Napredni algoritmi i strukture podataka

Merkle stabla, Serijalizacija stabla

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

- Formalno gledamo, Merkle stabla uzimaju skup podataka $(x_1, \ldots, x_n)$ na ulaz
- Povratnu vrednost je **Merkle root hash** $h = \mathrm{MHT}(x_1, \ldots, x_n)$
- 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 $H(a) = H(b)$

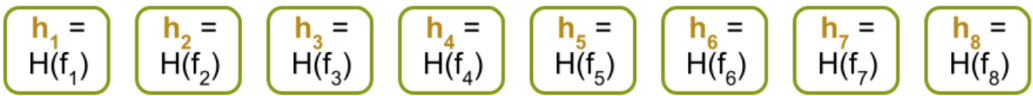
Merkle stablo — formiranje

- Algoritam za formiranje Merkle stabla je relativno jednostavan
- Merkle stablo ima **bottom-top** pristup, 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**

- Svaki **list** propustimo kroz *hash* funkciju, a 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* elment 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

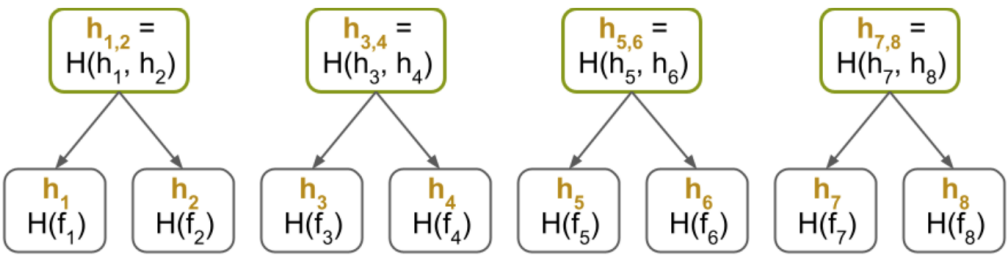
- Pretpostavimo da imamo 8 blokova podataka (fajlova) $f = (f_1, ..., f_8)$
- Svaki podataka f_i propustimo kroz hash funckiju H i dobijamo njegov *hash*
- Dobijamo hash vrednost za prvi nivo $h_i = H(fi), h_i = (h_1, ..., h_8)$
- H reprezentuje **collision-resistant hash** funkciju



(Decentralized Thoughts, Merkle trees)

Merkle stablo - formiranje, primer

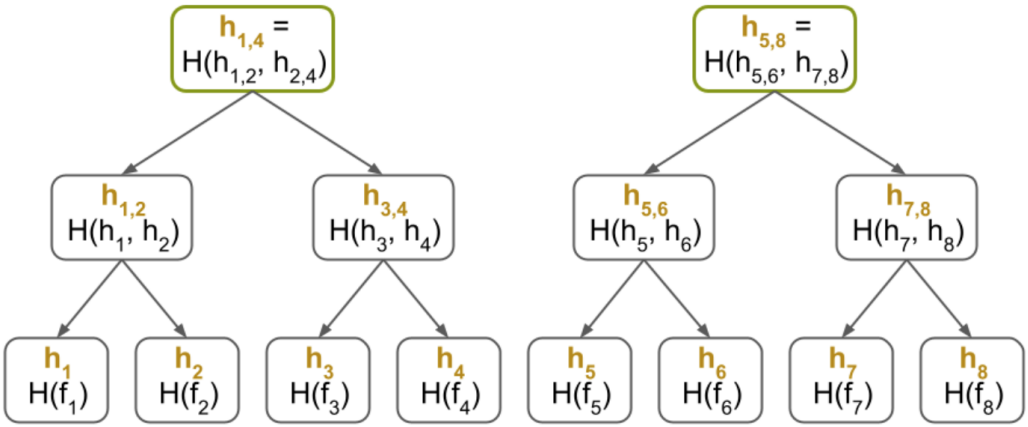
- Zabava se nastavlja u istom maniru, samo za naredne nivoe nam trebaju parovi :)
- Heširamo svaka dva susedna *hash-a*, da bi formirali sledeći nivo
  $h_{k,m} = H(h_i, h_{i+1})$
- Ako nam fali *hash*, da bi svako imao suseda :(, prosto napravimo prazan *hash* i nastavimo dalje (nećemo se stresirati oko gluposti)



(Decentralized Thoughts, Merkle trees)

Merkle stablo - formiranje, primer

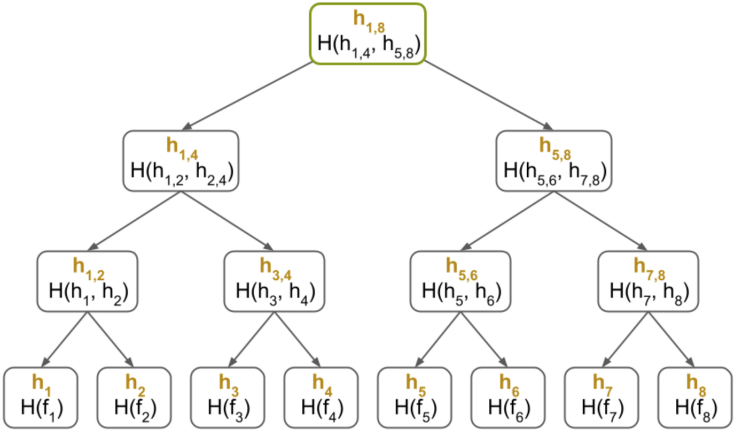
- Zabava se nastavlja isto kao i prethodno :)



(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)

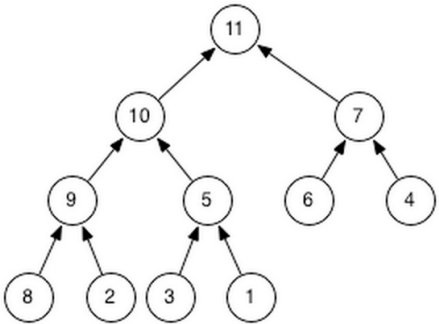
Merkle stablo — napomena

- Ono što smo deobili 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 $h_i = (h_1, ..., h_8)$
- 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** da bi formirali par
- Može se lako generalizovati i izražunati Merkle stablo za bilo koji broj n podataka
- Formalno zapisano, prethodni primer se može zapisati kao $h_{1,8} = \mathrm{MHT}(f_1, ..., f_8)$
- 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 ako na nekom nivo 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)

Merkle stabla - Zadaci

- Implementirati Merkle stablo za proizvoljan skup podataka
- Koristiti elemente date u kroz helper fajl
- Serijalizovati Merkle stablo u fajl pod nazivom *Metadata.txt*
- U fajlu treba da budu zapisano samo *hash* vrednosti u tekstualnom obliku kao niz elemenata (za primer pogledati slajd o serijalizaciji)