

Hashing

Neven Miculinić Marin Smiljanić

Fakultet elektrotehnike i računarstva

17. siječanj, 2014

Kratki uvod i malo teorije

Što je hashing ?

Hashing možemo definirati kao mapiranje nekih većih objekata u manje kako bismo ih mogli lakše uspoređivati.

Kratki uvod i malo teorije

Što je hashing ?

Hashing možemo definirati kao mapiranje nekih većih objekata u manje kako bismo ih mogli lakše uspoređivati.

Idealno: $hash : D \rightarrow K$ bijekcija

Stvarno: $hash : D \rightarrow K$ surjekcija (kolizije!)

Kratki uvod i malo teorije

Što je hashing ?

Hashing možemo definirati kao mapiranje nekih većih objekata u manje kako bismo ih mogli lakše uspoređivati.

Idealno: $hash : D \rightarrow K$ bijekcija

Stvarno: $hash : D \rightarrow K$ surjekcija (kolizije!)

Hash funkcije uvodimo kako bi na “računalu pristupačniji” način predstavili podatke.

Jedan jednostavan primjer

Trodimenzionalni niz cijelih brojeva čija je svaka dimenzija reda veličine 10^6 .

Jedan jednostavan primjer

Trodimenzionalni niz cijelih brojeva čija je svaka dimenzija reda veličine 10^6 .

Takav niz ne možemo spremiti u memoriju pa je potrebno naći drugo rješenje.

Jedan jednostavan primjer

Trodimenzionalni niz cijelih brojeva čija je svaka dimenzija reda veličine 10^6 .

Takav niz ne možemo spremiti u memoriju pa je potrebno naći drugo rješenje.

Hash funkcija oblika $h : A \times A \times A \rightarrow B$

Jedan jednostavan primjer

Trodimenzionalni niz cijelih brojeva čija je svaka dimenzija reda veličine 10^6 .

Takav niz ne možemo spremiti u memoriju pa je potrebno naći drugo rješenje.

Hash funkcija oblika $h : A \times A \times A \rightarrow B$

npr. $h(x, y, z) = (x \cdot a^2 + y \cdot a + z) \text{ mod } p$

pri čemu možemo za p uzeti 1000003, a za a neki nasumični broj iz $[0, p - 1]$

Pojava kolizije

Chaining metoda rješavanja kolizije

Prepostavimo da je $N[h(x, y, z)]$ lista četvorki (x, y, z, v) , pričemu (x, y, z) predstavljaju ključ, a v vrijednost.

Pojava kolizije

Chaining metoda rješavanja kolizije

Prepostavimo da je $N[h(x, y, z)]$ lista četvorki (x, y, z, v) , pri čemu (x, y, z) predstavljaju ključ, a v vrijednost.

Ako želimo elementu (x, y, z) dodijeliti neku vrijednost v prvo ćemo ga potražiti u listi koja mu je pridružena *hash* funkcijom.

Pojava kolizije

Chaining metoda rješavanja kolizije

Prepostavimo da je $N[h(x, y, z)]$ lista četvorki (x, y, z, v) , pri čemu (x, y, z) predstavljaju ključ, a v vrijednost.

Ako želimo elementu (x, y, z) dodijeliti neku vrijednost v prvo ćemo ga potražiti u listi koja mu je pridružena *hash* funkcijom.

Ako ga pronađemo samo ćemo mu promijeniti pridruženu vrijednost, inače ćemo ga dodati na kraj liste.

Analiza hash funkcije

Kako znamo da će naša funkcija biti homogena ?

Promatrajmo dvije trojke, (x_0, y_0, z_0) i (x_1, y_1, z_1) .

Analiza hash funkcije

Kako znamo da će naša funkcija biti homogena ?

Promatrajmo dvije trojke, (x_0, y_0, z_0) i (x_1, y_1, z_1) .

$$h(x_0, y_0, z_0) = h(x_1, y_1, z_1)$$

Analiza hash funkcije

Kako znamo da će naša funkcija biti homogena ?

Promatrajmo dvije trojke, (x_0, y_0, z_0) i (x_1, y_1, z_1) .

$$h(x_0, y_0, z_0) = h(x_1, y_1, z_1)$$

$$(x_0 - x_1) \cdot a^2 + (y_0 - y_1) \cdot a + (z_0 - z_1) \equiv 0 \pmod{p}$$

Analiza hash funkcije

Kako znamo da će naša funkcija biti homogena ?

