

Tady nic není.

Něco tady bude.

# Invertované Integerové Indexy

---

Domácí úkol č.2

# Intro

---

**Vstup:** 1 celý internet webových stránek

- klíčová slova v dokumentech,  $m : n$
- obsah internetu se občas mění
- klíčových slov jsou miliony
- dokumentů jsou miliardy

**Úkol:**

**Vstup:** 1 celý internet webových stránek

- klíčová slova v dokumentech,  $m : n$
- obsah internetu se občas mění
- klíčových slov jsou miliony
- dokumentů jsou miliardy

**Úkol:** Najděte stránky, na kterých někdo pomocí Lorenzových rovnic spočítal změnu náboje a trajektorie krávy obsahující 10% oceli při středně rychlém průletu magnetickým polem země.

**Uživatel očekává výsledky za 5 milisekund.**

V jaké datové struktuře ukládat data pro hledání tak, aby bylo možné rychle získat relevantní výsledky?

**Upravený úkol:** Máme množinu dokumentů  $D$ , množinu klíčových slov  $K$ , a dotaz  $Q \subseteq K$ . Najděte výsledek  $R \subseteq D$  kde platí, že

$$x \in R \iff Q \subseteq K_x.$$

V jaké datové struktuře ukládat data pro hledání tak, aby bylo možné rychle získat relevantní výsledky?

**Upravený úkol:** Máme množinu dokumentů  $D$ , množinu klíčových slov  $K$ , a dotaz  $Q \subseteq K$ . Najděte výsledek  $R \subseteq D$  kde platí, že

$$x \in R \iff Q \subseteq K_x.$$

Tradiční indexování pomocí stromů selže kvůli dimenzionalitě ( $d \geq 10^9$ ).

Řešení:



V jaké datové struktuře ukládat data pro hledání tak, aby bylo možné rychle získat relevantní výsledky?

**Upravený úkol:** Máme množinu dokumentů  $D$ , množinu klíčových slov  $K$ , a dotaz  $Q \subseteq K$ . Najděte výsledek  $R \subseteq D$  kde platí, že

$$x \in R \iff Q \subseteq K_x.$$

Tradiční indexování pomocí stromů selže kvůli dimenzionalitě ( $d \geq 10^9$ ).

Řešení: Inverze — místo dokumentů indexujeme klíčová slova, pro každé klíčové slovo si uložíme obyčejný seznam dokumentů, ve kterých se vyskytuje.

Dokumenty budeme číslovat.

**rovnice** 1, 2, 3, 15, 17, 35, 66, 89, 103

**Lorenz** 1, 15, 89

**java** 2, 5, 34, 37, 63, 86, 103, 110

**trajektorie** 2, 3, 4, 15, 24, 66, 91

**kráva** 6, 15, 53, 55, 58, 69

**koťata** 1, 2, 3, 4, 5, 6, 7, 8, ...

**magnet** 7, 15, 22, 24, 50, 63, 72

Dokumenty budeme číslovat.

**rovnice** 1, 2, 3, 15, 17, 35, 66, 89, 103

**Lorenz** 1, 15, 89

**trajektorie** 2, 3, 4, 15, 24, 66, 91

**kráva** 6, 15, 53, 55, 58, 69

**magnet** 7, 15, 22, 24, 50, 63, 72

Dokumenty budeme číslovat.

**rovnice** 1, 2, 3, 15, 17, 35, 66, 89, 103

**Lorenz** 1, 15, 89

**trajektorie** 2, 3, 4, 15, 24, 66, 91

**kráva** 6, 15, 53, 55, 58, 69

**magnet** 7, 15, 22, 24, 50, 63, 72

Dokumenty budeme číslovat.

**rovnice** 1, 2, 3, 15, 17, 35, 66, 89, 103

**Lorenz** 1, 15, 89

**trajektorie** 2, 3, 4, 15, 24, 66, 91

**kráva** 6, 15, 53, 55, 58, 69

**magnet** 7, 15, 22, 24, 50, 63, 72

Dokumenty budeme číslovat.

**rovnice** 1, 2, 3, 15, 17, 35, 66, 89, 103

**Lorenz** 1, 15, 89

**trajektorie** 2, 3, 4, 15, 24, 66, 91

**kráva** 6, 15, 53, 55, 58, 69

**magnet** 7, 15, 22, 24, 50, 63, 72

Dokumenty budeme číslovat.

