Análise do Union-Find

CLRS cap 21

Algoritmo de Kruskal

Dado um grafo G conexo com custo c_e para cada aresta e, encontra uma árvore geradora em G de custo mínimo.

KRUSKAL (G, c)

$A \leftarrow \emptyset$
sejam $e_1, \ldots, e_m$ as arestas de G ordenadas por c
para cada $u \in V(G)$ **faça** $\text{MakeSet}(u)$
para $i \leftarrow 1$ **até** m **faça**
    sejam u e v as pontas de e_i
    se $\text{FindSet}(u) \neq \text{FindSet}(v)$
        então $A \leftarrow A \cup \{e_i\}$
                $\text{Union}(u, v)$
devolva A

Union-find mantém a partição do conjunto de vértices nas componentes da floresta A.

Coleção de conjuntos disjuntos

Queremos uma ED boa para representar uma partição de um conjunto, e as seguintes operações sobre a partição:

- MakeSet(x): cria um conjunto unitário com o elemento x;
- FindSet(x): devolve o identificador do conjunto da partição que contém x;
- Union(x, y): substitui os conjuntos da partição que contêm x e y pela união deles.

Coleção de conjuntos disjuntos

Queremos uma ED boa para representar uma partição de um conjunto, e as seguintes operações sobre a partição:

- MakeSet(x): cria um conjunto unitário com o elemento x;
- FindSet(x): devolve o identificador do conjunto da partição que contém x;
- Union(x, y): substitui os conjuntos da partição que contêm x e y pela união deles.

O identificador de um conjunto é um elemento do conjunto: o seu representante.

Coleção de conjuntos disjuntos

Queremos uma ED boa para representar uma partição de um conjunto, e as seguintes operações sobre a partição:

- MakeSet(x): cria um conjunto unitário com o elemento x;
- FindSet(x): devolve o identificador do conjunto da partição que contém x;
- Union(x, y): substitui os conjuntos da partição que contêm x e y pela união deles.

O identificador de um conjunto é um elemento do conjunto: o seu representante.

Como podemos armazenar cada conjunto da partição?

Implementação 1 do union-find

Make-Set (x)

$\mathrm{pai}[x] \leftarrow x$

Implementação 1 do union-find

Make-Set (x)

$\text{pai}[x] \leftarrow x$

Find (x)

$r \leftarrow x$

enquanto $\text{pai}[r] \neq r$ **faça**

  $r \leftarrow \text{pai}[r]$

devolva r

Implementação 1 do union-find

```
Make-Set (x)
    pai[x] ← x

Find (x)
    r ← x
    enquanto pai[r] ≠ r faça
        r ← pai[r]
    devolva r

Union (x, y)                    ▷ x e y representantes distintos
    pai[y] ← x
```

Implementação 1 do union-find

```
Make-Set (x)
    pai[x] ← x

Find (x)
    r ← x
    enquanto pai[r] ≠ r faça
        r ← pai[r]
    devolva r

Union (x, y)                ▷ x e y representantes distintos
    pai[y] ← x
```

Consumo de tempo: do Find pode ser muito ruim... $\Theta(n)$.
Temos que fazer melhor...

Implementação 2

Heurística dos tamanhos

$\text{Make-Set}\ (x)$
1 $\quad \text{pai}[x] \leftarrow x$
2 $\quad \text{rank}[x] \leftarrow 0 \qquad \triangleright$ altura da árvore

Implementação 2

Heurística dos tamanhos

Make-Set (x)
1 $\mathrm{pai}[x] \leftarrow x$
2 $\mathrm{rank}[x] \leftarrow 0$ $\triangleright$ altura da árvore

Find (x): o mesmo de antes

Implementação 2

Heurística dos tamanhos

Make-Set (x)

    pai$[x] \leftarrow x$
    rank$[x] \leftarrow 0$ $\triangleright$ altura da árvore

Find (x): o mesmo de antes

Union (x, y) $\triangleright$ x e y representantes distintos

    se rank$[x] \geq$ rank$[y]$
        então pai$[y] \leftarrow x$
            se rank$[x] =$ rank$[y]$
                então rank$[x] \leftarrow$ rank$[x] + 1$
        senão pai$[x] \leftarrow y$

Implementação 2

Heurística dos tamanhos

```
Make-Set (x)
    pai[x] ← x
    rank[x] ← 0                        ▷ altura da árvore

Find (x): o mesmo de antes

Union (x, y)                           ▷ x e y representantes distintos
    se rank[x] ≥ rank[y]
        então pai[y] ← x
              se rank[x] = rank[y]
                 então rank[x] ← rank[x] + 1
        senão pai[x] ← y
```

Consumo de tempo: melhor... $\Theta(\lg n)$. (Por que? Sabe explicar?)

