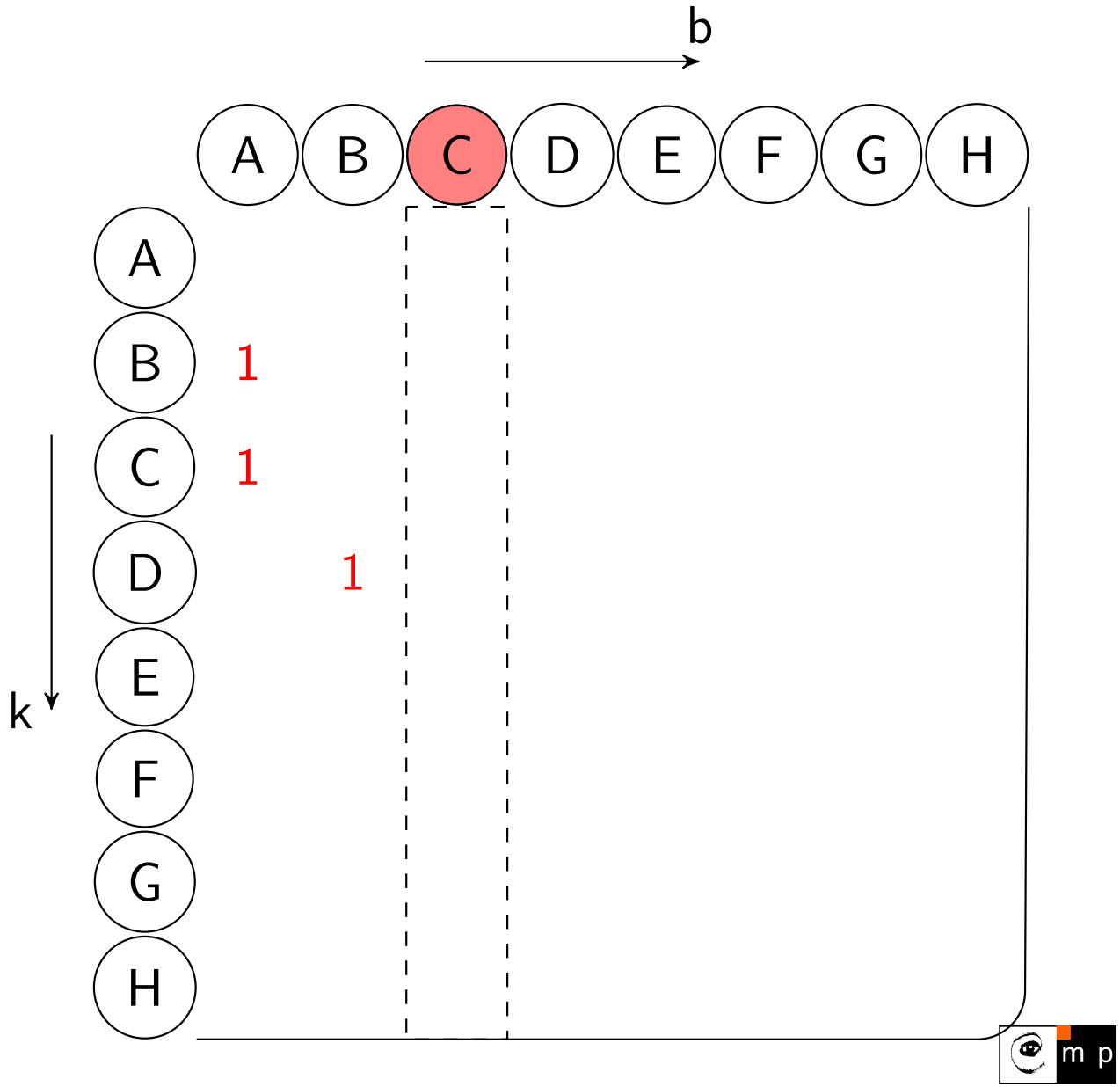
Konstrukce matice **P**, matice sousednosti



