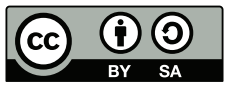
brmiversity: Umělá inteligence a teoretická informatika

Přednáška č. 5

Petr Baudiš ⟨pasky@ucw.cz⟩

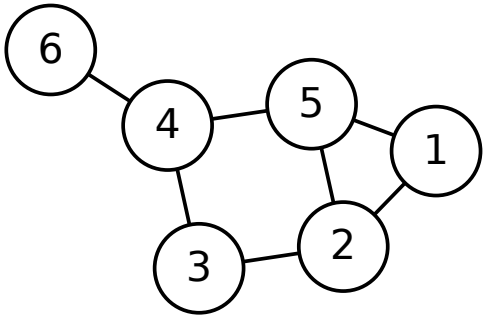
brmlab 2011



brmlab hackerspace prague

Outline

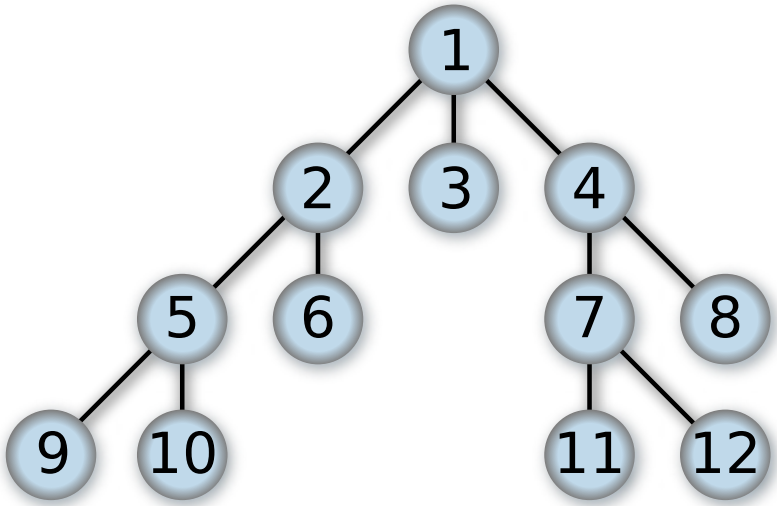
Prohledávání v grafech

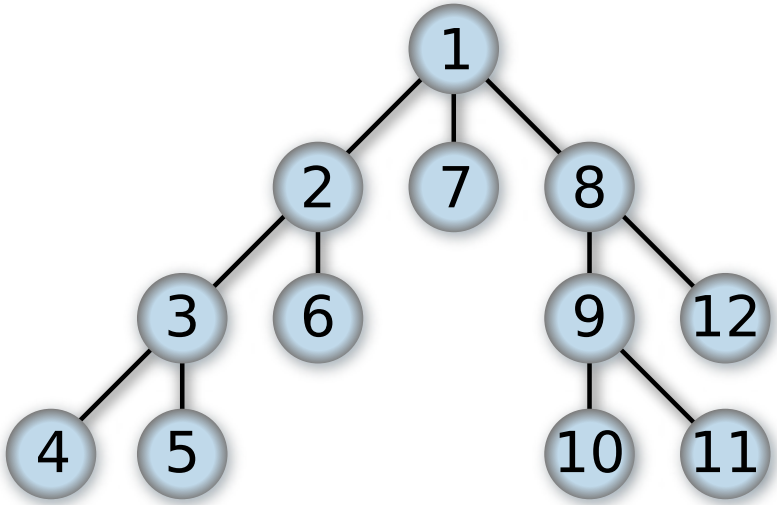


- Chceme projít celý graf
- Graf může být popsaný implicitně (průběh výpočtu, hry)
- Něco *hledáme* — určitý uzel (řešení), cestu, ...
- Hledání uzlu: BFS (prohledávání do šířky), DFS (prohledávání do hloubky)
- Nejkratší cesta: Dijkstra