Dá para fazer melhor ainda!

Implementação 3

Heurística da compressão dos caminhos

```
Find (x)
  if pai[x] ≠ x
     então pai[x] ← Find (pai[x])
  devolva pai[x]
```

Implementação 3

Heurística da compressão dos caminhos

Find (x)
if $\mathrm{pai}[x] \neq x$
    então $\mathrm{pai}[x] \leftarrow$ Find $(\mathrm{pai}[x])$
devolva $\mathrm{pai}[x]$

Consumo *amortizado* de tempo de cada operação:

$$\mathrm{O}(\lg^* n),$$

onde $\lg^* n$ é o número de vezes que temos que
aplicar o lg até atingir um número menor ou igual a 1.

Implementação 3

Heurística da compressão dos caminhos

```
Find (x)
1  if pai[x] ≠ x
2      então pai[x] ← Find (pai[x])
3  devolva pai[x]
```

Consumo *amortizado* de tempo de cada operação:

$$\mathrm{O}(\lg^* n),$$

onde $\lg^* n$ é o número de vezes que temos que aplicar o lg até atingir um número menor ou igual a 1.

Na verdade, é melhor do que isso, e há uma análise justa.

Union-Find

```
Make-Set (x)
    pai[x] ← x
    rank[x] ← 0

Find (x)
    if pai[x] ≠ x
        então pai[x] ← Find (pai[x])
    devolva pai[x]

Union (x, y)                    ▷ x e y representantes distintos
    se rank[x] ≥ rank[y]
        então  pai[y] ← x
                   se rank[x] = rank[y]
                       então rank[x] ← rank[x] + 1
        senão pai[x] ← y
```

Union-Find

Union (x, y)

$x' \leftarrow$ **Find** (x)

$y' \leftarrow$ **Find** (y)

se $x' \neq y'$

  então **Link** (x', y')

Link (x, y) $\quad \triangleright$ x e y representantes distintos

se $\mathrm{rank}[x] \geq \mathrm{rank}[y]$

  então $\mathrm{pai}[y] \leftarrow x$

    se $\mathrm{rank}[x] = \mathrm{rank}[y]$

      então $\mathrm{rank}[x] \leftarrow \mathrm{rank}[x] + 1$

  senão $\mathrm{pai}[x] \leftarrow y$

Consumo de tempo

Dada sequência de makeset, findset e union,
converta-a em uma sequência de makeset, findset e link.

Consumo de tempo

Dada sequência de makeset, findset e union,
converta-a em uma sequência de makeset, findset e link.

Sequência de m operações makeset, findset e link
das quais n são makeset.

Consumo de tempo

Dada sequência de makeset, findset e union,
converta-a em uma sequência de makeset, findset e link.

Sequência de m operações makeset, findset e link
das quais n são makeset.

Custo de pior caso de cada operação: $\mathrm{O}(\lg n)$.
Custo amortizado de cada operação: $\mathrm{O}(\lg^* n)$.

Consumo de tempo

Dada sequência de makeset, findset e union,
converta-a em uma sequência de makeset, findset e link.

Sequência de m operações makeset, findset e link
das quais n são makeset.

Custo de pior caso de cada operação: $\mathrm{O}(\lg n)$.
Custo amortizado de cada operação: $\mathrm{O}(\lg^* n)$.

Para definir $\lg^* n$, seja $\lg^{(1)} x = \lg x$.
Para $i \geq 2$, seja $\lg^{(i)} x = \lg(\lg^{(i-1)} x)$.

Consumo de tempo

Dada sequência de makeset, findset e union,
converta-a em uma sequência de makeset, findset e link.

Sequência de m operações makeset, findset e link
das quais n são makeset.

Custo de pior caso de cada operação: $\mathrm{O}(\lg n)$.
Custo amortizado de cada operação: $\mathrm{O}(\lg^* n)$.

Para definir $\lg^* n$, seja $\lg^{(1)} x = \lg x$.
Para $i \geq 2$, seja $\lg^{(i)} x = \lg(\lg^{(i-1)} x)$.

Então $\lg^* n \;=\; \min\{i : \lg^{(i)} n \leq 1\}$.

