Prioritní fronta, halda

Petr Ryšavý

17. září 2019

Katedra počítačů, FEL, ČVUT

HALDA

Prioritní fronta (anglicky *priority queue*)

- Kontejner na objekty, které jsou identifikovány klíčem
  - Osoby, odkazy v internetu, události s časy, atd.
- `insert`- přidej nový objekt do prioritní fronty
- `extract-min`- odeber z haldy objekt s nejmenším klíčem (pflichta se řeší libovolně)[1]

[1] Někdy prioritní fronta místo `extract-min` nabízí operaci `extract-max`. Nikdy ale obě zároveň.

Halda (anglicky *heap*)

- Podporuje obě operace v $\mathcal{O}(\log n)$.
- Nejčastější implementace prioritní fronty.
- Navíc operace
  - `heapify`- inicializace haldy v lineárním čase
  - `delete`- odstranění libovolného prvku z haldy $\mathcal{O}(\log n)$.

Užití haldy

- Pokud objevíme v algoritmu opakované počítání minima.
  - Např. rozdíl mezi *selection-sort* a *heap-sort*.

Užití haldy

- Pokud objevíme v algoritmu opakované počítání minima.
  - Např. rozdíl mezi *selection-sort* a *heap-sort*.
- Plánovač úloh
  - Například události s časovým klíčem, např. ve hře.

Užití haldy

- Pokud objevíme v algoritmu opakované počítání minima.
  - Např. rozdíl mezi *selection-sort* a *heap-sort*.
- Plánovač úloh
  - Například události s časovým klíčem, např. ve hře.
- *Median Maintenance*
  - sekvence $x_1, x_2, \ldots, x_n$, dodáváme prvky jeden po druhém,
  - v každém kroku je třeba říci medián z čísel $x_1, x_2, \ldots, x_i$,
  - podmínkou je, že to stíháme v $\mathcal{O}(\log n)$.

Užití haldy

- Pokud objevíme v algoritmu opakované počítání minima.
  - Např. rozdíl mezi *selection-sort* a *heap-sort*.
- Plánovač úloh
  - Například události s časovým klíčem, např. ve hře.
- *Median Maintenance*
  - sekvence $x_1, x_2, \ldots, x_n$, dodáváme prvky jeden po druhém,
  - v každém kroku je třeba říci medián z čísel $x_1, x_2, \ldots, x_i$,
  - podmínkou je, že to stíháme v $\mathcal{O}(\log n)$.
- zrychlení Dijkstrova algoritmu
  - Naivně $\mathcal{O}(mn)$, s haldou $\mathcal{O}(m \log n)$.

Programování - halda uživatelsky

- Proveďte úkoly 12.-15.

Přestávka

IMPLEMENTACE HALDY

Definice haldy

2 pohledy - jako strom a jako pole

- Halda je kořenový, binární strom, který je co nejvíce vyvážený.
- Poslední úroveň je zaplňována zleva.
- Je splněná vlastnost haldy:

$$\forall x : \mathrm{key}[x] \leq \text{ klíče potomků uzlu } x.$$

Důsledek

V každé haldě je minimální prvek uložen v kořeni.

Implementace v poli

- Očíslujme uzly podle úrovní
- Efektivnější na paměť (nepotřebujeme žádné ukazatele navíc)
- Efektivní vzorce pro výpočet adres rodiče a potomků

Implementace v poli

- Očíslujme uzly podle úrovní
- Efektivnější na paměť (nepotřebujeme žádné ukazatele navíc)
- Efektivní vzorce pro výpočet adres rodiče a potomků

$$\text{parent}(i) = \begin{cases} \frac{i}{2} & i \text{ sudé}, \\ \lfloor \frac{i}{2} \rfloor & i \text{ liché}, \end{cases}$$

$$\text{children}(i) = \{2i, 2i+1\}$$

`insert` a probublávání nahoru

- Vložíme nový klíč na konec haldy.
- Probubláváme ho nahoru tak dlouho, odkud není vlastnost haldy splněna.
- Čas běhu je $\log(n)$.

`extract-min` a probublávání nahoru

- Odebereme kořen.
- Nahradíme ho posledním uzlem.
- Probubláváme nový kořen dolů.
- Vždy prohazujeme s menším z potomků.

Programování - vlastní implementace haldy

- Proveďte úkol 16. Pokud neprogramujete v Javě, ale v C++, Pythonu, zkuste dodržet strukturu třídy.
- Vlastní implementaci haldy budeme potřebovat zítra v Dijkstrově algoritmu, dejte si tedy záležet.
- Popis a implementaci lze opět nalézt na `http://algs4.cs.princeton.edu/24pq/`.

Děkuji za pozornost.

Čas na otázky!