brmlab hackerspace prague

BFS, DFS





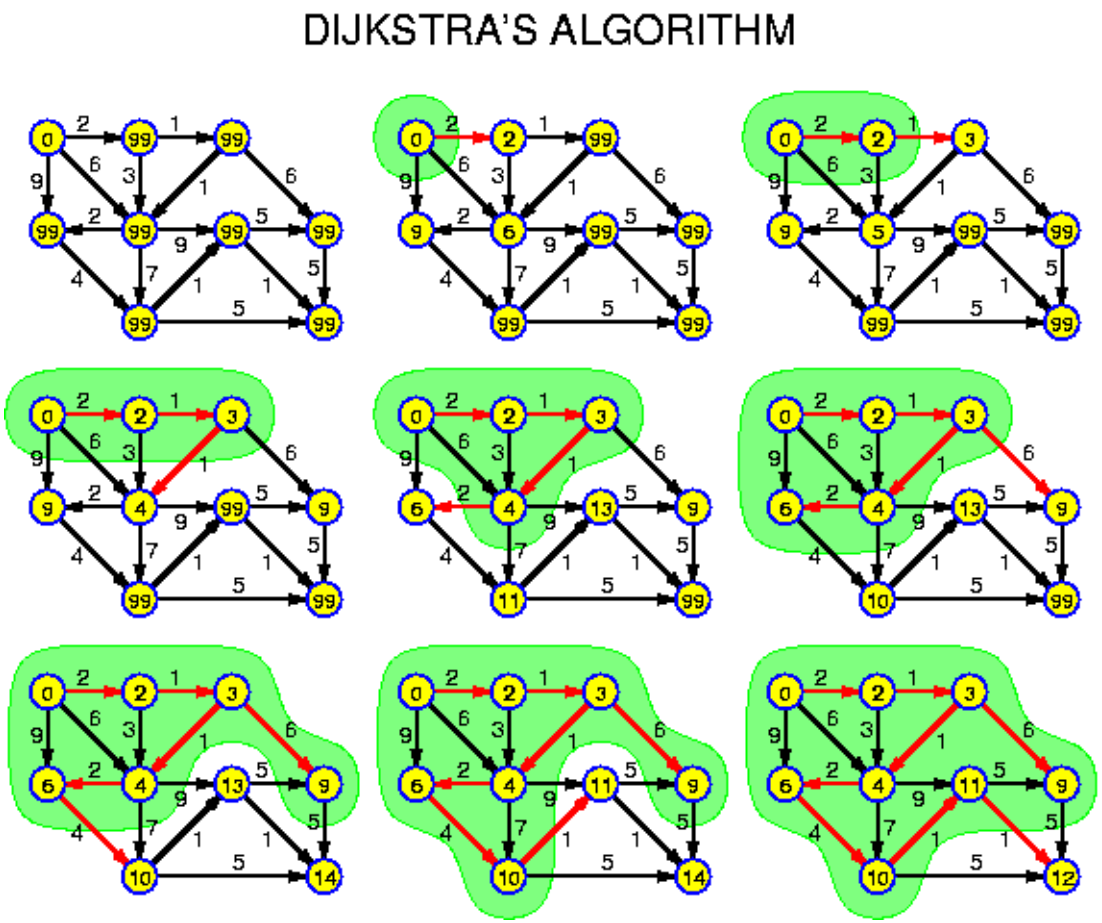
brmlab hackerspace prague

BFS, DFS

- BFS i DFS jsou úplné; fronta LIFO nebo FIFO
- Implicitní DFS: Backtracking, graf si nemusím držet v paměti
- Depth-limited search
- Iterative deepening
- Best-first search
- Oboustranné vyhledávání

Dijkstra

- Budujeme strom nejkratší cesty z vrcholu do zbytku grafu
- Vždy přidáme nejkratší hranu vedoucí ze stromu