**rovnice** 1, 2, 3, 15, 17, 35, 66, 89, 103

**Lorenz** 1, 15, 89

**trajektorie** 2, 3, 4, 15, 24, 66, 91

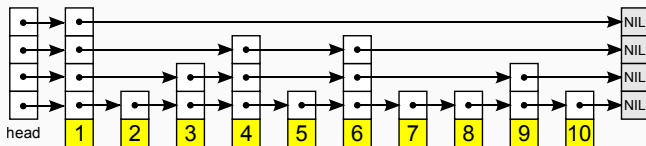
**kráva** 6, 15, 53, 55, 58, 69

**magnet** 7, 15, 22, 24, 50, 63, 72

# Posting list

Seznamu dokumentů, ve kterých se klíčové slovo vyskytuje, se říká *posting list*.

- Tradičně bývá setříděný, aby současné procházení dokumentů bylo rychlé.
- Triviální implementace: singly-linked list
- Lepší implementace: array, skip-list





- Méně I/O: delta encoding
  - 100000001, 100000002, 100000003, ...
  - 100000001, +1, +1, ...  
možné kódování: UTF-8
- Mírná pomoc CPU:
  - arraye
  - bloky dat ve skip-listech

Uloženou formu invertovaného indexu nejde moc jednoduše updatovat.

Uloženou formu invertovaného indexu nejde moc jednoduše updatovat.

Místo toho se vedle vyrobí druhý index.

- implementace se ptá obou

Uloženou formu invertovaného indexu nejde moc jednoduše updatovat.

Místo toho se vedle vyrobí druhý index.

- implementace se ptá obou (všech)
- když je indexů hodně, implementace některé sleje do jednoho (jde provést v  $\mathcal{O}(n)$ ).

Uloženou formu invertovaného indexu nejde moc jednoduše updatovat.

Místo toho se vedle vyrobí druhý index.

- implementace se ptá obou (všech)
- když je indexů hodně, implementace některé sleje do jednoho (jde provést v  $\mathcal{O}(n)$ ).

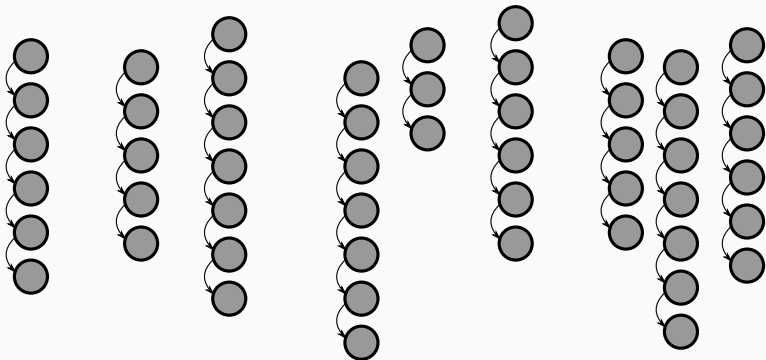
...tohle nebudeme řešit.

Typické problémové situace:

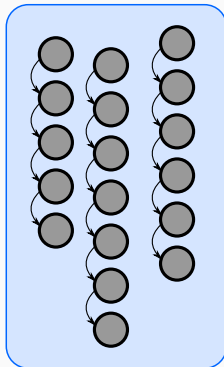
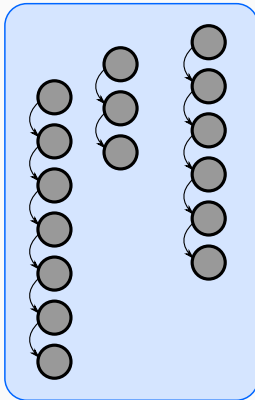
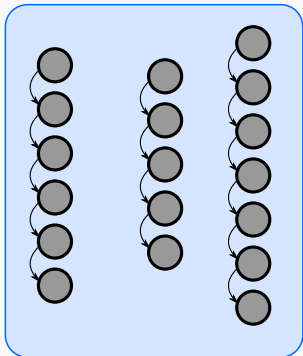
- dotaz je hodně složitý (stovky klíčových slov)
- indexy jsou husté, takže skipování pomůže jen trochu
- indexy jsou na více počítačích

Procházení seznamů je paralelizovatelné!