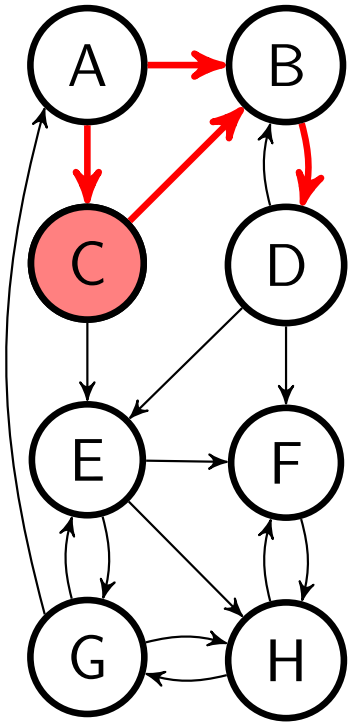


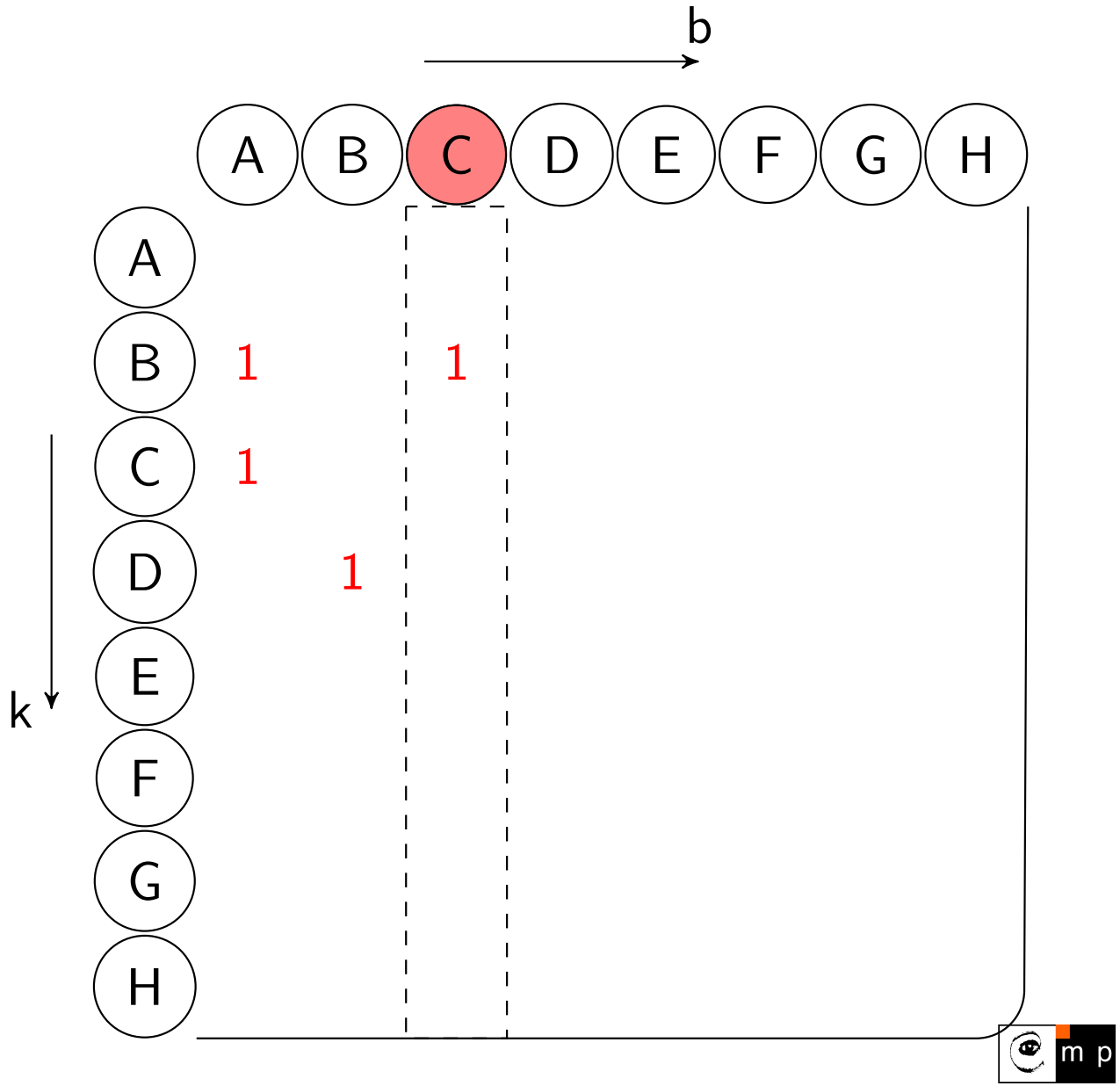
Konstrukce matice **P**, matice sousednosti



