

Napredni algoritmi i strukture podataka

Merkle stabla, Serijalizacija stabla



Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

- ▶ Formalno gledamo, Merkle stabla uzimaju skup podataka (x_1, \dots, x_n) na ulazu
- ▶ Povratnu vrednost je **Merkle root hash** $h = \text{MHT}(x_1, \dots, x_n)$
- ▶ MHT **collision-resistant hash funkcija**
- ▶ *Hash* funkcija je **collision-resistant hash funkcija** ako je teško pronaći dva ulaza koja *hash*-iraju isti izlaz
- ▶ Formalno, za ulaze a i b , $a \neq b$ ali $H(a) = H(b)$

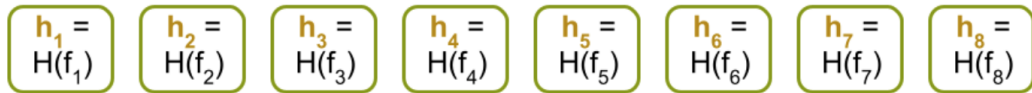
Merkle stablo — formiranje

- ▶ Algoritam za formiranje Merkle stabla je relativno jednostavan
- ▶ Merkle stablo ima **bottom-up** pristup izgradnje, zbog svoje specifičnosti
- ▶ Formiranje stabla počinje od dna tj. konkretizovanih podataka — **data block**
- ▶ Polako idemo do vrha, gradeći **Merkle root** element
- ▶ Prvi element koji gradimo je **list**
- ▶ Svaki podatak propustimo kroz *hash* funkciju, i tako formiramo prvi nivo — **list**

- ▶ Nakon toga, svaka **dva susedna** elementa grade naredni nivo propuštajući njihove zajedničke hash vrednosti kroz hash funkciju
- ▶ Pošto radimo sa binarnim stablima, ako na nekom nivou nemamo odgovarajući čvor, možemo da dodamo *empty* element da bi algoritam mogao da se nastavi
- ▶ Kada propustimo poslednja dva čvora kroz hash funkciju dobijamo **Merkle root** element
- ▶ Time se algoritam za formiranje završava i formirali smo Merkle stablo

Merkle stablo - formiranje, primer

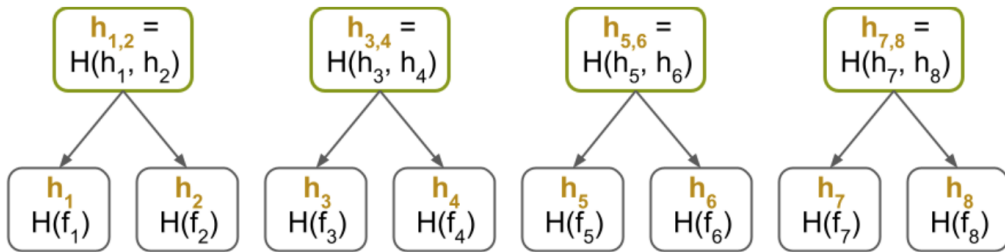
- ▶ Pretpostavimo da imamo 8 blokova podataka (fajlova) $f = (f_1, \dots, f_8)$
- ▶ Svaki podataka f_i propustimo kroz hash funkciju H i dobijamo njegov *hash*
- ▶ Dobijamo hash vrednost za prvi nivo $h_i = H(f_i)$, $h_i = (h_1, \dots, h_8)$
- ▶ H reprezentuje **collision-resistant hash** funkciju



(Decentralized Thoughts, Merkle trees)

Merkle stablo - formiranje, primer

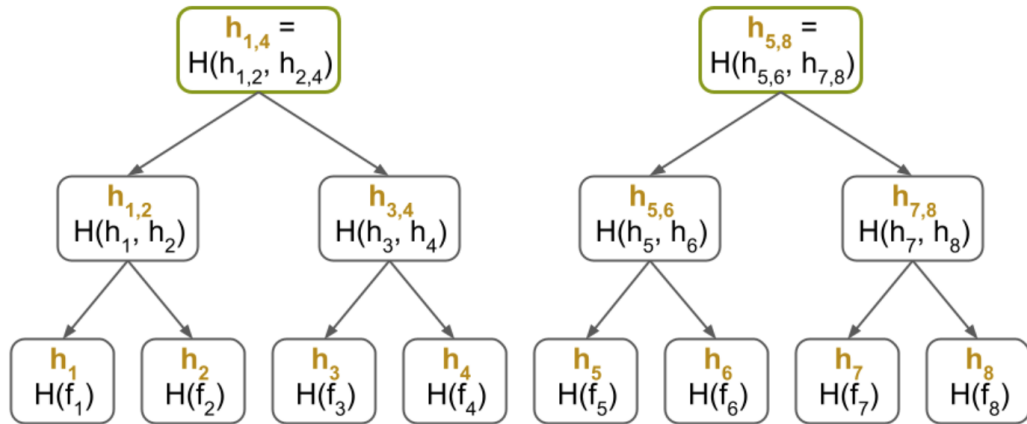
- ▶ Nakon formiranja listova, heširamo svaka dva susedna *hash*-a, da bi formirali sledeći nivo $h_{k,m} = H(h_i, h_{i+1})$
- ▶ Ako nam fali *hash*, da bi svako imao suseda :(), prosto napravimo prazan *hash* i nastavimo dalje