Outline

Prohledávání v grafech

- Kontext umělé inteligence?
- Graf může být obrovský
- Může nám stačit přibližné řešení
- O problému máme informace navíc (heuristiky)

- Agent naviguje v realném prostoru reprezentovaném grafem

Prohledávání v grafech

A*, HPA*, PRA*, inkrementální metody

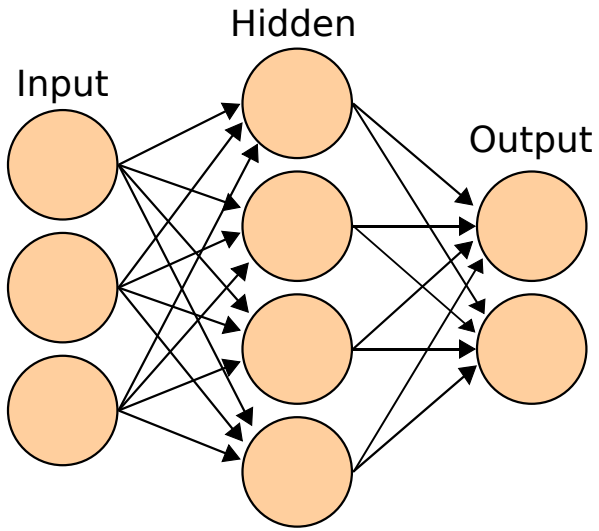
http://bajeluk.matfyz.cz/mff/algs-4-pathf.pdf

Otázky?

Příště UI: Zpracování neurčité informace.
Příště Agenti: Komunikace a znalosti v multiagentních systémech.

Outline

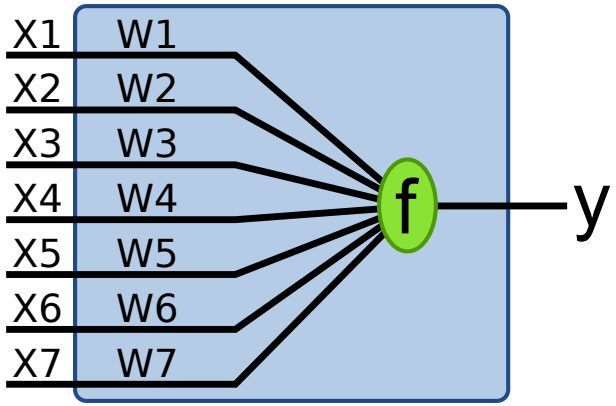
Umělá neuronová síť

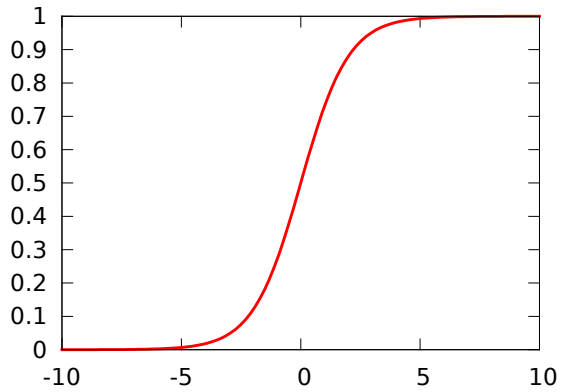


- Umělé neurony (“výpočetní krabičky”) dostávají vstupy (čísla) a na jejich základě generují výstup (číslo)
- Obvykle: Vrstvy striktně oddělené, vstupní vrstva se vstupy zvnějšku, výstupní vrstva s výstupem pro uživatele, skryté vrstvy vyhodnocují různé charakteristiky vstupů
- Dnes: Více vrstev neuronů, jak je učit?