Consumo de tempo

Dada sequência de makeset, findset e union,
converta-a em uma sequência de makeset, findset e link.

Sequência de m operações makeset, findset e link
das quais n são makeset.

Custo de pior caso de cada operação: $\mathrm{O}(\lg n)$.
Custo amortizado de cada operação: $\mathrm{O}(\lg^* n)$.

Para definir $\lg^* n$, seja $\lg^{(1)} x = \lg x$.
Para $i \geq 2$, seja $\lg^{(i)} x = \lg(\lg^{(i-1)} x)$.

Então $\lg^* n \;=\; \min\{i : \lg^{(i)} n \leq 1\}$.

A análise desta ED é vista na disciplina MAC6711.

Hashing

KT Secs 13.6

Hashing universal

U: conjunto universo (contém todas as possíveis chaves).

n: um número muito menor que $|U|$.

$\mathcal{H}$: conjunto de funções de U em $\{0, \ldots, n-1\}$.

Hashing universal

U: conjunto universo (contém todas as possíveis chaves).

n: um número muito menor que $|U|$.

$\mathcal{H}$: conjunto de funções de U em $\{0, \ldots, n-1\}$.

$\mathcal{H}$ é uma coleção universal de hashing se,
para cada par de chaves k, ℓ em U,
o número de funções h em $\mathcal{H}$ tais que $h(k) = h(\ell)$
é no máximo $|\mathcal{H}|/n$.

Hashing universal

U: conjunto universo (contém todas as possíveis chaves).

n: um número muito menor que $|U|$.

$\mathcal{H}$: conjunto de funções de U em $\{0, \ldots, n-1\}$.

$\mathcal{H}$ é uma coleção universal de hashing se,
para cada par de chaves k, ℓ em U,
o número de funções h em $\mathcal{H}$ tais que $h(k) = h(\ell)$
é no máximo $|\mathcal{H}|/n$.

Fixe $k, \ell \in U$.

O que acontece se escolhermos uma h em $\mathcal{H}$
aleatoriamente com probabilidade uniforme?

Hashing universal

U: conjunto universo (contém todas as possíveis chaves).

n: um número muito menor que $|U|$.

$\mathcal{H}$: conjunto de funções de U em $\{0, \ldots, n-1\}$.

$\mathcal{H}$ é uma coleção universal de hashing se,
para cada par de chaves k, ℓ em U,
o número de funções h em $\mathcal{H}$ tais que $h(k) = h(\ell)$
é no máximo $|\mathcal{H}|/n$.

Fixe $k, \ell \in U$.

O que acontece se escolhermos uma h em $\mathcal{H}$
aleatoriamente com probabilidade uniforme?

Qual é a chance de $h(k) = h(\ell)$?

Hashing universal

n: um número muito menor que $|U|$.

$\mathcal{H}$: conjunto de funções de U em $\{0, \ldots, n-1\}$.

$\mathcal{H}$ é uma coleção universal de hashing se,
para cada par de chaves k, ℓ em U, o número de funções h em $\mathcal{H}$ tais que $h(k) = h(\ell)$ é no máximo $|\mathcal{H}|/n$.

Fixe $k, \ell \in U$.

O que acontece se escolhermos uma h em $\mathcal{H}$ aleatoriamente com probabilidade uniforme?

Qual é a chance de $h(k) = h(\ell)$?

É, dentre todas as $|\mathcal{H}|$ funções h, escolhermos uma das no máximo $|\mathcal{H}|/n$ para as quais vale a igualdade.

Hashing universal

n: um número muito menor que $|U|$.

$\mathcal{H}$: conjunto de funções de U em $\{0, \ldots, n-1\}$.

$\mathcal{H}$ é uma coleção universal de hashing se,
para cada par de chaves k, ℓ em U, o número de
funções h em $\mathcal{H}$ tais que $h(k) = h(\ell)$ é no máximo $|\mathcal{H}|/n$.

Fixe $k, \ell \in U$.

O que acontece se escolhermos uma h em $\mathcal{H}$
aleatoriamente com probabilidade uniforme?

Qual é a chance de $h(k) = h(\ell)$?

É, dentre todas as $|\mathcal{H}|$ funções h, escolhermos uma
das no máximo $|\mathcal{H}|/n$ para as quais vale a igualdade.
Ou seja, é no máximo $1/n$.

Formalizando...

