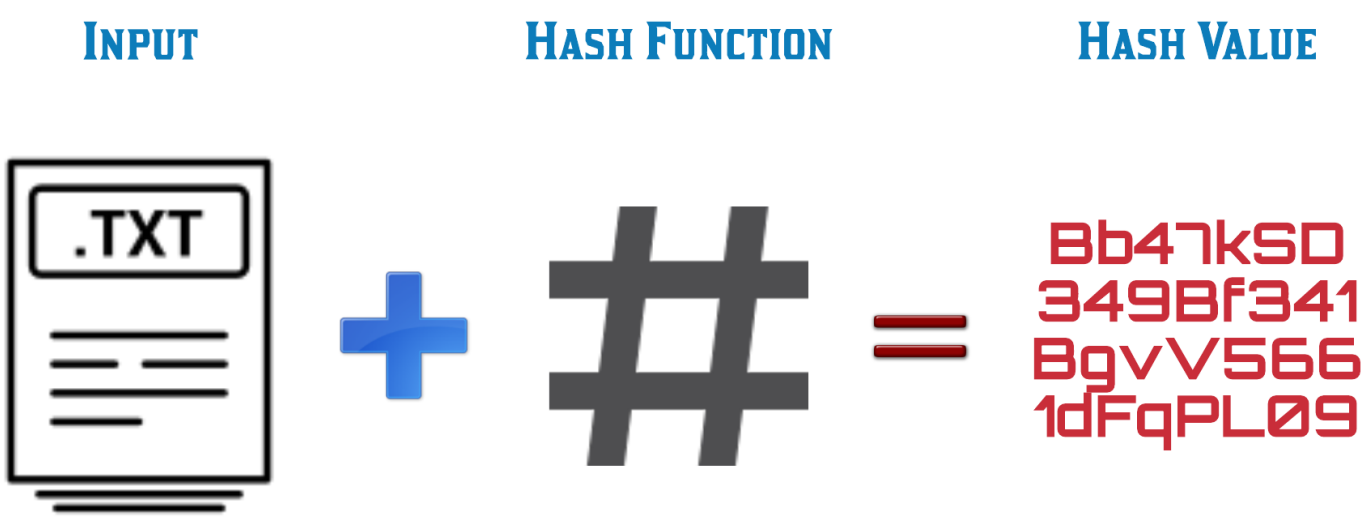
1-oji užduotis: Hash generatoriaus kūrimas

Įvadas

Maišos funkcija (angl. *hash function*) yra labai svarbi *blockchain* tinklų, tokių kaip *Bitcoin, Ethereum* ir kt. sudedamoji dalis. Hash'avimo metu bet koks įvedimo tekstas (m) panaudojant matematines *hash* funkcijas: $h = h(m)$ yra paverčiamas unikaliu fiksuoto dydžio pseudo-atsitiktiniu simbolių rinkiniu, vadinamu *maišos kodu*. Tradicinė tokių *hash* generatorių veikimo schema yra pateikta žemiau esančiame paveiksle:



Kad geriau pajusti veikimą, rekomenduojame pasibandyti, kaip veikia vieni geriausių ir plačiausiai naudojamų, maišos kodo generatorių, pvz., SHA256.

Praktinės užduoties formuluotė

Sukurkite Jūsų (t.y. pabandykite neieškoti *hash* funkcijos realizacijos pavyzdžių internete) maišos funkciją (*hash* kodų generatorių), kuris **pasižymėtų šiais *hash* funkcijoms keliamais reikalavimais**:

1. Maišos funkcijos įėjimas (angl. *input*) gali būti bet kokio dydžio simbolių eilutė (angl. *string*).
2. Maišos funkcijos išėjimas (angl. *output*) visuomet yra to paties, fiksuoto, dydžio rezultatas (pageidautina 256 bit'ų ilgio, t.y., 64 simbolių hex'as).
3. Maišos funkcija yra ***deterministinė***, t. y., tam pačiam įvedimui (*input*'ui) išvedimas (*output*'as) visuomet yra tas pats.

4. Maišos funkcijos reikšmė/kodas (hash'as) bet kokiai input'o reikšmei yra apskaičiuojamas greitai - efektyviai.
5. Iš *hash* funkcijos rezultato (*output*'o) praktiškai neįmanoma atgaminti pradinio įvedimo (*input*'o).
6. Maišos funkcija yra ***atspari "kolizijai"*** (angl. *collision resistance*), t.y., praktiškai neįmanoma surasti tokių dviejų skirtingų argumentų $m_1 \neq m_2$, kad jiems gautume tą patį *hash*'ą: $h(m_1) = h(m_2)$.
7. Bent minimaliai pakeitus įvedimą, pvz., vietoj "Lietuva" pateikus "lietuva", maišos funkcijos rezultatas-maišos kodas turi skirtis iš esmės, t.y., turi būti tenkinamas taip vadinamas lavinos efektas (angl. Avalanche effect). Žemiau esančioje lentelėje šis efektas iliustruotas panaudojant SHA256 generatorių:

Įvedimas (*input*)	Išvedimas (*hash'as* gautas iš SHA256)
lietuva	f51f6afefb2616f48bbddeeada2d729244a00fa0817f9ceb5c5419aa04b31172
Lietuva	5109820f748796128b8bafd3806d05511bc89ad77fc3cda960facf37a639bc7f
Lietuva!	f4ac741acca7dd6f5f7e6fd1e382eca604a26ba21a83a6a2215d7be830a8faa6

Reikalavimai versijai (`v0.1`) (Terminas: 2024-09-26)

- Pagal praktinės užduoties formuluotę, realizuokite *hash*'avimo generatorių (pageidautina `C++` ar jai ekvivalenčioje/giminingoje programavimo kalboje).
  - Programos realizavimas turi būti versijuojamas (pvz. `v0.1`, `v0.2` ir t.t.) *git*'e bei patalpintas Jūsų asmeniniame Github viešoje (angl. *public*) repozicijoje.
- Programos realizacijoje hash'avimui reikiamą *input*'ą, esantį išoriniame faile, reikia nurodyti per Command Line Argument'ą.
  - Papildomai, turi būti realizuota galimybė input'ą įvesti ir ranka.
- Repozicijos `README.md` faile aprašykite Jūsų maišos funkcijos idėją pseudo-kodo stiliumi, t.y., paprastai akcentuojant kokie žingsniai yra atliekami hash'avimo metu.
- Atlikite eksperimentinę analizę (žr. žemiau Eksperimentinis tyrimas ir rezultatų analizė), kurios metu įsitikinkite, kad Jūsų *hash* funkcija-generatorius iš tiesų pasižymi aukščiau (žr. Užduoties formuluotė) aprašytais *hash* funkcijoms keliamais reikalavimais.
  - Atliktą tyrimą ir gautuosius rezultatus išsamiai aprašykite `README.md` faile.
  - Pažymime, kad atsiskaitomosios paskaitos metu reikės pademonstruoti, kaip buvo atliekamas tyrimas ir kaip testavote, kad Jūsų maišos funkcija pasižymi šiomis savybėmis.

- Jei matote, kad Jūsų hash funkcija turi daug ir labai svarbių trūkumų, patobulinkite ją (dabar jau galite idėjų pasisemti ir iš žinomų hash funkcijų), sukurkite kaip naują release, pvz. `v0.2` bei pakartokite visus eksperimentus taip įsitikinant, kad naujoji versija gerokai pranašesnė už pirmykštę.
- Papildykite `README.md` failą naujos versijos rezultatais.

Eksperimentinis tyrimas ir rezultatų analizė

1. Susikurkite testinių įvedimo failų pavyzdžių, tokių kad:
   - Bent du failai būtų sudaryti **tik iš vieno, tačiau skirtingo**, simbolio.
   - Bent du failai būtų sudaryti iš daug (> 1000) atsitiktinai sugeneruotų simbolių.
   - Bent du failai būtų sudaryti iš daug (> 1000) simbolių, bet **skirtųsi vienas nuo kito tik vienu (pvz. vidurinėje pozicijoje esančiu) simboliu**.
   - Tuščio failo.
2. Naudojant pirmame žingsnyje susikurtus testinius failus, kaip Jūsų programos *input*'us, įsitikinkite, kad Jūsų *hash funkcija* atitinka 1-3 reikalavimus, t.y., nepriklausomai nuo Input'o, Output'ai visada yra vienodo dydžio, o to paties failo *hash*'as yra tas pats (deterministiškumas).
3. Ištirkite Jūsų sukurtos *hash* funkcijos efektyvumą, t.y., patikrinkite, kaip Jūsų hash'avimo funkcija atitinka 4-ą reikalavimą. Tuo tikslu pirmiausiai suhash'uokite vieną eilutę iš failo konstitucija.txt ir išmatuokite kiek laiko visa tai užtruko.
   - Tuomet pakartokite eksperimentą hash'uojant 2 eilutes, 4 eilutes, 8 eilutes ir t.t. (16, 32, 64, ...). Pažymime, kad reikia matuoti, tik *hash*'avimo funkcijos veikimo laiką (be input'o nuskaitymo/parengimo). Reiktų pateikti suvidurkintą (kartojant tą patį eksperimentą pvz. 5 kartus) hash'avimo laiko priklausomybę nuo input dydžio (eilučių skaičiaus). Kitaip sakant, reikia ištirti, kaip hash'avimo laikas didėja, didėjant input dydžiui.
   - Rezultatą prašytume pavaizduoti grafiškai, parodant kaip laikas priklauso nuo input dydžio.
   - Galima (bet ne privaloma) ir algoritmų sudėtingumą (tiesinis, kvadratinis, logaritminis, eksponentinis ir pan.) pasitelkti, norint įvertinti šią priklausomybę.
4. Susigeneruokite bent 100 000 atsitiktinių simbolių eilučių (`string` 'ų) porų, pvz. (**asdfg**, **hijkl**), apsiribojant iki 1000 simbolių ilgiu. Toje pačioje poroje esančių *string*'ų ilgiai turi sutapti, tačiau skirtingos poros gali būti skirtingo ilgio. Rekomenduojame susigeneruoti taip: 25 000 porų, kurių ilgis 10 simbolių, kitas 25 000 porų, kurių ilgis - 100, dar kitas 25 000 poras - 500, ir galiausiai likusias 25 000 poras, kurių ilgis - 1000 simbolių.
5. Naudodami 4 žingsnyje sugeneruotas poras, patikrinkite, ar visais atvejais gautieji **porų** hash'ai nesutampa. O jeigu sutampta, tai kaip dažnai tai nutinka. Tokiu būdu (jei visuomet *hash*'ai nesutampa) bent dalinai įsitikinsite, kad Jūsų *hash* funkcija atitinka 6-ą