brmlab hackerspace prague

Připomenutí: Umělý neuron

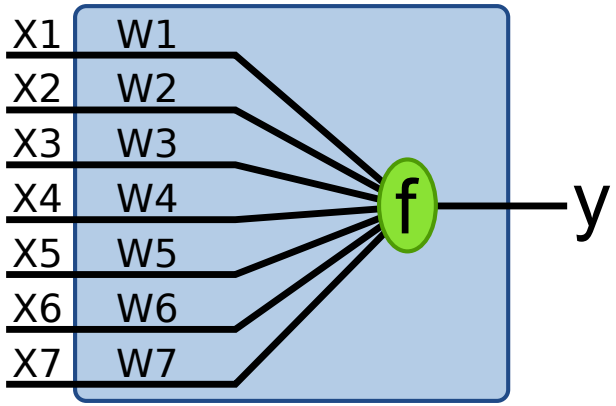


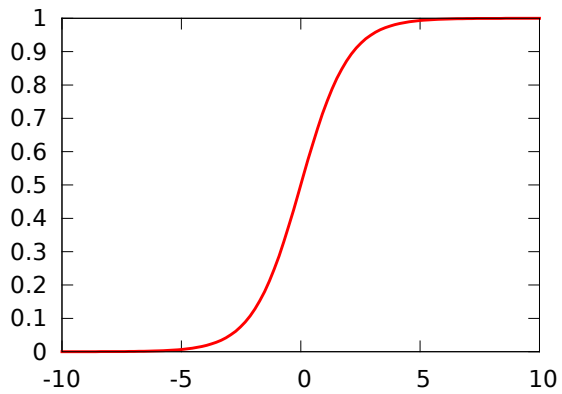


- n vstupů a práh $\Leftarrow$ výstup
- Lineární kombinace — vstupy mají různé váhy, vynásobíme, sečteme a otestujeme
- Výstup je “skoro” 1/0: sigmoida

brmlab hackerspace prague

Připomenutí: Umělý neuron





- n vstupů a práh $\Leftarrow$ výstup
- Lineární kombinace — vstupy mají různé váhy, vynásobíme, sečteme a otestujeme
- Výstup je "skoro" 1/0: sigmoida

- Vstup $\vec{x}$, váhový vektor $\vec{w}$, práh 0
- $\xi = \sum_{i=0}^{n} w_i \cdot x_i$
- Výstupní funkce: $y = f(\xi) = \frac{1}{1+e^{-\lambda\xi}}$

brmlab hackerspace prague

Vícevrstvá NN

- m vrstev, v každé n_m neuronů, propojení vždy pouze $n_i \rightarrow n_{i+1}$
- Inicializace vstupů v první vrstvě
- Iterativní posílání výstupů z jedné vrstvy do vstupů další vrstvy
- Výstup poslední vrstvy je výstup celé sítě

Vícevrstvá NN

- m vrstev, v každé n_m neuronů, propojení vždy pouze $n_i \rightarrow n_{i+1}$
- Inicializace vstupů v první vrstvě
- Iterativní posílání výstupů z jedné vrstvy do vstupů další vrstvy
- Výstup poslední vrstvy je výstup celé sítě

- Víme, kterým vstupům mají odpovídat které výstupy
- Vstupy a výstupy spolu však souvisejí pouze nepřímo
- Jak odvodit konkretní váhy spojů v síti?

Učení vícevrstvé NN

- Algoritmus zpětného šíření
  (existují alternativy, příliš se nepoužívají)
- Myšlenka: Závislosti mezi vstupy a výstupy dokážeme přiměřeně matematicky popsat
- Chceme upravit váhy podle *chyby*, kterou propagovaly; větší váha nese větší chybu