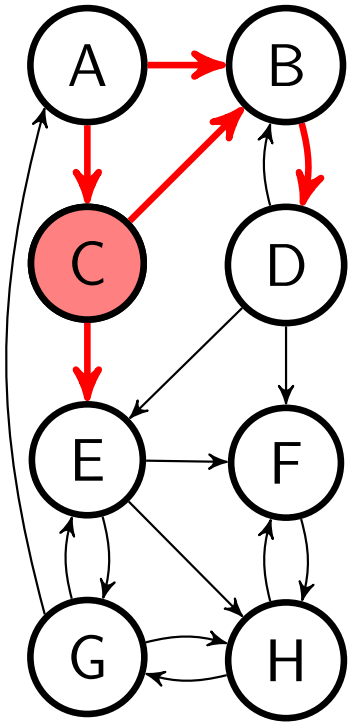


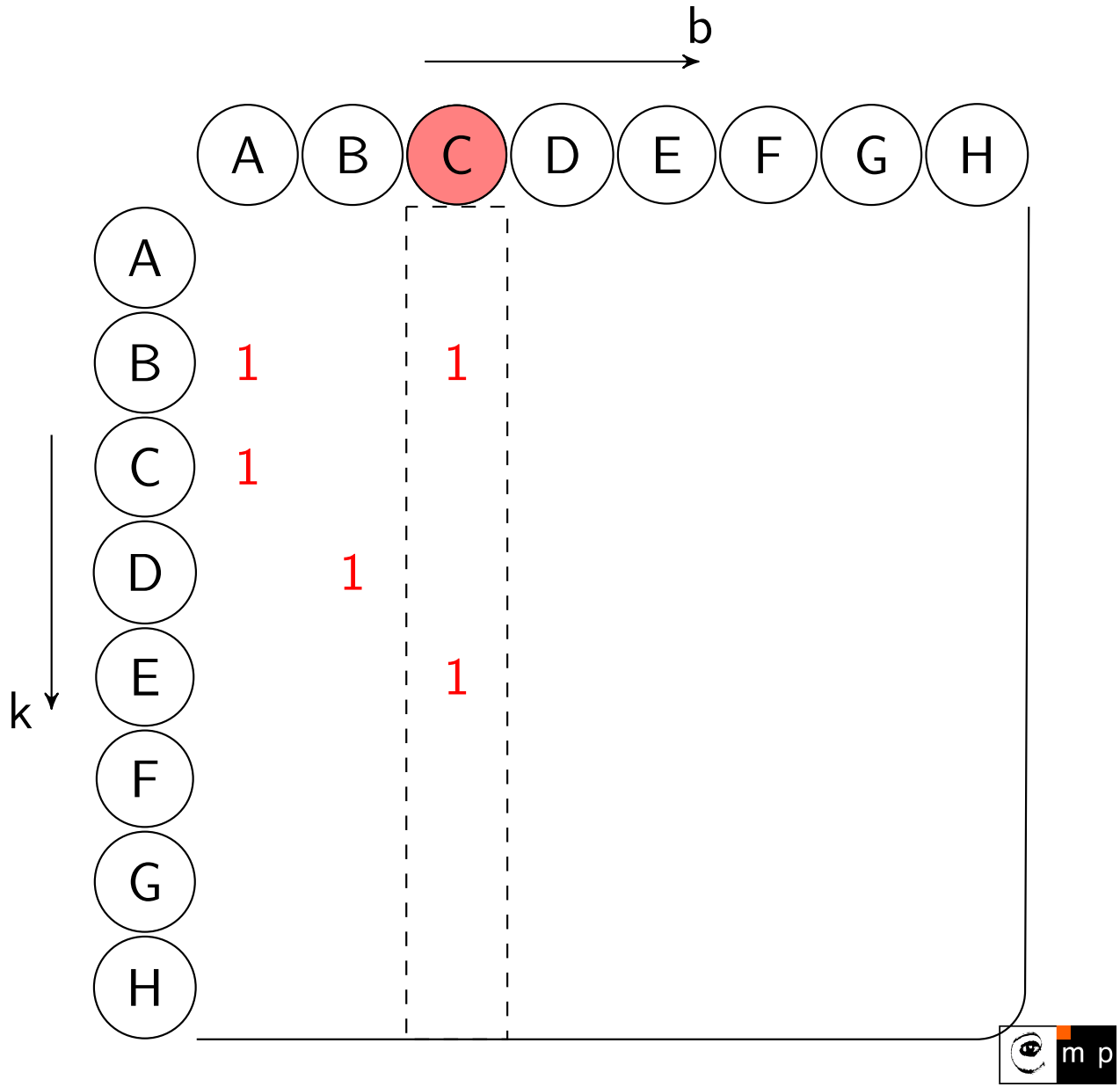
Konstrukce matice **P**, matice sousednosti