(Decentralized Thoughts, Merkle trees)

Merkle stablo - formiranje, primer

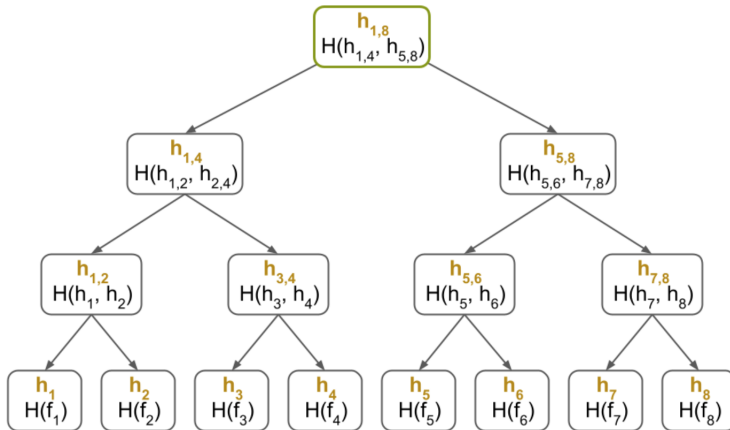
- Formiramo naredni nivo stabla



(Decentralized Thoughts, Merkle trees)

Merkle stablo - formiranje, primer

Idemo isto... i dobijamo $h_{1,8} = H(h_{1,4}, h_{5,8})$



(Decentralized Thoughts, Merkle trees)

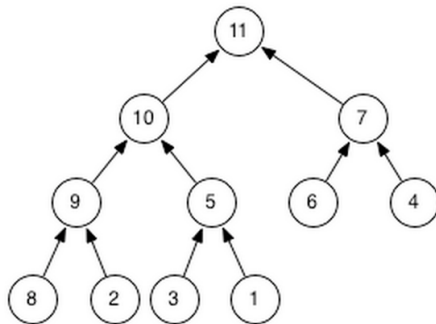
Merkle stablo — napomena

- ▶ Ono što smo dobili na kraju $h_{1,8}$ je **Merkle root hash**
- ▶ Obratiti pažnju da svaki čvor u stablu čuva **hash** vrednost
- ▶ Listovi čuvaju hash vrednost (blokova) podataka $h_i = (h_1, \dots, h_8)$
- ▶ Čvorovi koji nisu listovi, i nisu **Merkle root hash**, čuvaju *hash* vrednost svoje dece — **internal node**

- ▶ Ako nam na nekom nivou fali par za neki element, prosto dodamo **prazan hash** kako bismo formirali par
- ▶ Može se lako generalizovati i izračunati Merkle stablo za bilo koji broj n podataka
- ▶ Formalno zapisano, prethodni primer se može zapisati kao $h_{1,8} = \text{MHT}(f_1, \dots, f_8)$
- ▶ Merkle stabla se formiraju rekurzivno, od dna ka vrhu
- ▶ Ovaj proces može biti procesno zahtevan!
- ▶ To nikada nemojte izgubiti iz vida

Serijalizacija stabla — jedan primer

- ▶ Ako imamo stablo kao sa slike
- ▶ Treba da idemo kroz njega, nekim od poznatih algoritama
- ▶ Jedna opcija je da idemo po nivoima:
 - ▶ [11 10 7 9 5 6 4 8 2 3 1]
- ▶ Treba voditi računa da ako na nekom nivou imamo manjka elmenata, treba da zapišemo nekakav marker da nam bude jasan znak za kasnije!
- ▶ Ovo neće biti problem kod Merkle stabala, ali u opštem slučaju treba voditi računa



(Ritambhara, Storing Binary Tree in a file)

Zadaci

- ▶ Implementirati Merkle stablo za proizvoljan skup podataka
- ▶ Koristiti elemente date u helper fajlu
- ▶ Serijalizovati Merkle stablo i sačuvati ga u fajlu