Promatrajmo dvije trojke, (x_0, y_0, z_0) i (x_1, y_1, z_1) .

$$h(x_0, y_0, z_0) = h(x_1, y_1, z_1)$$

$$(x_0 - x_1) \cdot a^2 + (y_0 - y_1) \cdot a + (z_0 - z_1) \equiv 0 \pmod{p}$$

Polinom je maksimalno drugog stupnja pa tako ima maksimalno dvije različite nultočke (vrijedi samo za prost broj p).

Analiza hash funkcije

Kako znamo da će naša funkcija biti homogena ?

Promatrajmo dvije trojke, (x_0, y_0, z_0) i (x_1, y_1, z_1) .

$$h(x_0, y_0, z_0) = h(x_1, y_1, z_1)$$

$$(x_0 - x_1) \cdot a^2 + (y_0 - y_1) \cdot a + (z_0 - z_1) \equiv 0 \pmod{p}$$

Polinom je maksimalno drugog stupnja pa tako ima maksimalno dvije različite nultočke (vrijedi samo za prost broj p).

Vjerojatnost da je nasumični a nultočka ovog polinoma je $2/p$, što za gore navedeni p iznosi $2 \cdot 10^{-6}$.

Analiza hash funkcije

Birthday Paradox

Kolika je šansa da postoji kolizija među bilo koja dva para?

Analiza hash funkcije

Birthday Paradox

Kolika je šansa da postoji kolizija među bilo koja dva para?

$$p(n) = 1 - \frac{p!}{p^n(p-n)!}$$

Analiza hash funkcije

Birthday Paradox

Kolika je šansa da postoji kolizija među bilo koja dva para?

$$p(n) = 1 - \frac{p!}{p^n(p-n)!}$$

$$p(n) \approx 1 - e^{\frac{n(n-1)}{2p}}$$

Analiza hash funkcije

Birthday Paradox

Kolika je šansa da postoji kolizija među bilo koja dva para?

$$p(n) = 1 - \frac{p!}{p^n(p-n)!}$$

$$p(n) \approx 1 - e^{\frac{n(n-1)}{2p}}$$

Ugrubo za $n = \sqrt{p} \rightarrow p(n) \approx 50\%$

Prethodni primjer se jednostavno poopći i na razne druge tipove indeksa i time dobivamo podatkovnu strukturu *hash tablice*.

Prethodni primjer se jednostavno poopći i na razne druge tipove indeksa i time dobivamo podatkovnu strukturu *hash tablice*.

UBACI(X) : $S = S \cup X$

IZBACI(X) : $S = S / X$

PRONADI(k) : $X \in S; t.d.X.key = k$, inace $X = NULL$

Hash tablice

Prethodni primjer se jednostavno poopći i na razne druge tipove indeksa i time dobivamo podatkovnu strukturu *hash tablice*.

UBACI(X) : $S = S \cup X$

IZBACI(X) : $S = S / X$

PRONADI(k) : $X \in S$; t.d. $X.key = k$, inace $X = NULL$

Load factor $\alpha = \text{broj elemenata} / \text{broj mjesta}$

Hashing stringova

Stringovi su uvjerljivo najčešće “žrtve” *hashiranja*.

Stringovi su uvjerljivo najčešće “žrtve” *hashiranja*.

Usporedbe stringova znak po znak nisu uvjek dovoljno efikasne.

Hash funkcija za stringove

Hash funkcije koje obično koristimo za stringove također imaju oblik polinoma.

Hash funkcija za stringove

Hash funkcije koje obično koristimo za stringove također imaju oblik polinoma.

Neka je S string s kojim radimo, n njegova duljina i proste brojeve p i q , $q \gg p$.

Hash funkcija za stringove

Hash funkcije koje obično koristimo za stringove također imaju oblik polinoma.

Neka je S string s kojim radimo, n njegova duljina i proste brojeve p i q , $q \gg p$.

Pomoćni niz H definiramo rekurzivno na sljedeći način.

Hash funkcija za stringove

Hash funkcije koje obično koristimo za stringove također imaju oblik polinoma.

Neka je S string s kojim radimo, n njegova duljina i proste brojeve p i q , $q \gg p$.

Pomoćni niz H definiramo rekurzivno na sljedeći način.

Potencije od p prethodno izračunamo radi veće efikasnosti

Hash funkcija za stringove

Hash funkcije koje obično koristimo za stringove također imaju oblik polinoma.

Neka je S string s kojim radimo, n njegova duljina i proste brojeve p i q , $q \gg p$.

Pomoćni niz H definiramo rekurzivno na sljedeći način.

Potencije od p prethodno izračunamo radi veće efikasnosti

$$H_0 \equiv S_0 \pmod{q}$$

$$H_i \equiv p \cdot H_{i-1} + S_i \pmod{q}$$

Hash funkcija za stringove

Hash funkcija za stringove

No što nam predstavlja niz H ?