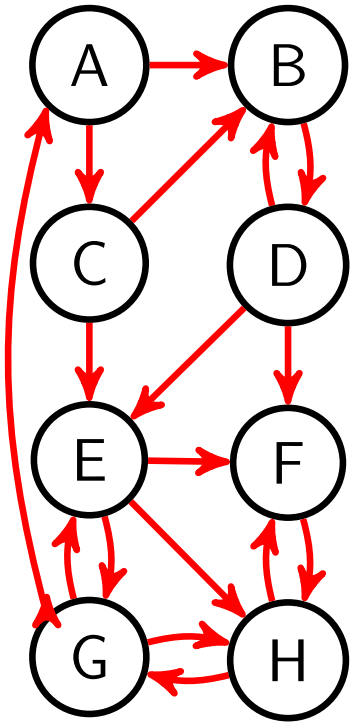


Konstrukce matice **P**, matice sousednosti

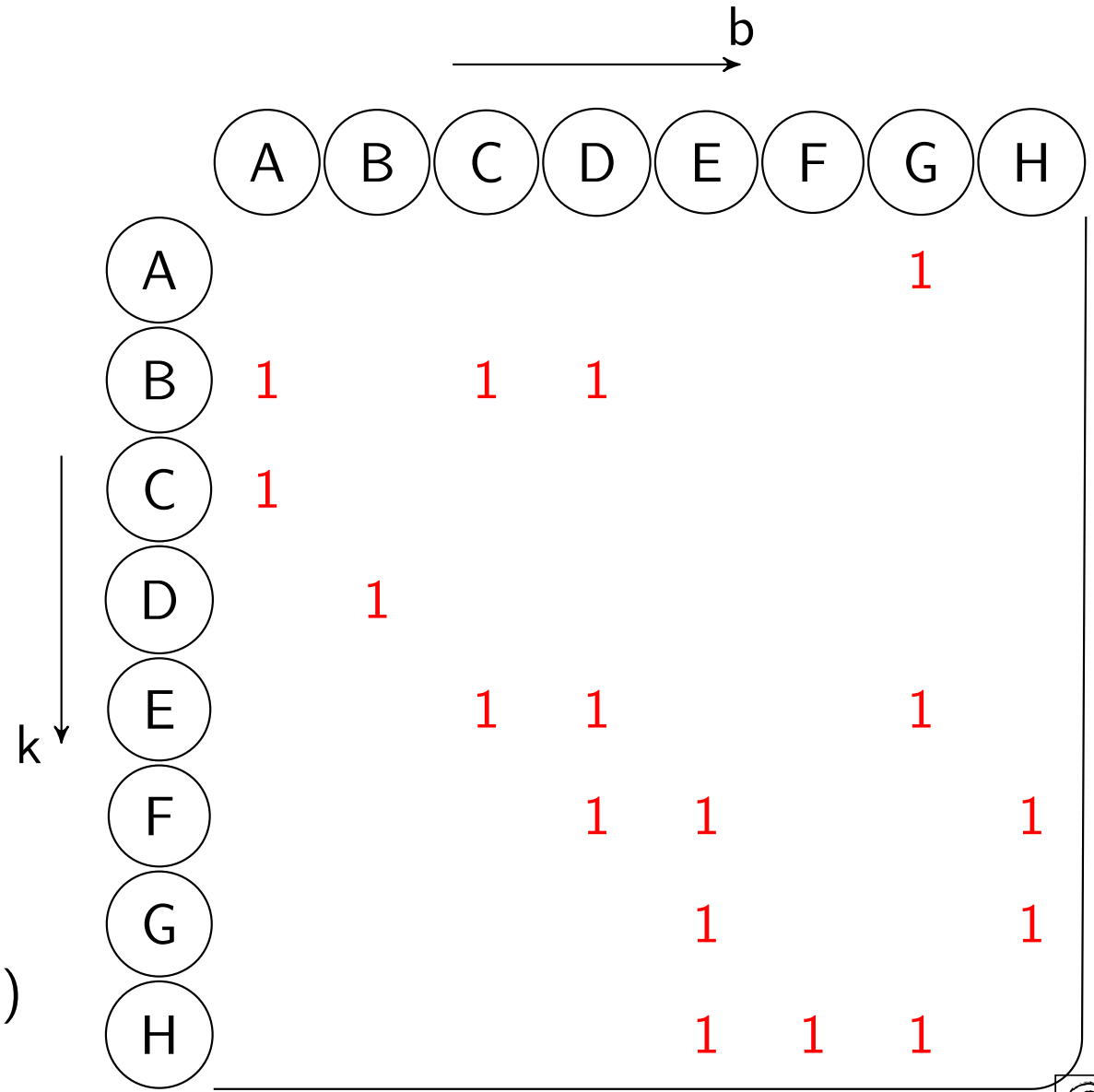




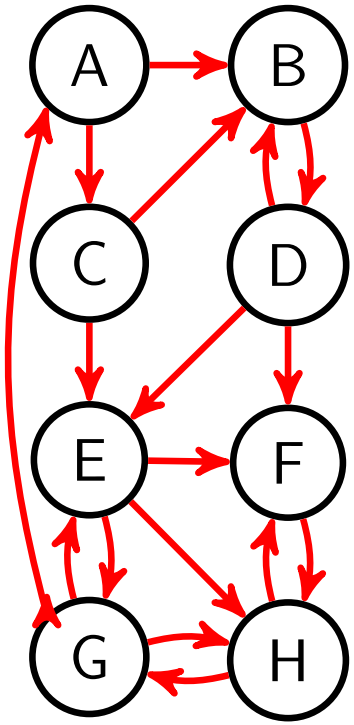
Konstrukce matice **P**, matice sousednosti



adjacency matrix
(matice sousednosti)



Konstrukce matice **P**, normování



b

k \ b	A	B	C	D	E	F	G	H
A							$\frac{1}{3}$	
B	$\frac{1}{2}$		$\frac{1}{2}$	$\frac{1}{3}$				
C	$\frac{1}{2}$							
D		1						
E			$\frac{1}{2}$	$\frac{1}{3}$			$\frac{1}{3}$	
F				$\frac{1}{3}$	$\frac{1}{3}$			$\frac{1}{2}$
G					$\frac{1}{3}$			$\frac{1}{2}$
H					$\frac{1}{3}$	1	$\frac{1}{3}$	

Systém $W_k = \sum P_{kb} W_b$: řešení

$$\mathbf{W} = \mathbf{PW}$$

- ale vždyť to vypadá jako rovnice pro vlastní vektory

Systém $W_k = \sum P_{kb} W_b$: řešení

$$\lambda \mathbf{W} = \mathbf{P}\mathbf{W}$$

- ale vždyť to vypadá jako rovnice pro vlastní vektory ($\lambda = 1$)!

Systém $W_k = \sum P_{kb} W_b$: řešení

$$\lambda \mathbf{W} = \mathbf{P}\mathbf{W}$$

- ale vždyť to vypadá jako rovnice pro vlastní vektory ($\lambda = 1$)!
- existuje však vlastní vektor pro $\lambda = 1$?
- ano:

$$\forall b\colon \sum_k P_{kb} = 1$$

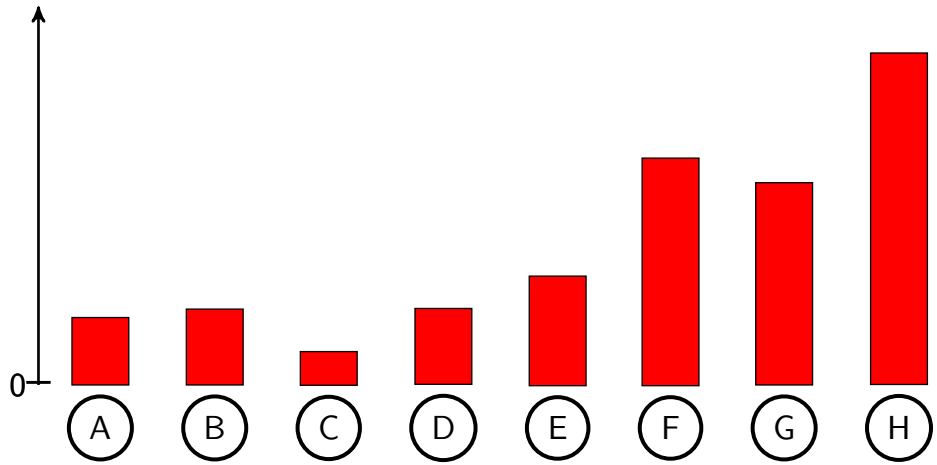
to je však totéž, co

$$\mathbf{P}^T \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

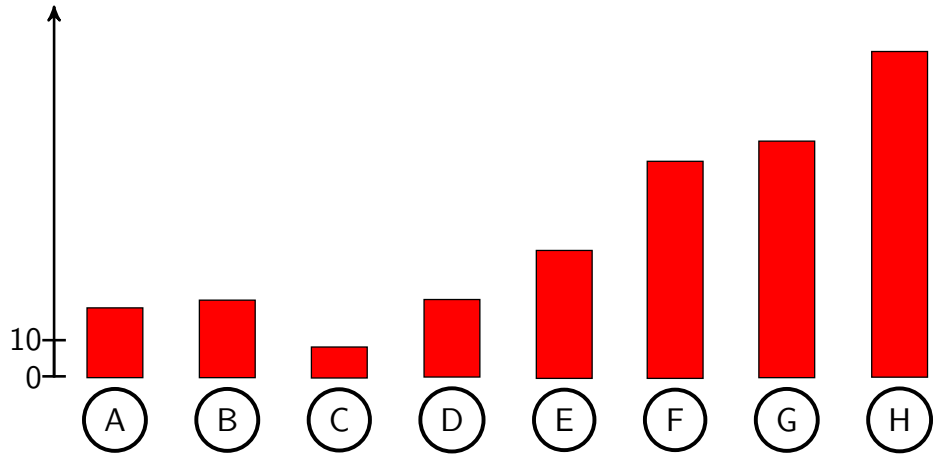
a vlastní čísla matic a transponovaných matic jsou stejná.

Vlastní vektor $\mathbf{v}_1$

- vypočtený vektor $\mathbf{v}_1$



- srovnání s četností navšívení jednotlivých webů (náhodná procházka, 900 iterací)



Systém $\mathbf{W} = \mathbf{PW}$: jednoznačnost a výpočet

- lineární algebra dává další vhled na to, kdy je řešení jednoznačné.
- pro jednoznačnost je potřeba, aby se matice upravila přičtením konstantní matice
- to odpovídá tomu, že náhodná procházka většinou probíhá po odkazech, někdy s náhodnými přeskoky

Co plyne z odvození PageRanku?

- úžasný efekt mají mnohonásobné pohledy na věc. Jedno může pohánět druhé, analýza mohutně profituje.
- jednoduchá myšlenka, přístupná každému, která změnila svět
- neznamená to ale, že na ni každý přijde (známý efekt "no jistě!", který se dostavuje po odhalení řešení). Dnes nám přijde velice přirozené, že všechny souřadné soustavy by měly být rovnocenné (Einsteinova teorie relativity)
- jednoduché věci (většinou) fungují
- je to potvrzení toho, že je třeba analyzovat problém postupně a postupovat od principielně jednoduššího řešení ke složitějšímu

Google story

- 1995: Larry Page a Sergey Brin se potkali na Stanfordu (bylo jim kolem 21 let)
- 1996-1997: odvození, implementace hledacího stroje, vymyšlení názvu Google
- 1998: 'lepší než cokoli jiného'
- 2004: IPO na Wall St, \$85/akcie (market cap \$25 miliard)
- od počátku skutečný leader v inovacích

Reference

- wikipedia
- David Austin: How Google Finds Your Needle in the Web's Haystack
- KURT BRYAN AND TANYA LEISE: The $25,000,000,000 Eigenvector: The Linear Algebra Behind Google