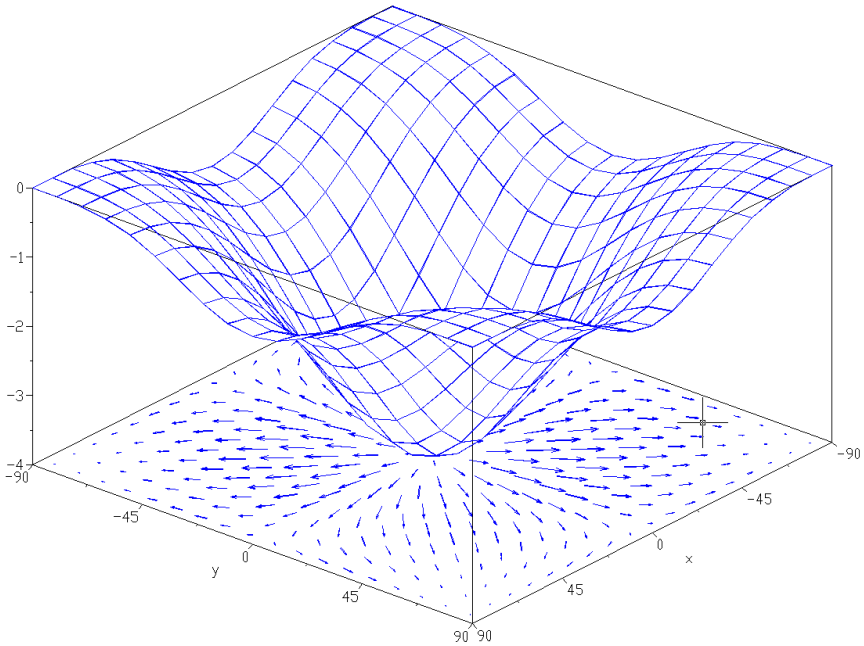
- Iterujeme učení podle vstupních množin:
  - Zjistíme chybu výstupu
  - Spočítáme *gradient* chyby podle vah jednotlivých spojů
  - Chybu se pokusíme zredukovat posunutím vah proti gradientu
  - Chybu "zpětně šíříme" do předchozí vrstvy a opakujeme

Derivace a gradient

Derivace:

Non-free artwork.

Gradient:



Učení poslední vrstvy

- Chybová funkce: $E = \frac{1}{2} \sum_p \sum_j (y_{p,j} - d_{p,j})^2$
  Chceme její hodnotu *minimalizovat*
- Chyba se mění podle váh neuronů (na těch závisí $y_{p,j}$)
  Posuneme se *proti* této změně: $\Delta_E w_{i,j} = -\frac{\partial E}{\partial w_{i,j}}$

Učení poslední vrstvy

- Chybová funkce: $E = \frac{1}{2} \sum_p \sum_j (y_{p,j} - d_{p,j})^2$
  Chceme její hodnotu *minimalizovat*
- Chyba se mění podle váh neuronů (na těch závisí $y_{p,j}$)
  Posuneme se *proti* této změně: $\Delta_E w_{i,j} = -\frac{\partial E}{\partial w_{i,j}}$

$$\xi = \sum_k w_k \cdot x_k \qquad y = f(\xi)$$

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial \xi_j} \cdot \frac{\partial \xi_j}{\partial w_{i,j}} = (y_j - d_j) \cdot f'(\xi_j) \cdot y_i = \delta_i \cdot y_i$$

Učení poslední vrstvy

- Chybová funkce: $E = \frac{1}{2} \sum_p \sum_j (y_{p,j} - d_{p,j})^2$
  Chceme její hodnotu *minimalizovat*
- Chyba se mění podle váh neuronů (na těch závisí $y_{p,j}$)
  Posuneme se *proti* této změně: $\Delta_E w_{i,j} = -\frac{\partial E}{\partial w_{i,j}}$

$$\xi = \sum_k w_k \cdot x_k \qquad y = f(\xi)$$

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial \xi_j} \cdot \frac{\partial \xi_j}{\partial w_{i,j}} = (y_j - d_j) \cdot f'(\xi_j) \cdot y_i = \delta_i \cdot y_i$$

Derivace sigmoidy $f'(\xi) = \left(\frac{1}{1+e^{-\lambda\xi}}\right)' = \lambda \cdot f(\xi) \cdot (1 - f(\xi))$

brmlab hackerspace prague

Učení vnitřních vrstev

Podobná myšlenka jako poslední vrstva,
jen využíváme informaci o chybě z minulé vrstvy.