reikalavimą, t.y., atsparumą kolizijai.

6. Susigeneruokite bent 100 000 atsitiktinių simbolių eilučių (`string` 'ų) porų, apsiribojant iki 1000 simbolių eilučių ilgiu (kaip ir aukščiau), taip, kad jos skirtųsi tik vienu simboliu pvz.: (**a**sdfg, **b**sdfg). Įvertinkite Jūsų gautų hash'ų procentinį "skirtingumą":
    - **bitų lygmenyje**;
    - **hex'ų lygmenyje**;
    - Išveskite minimalią, maksimalią ir vidurkines "skirtingumo" reikšmes. Tokiu būdu įsitikinsite, kaip gerai Jūsų *hash* funkcija atitinka 7-ą reikalavimą (lavinos efektą).
7. Galiausiai `README.md` faile apibendrinkite viso šio atlikto tyrimo išvadas: kur yra Jūsų *hash* funkcijos stiprybės ir kokie buvo nustatyti trūkumai?

Darbų vertinimas (Preliminari atsiskaitymo data: 2024-10-03)

- Iki 2.0 balų gausite atlikę visas aukščiau aprašytas užduotis pagal pateiktus reikalavimus.
- Vertinant, bus griežtai tikrinama, kuriuo metu buvo atliekami commi'ai ir releas'ai, bei kaip Jūsų projektas "augo". Taip pat bus atsižvelgiama į kūrybiškumą - ar nėra atkartoti kiti žinomi algoritmai, ir žinoma, bus tikrinamas plagijavimas iš kitų.

Papildomos užduotys

1. Pabandykite kaip įmanoma *objektyviau* palyginti Jūsų Hash funkcijos spartą su `MD5`, `SHA-1`, `SHA-256` ar kita gerai žinoma *hash* funkcija. Paliekame Jums sugalvoti, kaip atlikti tokį palyginimą ir nuo jo objektyvumo priklausys ir bonus'o dydis. [**Papildomai: iki 0.25 balo**]
2. Parodykite, kad iš *hash* funkcijos rezultato (*output*'o) praktiškai neįmanoma atgaminti pradinio įvedimo (*input*'o), t. y., kaip Jūsų hash funkcijoje realizuota *hiding* ir *puzzle-friendliness* savybės. P.s. manau, kad šioje vietoje "druskos" tikrai nebus per daug :) [**Papildomai: iki 0.25 balo**]
3. Reiktų kiek įmanoma daugiau Jūsų grupės/pogrupio sukurtų *hash* funkcijų/generatorių apjungti/integruoti į vieno iš Jūsų programą. Aišku, tai gali būti ir visiškai naują programą, kurioje būtų iškviečiamos visų sukurtos funkcijos.
    - Tuomet atlikti aukščiau aprašytą *lyginamąją analizę* (pagal 3-6 eksperimentinio tyrimo-analizės atlikimo punktus) naudojant Jūsų grupės/pogrupio kolegų sukurtus hash generatorius. Gautus grupės/pogrupio rezultatus - agreguokite - sureitinguokite. [**Papildomai: iki 0.5 balo**]
    - Tuomet jeigu Jūsų sukurtas *hash* generatorius pateks tarp 25% geriausių Jūsų grupėje/pogrupyje (Q1 - pirmasis kvartilis), visi šių generatorių autoriai gaus [**Papildomai: 0.25 balo**].