Teorema: Seja $S \subseteq U$ tal que $|S| \leq n$ e $u \in U$.
Se h é escolhida aleatoriamente de uma coleção universal $\mathcal{H}$ e X é o número de elementos s em S tais que $h(s) = h(u)$, então $\mathrm{E}[X] \leq 1$ se $u \notin S$ e $\mathrm{E}[X] \leq 2$ se $u \in S$.

Formalizando...

Teorema: Seja $S \subseteq U$ tal que $|S| \leq n$ e $u \in U$.
Se h é escolhida aleatoriamente de uma coleção universal $\mathcal{H}$ e X é o número de elementos s em S tais que $h(s) = h(u)$, então $\mathrm{E}[X] \leq 1$ se $u \notin S$ e $\mathrm{E}[X] \leq 2$ se $u \in S$.

Prova:

Seja X_s a variável binária que vale 1 se $h(s) = h(u)$.

Formalizando...

Teorema: Seja $S \subseteq U$ tal que $|S| \leq n$ e $u \in U$.
Se h é escolhida aleatoriamente de uma coleção universal $\mathcal{H}$ e X é o número de elementos s em S tais que $h(s) = h(u)$, então $\mathrm{E}[X] \leq 1$ se $u \notin S$ e $\mathrm{E}[X] \leq 2$ se $u \in S$.

Prova:

Seja X_s a variável binária que vale 1 se $h(s) = h(u)$.

Note que $X = \sum_s X_s$.

Formalizando...

Teorema: Seja $S \subseteq U$ tal que $|S| \leq n$ e $u \in U$.
Se h é escolhida aleatoriamente de uma coleção universal $\mathcal{H}$ e X é o número de elementos s em S tais que $h(s) = h(u)$, então $\mathrm{E}[X] \leq 1$ se $u \notin S$ e $\mathrm{E}[X] \leq 2$ se $u \in S$.

Prova:

Seja X_s a variável binária que vale 1 se $h(s) = h(u)$.

Note que $X = \sum_s X_s$.

Então $\Pr\{X_u = 1\} = 1$ e $\Pr\{X_s = 1\} \leq \frac{1}{n}$ se $s \neq u$.

Formalizando...

Teorema: Seja $S \subseteq U$ tal que $|S| \leq n$ e $u \in U$.
Se h é escolhida aleatoriamente de uma coleção universal $\mathcal{H}$
e X é o número de elementos s em S tais que $h(s) = h(u)$,
então $\mathrm{E}[X] \leq 1$ se $u \notin S$ e $\mathrm{E}[X] \leq 2$ se $u \in S$.

Prova:

Seja X_s a variável binária que vale 1 se $h(s) = h(u)$.

Note que $X = \sum_s X_s$.

Então $\Pr\{X_u = 1\} = 1$ e $\Pr\{X_s = 1\} \leq \frac{1}{n}$ se $s \neq u$.

Logo
$$\mathrm{E}[X] = \sum_{s \in S} \mathrm{E}[X_s] \leq \sum_{s \in S} \frac{1}{n} = \frac{|S|}{n} \leq 1 \quad \text{se } u \notin S$$

Formalizando...

Teorema: Seja $S \subseteq U$ tal que $|S| \leq n$ e $u \in U$.
Se h é escolhida aleatoriamente de uma coleção universal $\mathcal{H}$
e X é o número de elementos s em S tais que $h(s) = h(u)$,
então $\mathrm{E}[X] \leq 1$ se $u \notin S$ e $\mathrm{E}[X] \leq 2$ se $u \in S$.

Prova:

Seja X_s a variável binária que vale 1 se $h(s) = h(u)$.

Note que $X = \sum_s X_s$.

Então $\Pr\{X_u = 1\} = 1$ e $\Pr\{X_s = 1\} \leq \frac{1}{n}$ se $s \neq u$.

Logo

$$\mathrm{E}[X] = \sum_{s \in S} \mathrm{E}[X_s] \leq \sum_{s \in S} \frac{1}{n} = \frac{|S|}{n} \leq 1 \quad \text{se } u \notin S$$

e

$$\mathrm{E}[X] = 1 + \sum_{s \in S \setminus \{u\}} \frac{1}{n} < 1 + \frac{|S|}{n} \leq 2 \quad \text{se } u \in S. \quad \square$$

Exemplo de coleção universal de hashing

Seja p um primo tal que $U \subseteq \{0, \ldots, p-1\}$.

$\mathbb{Z}_p$: conjunto $\{0, \ldots, p-1\}$.
$\mathbb{Z}_p^*$: conjunto $\{1, \ldots, p-1\}$.