# Paralelní prohledávání

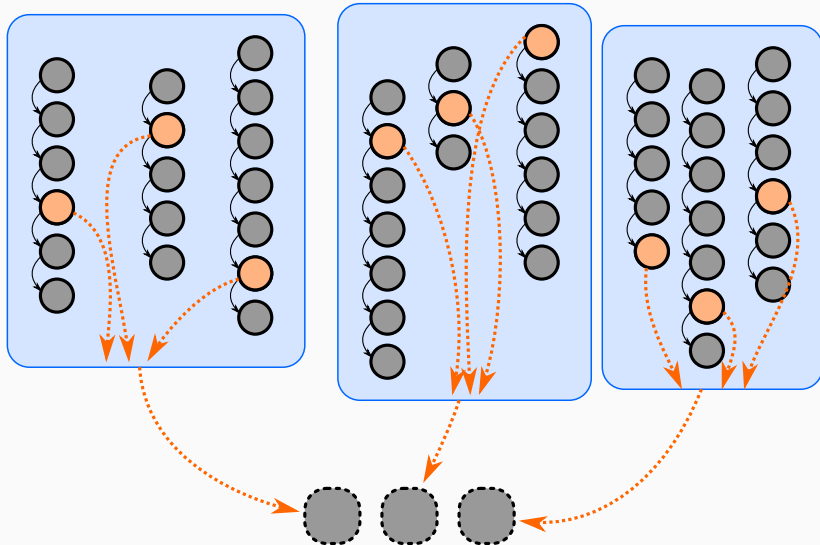


# Paralelní prohledávání

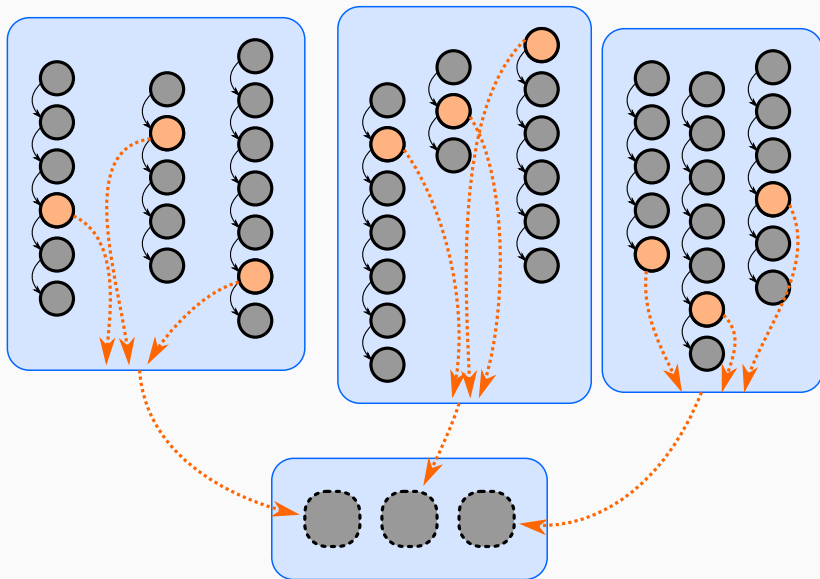




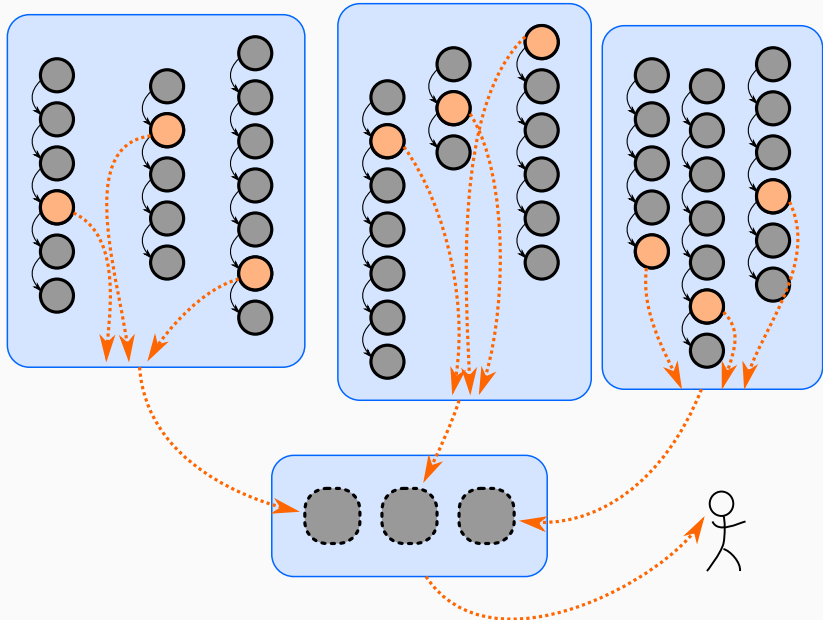
# Paralelní prohledávání



# Paralelní prohledávání



# Paralelní prohlédávání



## Zadání

---

Implementujte invertovaný index pro integerové objekty a jejich featury:

- číslo objektu je `uint64_t`
- číslo featury je `uint64_t`

```
template<typename Trunc, typename FeatObjLists>  
void create (Trunc&& trunc, FeatObjLists&& fs);
```

- `trunc` = rozhraní k úložišti, viz. `man 2 truncate`
- `trunc(123)` oseká/zvětší soubor s indexem na násobek velikosti `uint64_t`, vrátí (pravděpodobně) změněný pointer (`uint64_t*`) na začátek nové oblasti

```
template<typename Trunc, typename FeatObjLists>  
void create (Trunc&& trunc, FeatObjLists&& fs);
```

- `trunc` = rozhraní k úložišti, viz. man 2 `truncate`
- `trunc(123)` oseká/zvětší soubor s indexem na násobek velikosti `uint64_t`, vrátí (pravděpodobně) změněný pointer (`uint64_t*`) na začátek nové oblasti
- `fs.size()` vrátí počet všech možných featur (čísla:  $0 \cdots (n - 1)$ )
- `fs[n]` vrátí dopředně iterovatelný objekt popisující čísla objektů s featurou  $n$  (můžete ho vyrobit a projít víckrát)

Použití a požadavky:

- nespotřebovat moc paměti  
soubory jsou `mmap()`ované, takže moc paměti ve skutečnosti nezabírají
  - načtení všech dat do nemapované paměti najednou selže



Použití a požadavky:

- nespotřebovat moc paměti  
soubory jsou `mmap()`ované, takže moc paměti ve skutečnosti nezabírají
  - načtení všech dat do nemapované paměti najednou selže
- create by mělo zapsat kompaktní formu dat připravenou k prohledávání (času na to má relativně dost!).

Doporučená struktura souboru:

- počet featur
- indexu indexů s offsety začátků
- indexy

```
template<class Fs, class OutFn>  
void search (const uint64_t*index, size_t size, Fs&&  
fs, OutFn&& callback)
```

- fs je iterovatelný kontejner featur v dotazu
- callback je funktor na odevzdávání výsledku, kompatibilní s `void (*callback)(uint64_t)`
  - zavolejte ho na všechny čísla objektů vyhovujících query (od nejmenšího ID)

- Paralelizujte prohledávání!

- Paralelizujte prohledávání!

Možnosti implementace:

- worker pool/thread batch
- buffery/streamy

- Paralelizujte prohledávání!

Možnosti implementace:

- worker pool/thread batch
- buffery/streamy
- Použijte rozumná primitiva
  - mutexy — zamykání struktur
  - condition variables — čekání na událost
  - některé poměrně primitivní kontejnery není potřeba za určitých podmínek zamykat

- Paralelizujte prohledávání!  
Možnosti implementace:
  - worker pool/thread batch
  - buffery/streamy
- Použijte rozumná primitiva
  - mutexy — zamykání struktur
  - condition variables — čekání na událost
  - některé poměrně primitivní kontejnery není potřeba za určitých podmínek zamykat
- Tradiční výzva: lockless

- Paralelizujte prohledávání!  
Možnosti implementace:
  - worker pool/thread batch
  - buffery/streamy
- Použijte rozumná primitiva
  - mutexy — zamykání struktur
  - condition variables — čekání na událost
  - některé poměrně primitivní kontejnery není potřeba za určitých podmínek zamykat
- Tradiční výzva: lockless  
(lockless neznamená, že mutexy 1:1 nahradíte CAS spinlockem)

- Indexování může nějakou dobu trvat, manipuluje s obrovským objemem dat.
- Výstupem indexování je kompaktní, serializovatelná reprezentace indexů.
- Prohledávání musí být co nejrychlejší (tj. paralelní), manipuluje s velkým objemem dat.



**Konec**

---

# Co se bude hodnotit?

Must-have:

- Program neumře, nespadne, necrashne, jde skompilovat
- Program vrací rozumné výsledky (ideálně správné)
- Prohledávání využívá víc CPU

Optimalizace (použijte jako bonus):

- Přeskakování v seznamech (např. skip-pointery)
- Kompaktnější reprezentace jednotlivých integerů

Tradiční metriky:

- **méně přehlednějšího kódu**
- formátování, komentáře, rozumné názvy
- efektivita
- přenositelnost

**Q&A**