Hash funkcija za stringove

No što nam predstavlja niz H ?

Jako je sličan Hornerovom algoritmu za evaluiranje polinoma u jednoj točki.

$$H_i = \sum_{j=0}^i S_j \cdot p^{i-j}$$

Kao *hash* vrijednost zadanog stringa podrazumijevamo H_{n-1} , dok ostale članove niza nazivamo *prefiks hashevima*

Izdvajanje hash vrijednosti podstringova stringa S

Označimo sa $S_{i,j}$ podstring stringa S koji se provlači od znaka i do znaka j .

Izdvajanje hash vrijednosti podstringova stringa S

Označimo sa $S_{i,j}$ podstring stringa S koji se provlači od znaka i do znaka j .

Po našoj definiciji *hash* vrijednosti stringa dobivamo sljedeću sumu.

$$h(S_{i,j}) \equiv \sum_{k=i}^j S_k \cdot P_{j-k} \pmod{q}$$

Izdvajanje hash vrijednosti podstringova stringa S

Označimo sa $S_{i,j}$ podstring stringa S koji se provlači od znaka i do znaka j .

Po našoj definiciji *hash* vrijednosti stringa dobivamo sljedeću sumu.

$$h(S_{i,j}) \equiv \sum_{k=i}^j S_k \cdot P_{j-k} \pmod{q}$$

Tu vrijednost možemo dobiti iz već izračunatog niza H .

$$h(S_{i,j}) \equiv H_j - H_{i-1} \cdot p^{j-i+1} \pmod{q}$$

Traženje malog stringa u većem

Prepostavimo da imamo dva stringa S i Z s duljinama redom n i m , i neka je $n < m$, tj. tražimo S u Z .

Traženje malog stringa u većem

Prepostavimo da imamo dva stringa S i Z s duljinama redom n i m , i neka je $n < m$, tj. tražimo S u Z .

Izračunajmo hash vrijednost stringa S , h_s i pomoćne niz H za string Z .

Traženje malog stringa u većem

Prepostavimo da imamo dva stringa S i Z s duljinama redom n i m , i neka je $n < m$, tj. tražimo S u Z .

Izračunajmo hash vrijednost stringa S , h_s i pomoćne niz H za string Z .

Sada se ovo traženje svodi na ispitivanje je li $h(Z_{i,i+n-1})$ jednak h_s za sve indekse i .

Traženje malog stringa u većem

Prepostavimo da imamo dva stringa S i Z s duljinama redom n i m , i neka je $n < m$, tj. tražimo S u Z .

Izračunajmo hash vrijednost stringa S , h_s i pomoćne niz H za string Z .

Sada se ovo traženje svodi na ispitivanje je li $h(Z_{i,i+n-1})$ jednak h_s za sve indekse i .

Sličan princip možemo primijeniti na višedimenzionalne nizove.

Traženje malog stringa u većem Rabin-Karp

Poboljšanje memorejske efikasnosti?

Traženje malog stringa u većem Rabin-Karp

Poboljšanje memorejske efikasnosti?

Rolling hash

Traženje malog stringa u većem Rabin-Karp

Poboljšanje memorijске efikasnosti?

Rolling hash

Izračunamo $h(Z_{0,n-1})$

Traženje malog stringa u većem Rabin-Karp

Poboljšanje memorejske efikasnosti?

Rolling hash

Izračunamo $h(Z_{0,n-1})$

Koristimo rekurziju:

$$h(Z_{i+1,i+n}) \equiv [h(Z_{i,i+n-1}) - Z_i \cdot p^n] \cdot p + Z_{i+n} \pmod{q}$$

Ako umjesto samo jednog pomoćnog niza H konstruiramo i niz H^r kao prefiks hasheva obrnutog stringa.

Ako umjesto samo jednog pomoćnog niza H konstruiramo i niz H^r kao prefiks hasheva obrnutog stringa.

Sada se ispitivanje je li podstring $S[i, j]$ palindrom svodi na ispitivanje jednakosti $h(S_{i, j})$ i $h(S_{n-j-1, n-i-1}^r)$

Ako umjesto samo jednog pomoćnog niza H konstruiramo i niz H^r kao prefiks hasheva obrnutog stringa.

Sada se ispitivanje je li podstring $S[i, j]$ palindrom svodi na ispitivanje jednakosti $h(S_{i, j})$ i $h(S_{n-j-1, n-i-1}^r)$

Periodičnost stringa također je jednostavna za provjeriti. Uzmemo svaki prefiks stringa, pomicemo se za njegovu duljinu i pritom testiramo jednakost.

Pitanja ? Želje ? Pozdravi ?