Otázky?

Příště: Statistické zpracování dat, PCA.

Outline

Rekapitulace

- Rekapitulace: Nezajímá nás, jak rychle to poběží, ale jestli to někdy doběhne.
- Zkoumáme výpočetní *možnosti* algoritmů.

- Pro modelování výpočetních mezí potřebujeme matematický popis programů.
- Primitivně rekurzivní, obecně rekurzivní a částečně rekurzivní funkce.

Rekurzivní funkce

- Funkce: $o(x) = 0 \, \forall x$, $s(x) = x + 1 \, \forall x$, $I_n^j(x_1, \ldots, x_n) = x_j$

Rekurzivní funkce

- Funkce: $o(x) = 0 \, \forall x$, $s(x) = x + 1 \, \forall x$, $I_n^j(x_1, \ldots, x_n) = x_j$
- Operátor substituce: $S_n^m(f, g_1, \ldots, g_m) = h$,
  $h(x_1, \ldots, x_n) \simeq f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$
- Operátor prim. rekurze: $R_n(f, g) = h$,
  $h(0, x_2 \ldots, x_n) \simeq f(x_2, \ldots, x_n)$,
  $h(i + 1, x_1, \ldots, x_n) \simeq g(i, h(i, x_2, \ldots, x_n), x_2, \ldots, x_n)$
- Operátor minimalizace: $M_n(f) = h$,
  $h(x_1, \ldots, x_n) = z \Leftrightarrow f(x_1, \ldots, x_n, z) = 0$ a z je nejmenší

Rekurzivní funkce

- Funkce: $o(x) = 0 \, \forall x$, $s(x) = x + 1 \, \forall x$, $I_n^j(x_1, \ldots, x_n) = x_j$
- Operátor substituce: $S_n^m(f, g_1, \ldots, g_m) = h$,
  $h(x_1, \ldots, x_n) \simeq f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$
- Operátor prim. rekurze: $R_n(f, g) = h$,
  $h(0, x_2 \ldots, x_n) \simeq f(x_2, \ldots, x_n)$,
  $h(i + 1, x_1, \ldots, x_n) \simeq g(i, h(i, x_2, \ldots, x_n), x_2, \ldots, x_n)$
- Operátor minimalizace: $M_n(f) = h$,
  $h(x_1, \ldots, x_n) = z \Leftrightarrow f(x_1, \ldots, x_n, z) = 0$ a z je nejmenší
- **Primitivně rekurzivní funkce:** Bez operátoru minimalizace.
- **Obecně rekurzivní funkce:** Na každém vstupu doběhnou.
- **Částečně rekurzivní funkce:** Mohou a nemusí doběhnout.

brmlab hackerspace prague

Druhy rekurzivních funkcí

- **Primitivně rekurzivní funkce:** Bez operátoru minimalizace.
- **Obecně rekurzivní funkce:** S minimalizací, ale každém vstupu doběhnou.
- **Částečně rekurzivní funkce:** Mohou a nemusí doběhnout.

Druhy rekurzivních funkcí

- **Primitivně rekurzivní funkce:** Bez operátoru minimalizace.
- **Obecně rekurzivní funkce:** S minimalizací, ale každém vstupu doběhnou.
- **Částečně rekurzivní funkce:** Mohou a nemusí doběhnout.

- $PRF \subset ORF \subset CRF$
- Neostré inkluze celkem zřejmé
- ČRF, která není ORF — třeba minimalizace funkce s
- ORF, která není PRF — třeba univerzální funkce pro PRF

Druhy rekurzivních funkcí

- **Primitivně rekurzivní funkce:** Bez operátoru minimalizace.
- **Obecně rekurzivní funkce:** S minimalizací, ale každém vstupu doběhnou.
- **Částečně rekurzivní funkce:** Mohou a nemusí doběhnout.

- $PRF \subset ORF \subset CRF$
- Neostré inkluze celkem zřejmé
- ČRF, která není ORF — třeba minimalizace funkce s
- ORF, která není PRF — třeba univerzální funkce pro PRF

- Každé ČRF můžu přiřadit číslo — na základě jejího odvození

Kleenova věta

- **Univerzální TS:** "Emulátor" — vrátí výstup libovolného TS na základě jeho popisu a vstupu
- **Univerzální RF $\Psi(e, \vec{x})$:** "Emulátor" — vrátí výstup libovolné RF na základě jejího čísla e a parametrů $\vec{x}$

Kleenova věta

- **Univerzální TS:** "Emulátor" — vrátí výstup libovolného TS na základě jeho popisu a vstupu
- **Univerzální RF $\Psi(e, \vec{x})$:** "Emulátor" — vrátí výstup libovolné RF na základě jejího čísla e a parametrů $\vec{x}$

- **s-m-n věta:** Curryfikace RF. $\Psi(e, \vec{z}, \vec{x}) = \Psi(s(e, \vec{z}), \vec{x})$

Kleenova věta

- **Univerzální TS:** "Emulátor" — vrátí výstup libovolného TS na základě jeho popisu a vstupu
- **Univerzální RF $\Psi(e, \vec{x})$:** "Emulátor" — vrátí výstup libovolné RF na základě jejího čísla e a parametrů $\vec{x}$

- **s-m-n věta:** Curryfikace RF. $\Psi(e, \vec{z}, \vec{x}) = \Psi(s(e, \vec{z}), \vec{x})$

- **Turingův predikát $T(e, \vec{x}, y)$:** Propojení s logikou, je pravdivý, pokud program e s parametry $\vec{x}$ vrátí y.
- V logice lze přes T popsat ČRP, ORP, PRP (. . . predikáty).

Univerzální funkce

- Máme univerzální ČŘF emulující jakouliv ČRF.
- Umíme udělat univerzální PRF pro všechny PRF?

Univerzální funkce

- Máme univerzální ČŘF emulující jakouliv ČRF.
- Umíme udělat univerzální PRF pro všechny PRF?
- Neumíme. Mějme $f(e, x)$ univerzální PRF. Zavedeme PRF $g(x) = f(x, x) + 1$ s číslem e: $f(e, x) = g(x)$. Ale tedy $f(e, e) = g(e) = f(e, e) + 1$, což je spor.

Univerzální funkce

- Máme univerzální ČŘF emulující jakouliv ČRF.
- Umíme udělat univerzální PRF pro všechny PRF?
- Neumíme. Mějme $f(e, x)$ univerzální PRF. Zavedeme PRF $g(x) = f(x, x) + 1$ s číslem e: $f(e, x) = g(x)$. Ale tedy $f(e, e) = g(e) = f(e, e) + 1$, což je spor.

- Univerzální RF pro ORF je ČRF.

Množiny popsané funkcemi

- Množiny obvykle definujeme *predikáty* — např. $\{x \mid x \in N, 5 < x < 10\}$
- Predikát koresponduje k funkci
- Tedy **primitivně rekurzivní**, **rekurzivní** a **částečně rekurzivní** množiny, RF rozhoduje o prvku, patří-li do množiny
- M je rekurzivní, právě když je ona i doplněk rekurzivně spočetné

Otázky?

Příště: Věty o rekurzi a jejich aplikace.

Děkuji vám

pasky@ucw.cz

Příště: Unlecture.
Popříště: Pravděpodobnost. Základy, použití v AI, pravděpodobnostní algoritmy, "pravděpodobnostní datové struktury" (hashe). Ale kdy?

brmlab hackerspace prague