Exemplo de coleção universal de hashing

Seja p um primo tal que $U \subseteq \{0, \ldots, p-1\}$.

$\mathbb{Z}_p$: conjunto $\{0, \ldots, p-1\}$.
$\mathbb{Z}_p^*$: conjunto $\{1, \ldots, p-1\}$.

Para todo a em $\mathbb{Z}_p^*$ e b em $\mathbb{Z}_p$, seja

$$h_{a,b}(k) \;=\; ((ak + b) \bmod p) \bmod n,$$

para todo k em U.

Exemplo de coleção universal de hashing

Seja p um primo tal que $U \subseteq \{0, \ldots, p-1\}$.

$\mathbb{Z}_p$: conjunto $\{0, \ldots, p-1\}$.
$\mathbb{Z}_p^*$: conjunto $\{1, \ldots, p-1\}$.

Para todo a em $\mathbb{Z}_p^*$ e b em $\mathbb{Z}_p$, seja

$$h_{a,b}(k) \;=\; ((ak + b) \bmod p) \bmod n,$$

para todo k em U.

A coleção $\mathcal{H} \;=\; \{h_{a,b} : a \in \mathbb{Z}_p^* \text{ e } b \in \mathbb{Z}_p\}$ é universal.

Exemplo de coleção universal de hashing

Seja p um primo tal que $U \subseteq \{0, \ldots, p-1\}$.

$\mathbb{Z}_p$: conjunto $\{0, \ldots, p-1\}$.
$\mathbb{Z}_p^*$: conjunto $\{1, \ldots, p-1\}$.

Para todo a em $\mathbb{Z}_p^*$ e b em $\mathbb{Z}_p$, seja

$$h_{a,b}(k) \;=\; ((ak + b) \bmod p) \bmod n,$$

para todo k em U.

A coleção $\mathcal{H} \;=\; \{h_{a,b} : a \in \mathbb{Z}_p^* \text{ e } b \in \mathbb{Z}_p\}$ é universal.

Note que $|\mathcal{H}| = p(p-1)$.

Exemplo de coleção universal de hashing

Para todo a em $\mathbb{Z}_p^*$ e b em $\mathbb{Z}_p$, seja

$$h_{a,b}(k) \;=\; ((ak+b) \bmod p) \bmod n, \quad \text{para todo } k \text{ em } U.$$

A coleção $\mathcal{H} \;=\; \{h_{a,b} : a \in \mathbb{Z}_p^* \text{ e } b \in \mathbb{Z}_p\}$ é universal.

Exemplo de coleção universal de hashing

Para todo a em $\mathbb{Z}_p^*$ e b em $\mathbb{Z}_p$, seja

$$h_{a,b}(k) \ = \ ((ak + b) \bmod p) \bmod n, \quad \text{para todo } k \text{ em } U.$$

A coleção $\mathcal{H} \ = \ \{h_{a,b} : a \in \mathbb{Z}_p^* \text{ e } \ b \in \mathbb{Z}_p\}$ é universal.

Esboço da prova: Sejam $k, \ell \in U$ tais que $k \neq \ell$.

Exemplo de coleção universal de hashing

Para todo a em $\mathbb{Z}_p^*$ e b em $\mathbb{Z}_p$, seja

$$h_{a,b}(k) \;=\; ((ak+b) \bmod p) \bmod n, \quad \text{para todo } k \text{ em } U.$$

A coleção $\mathcal{H} \;=\; \{h_{a,b} : a \in \mathbb{Z}_p^* \text{ e } \; b \in \mathbb{Z}_p\}$ é universal.

Esboço da prova: Sejam $k, \ell \in U$ tais que $k \neq \ell$.

Queremos determinar quantas $h_{a,b} \in \mathcal{H}$ são tais que $h_{a,b}(k) = h_{a,b}(\ell)$.

Exemplo de coleção universal de hashing

Para todo a em $\mathbb{Z}_p^*$ e b em $\mathbb{Z}_p$, seja

$$h_{a,b}(k) = ((ak + b) \bmod p) \bmod n, \quad \text{para todo } k \text{ em } U.$$

A coleção $\mathcal{H} = \{h_{a,b} : a \in \mathbb{Z}_p^* \text{ e } b \in \mathbb{Z}_p\}$ é universal.

Esboço da prova: Sejam $k, \ell \in U$ tais que $k \neq \ell$.

Queremos determinar quantas $h_{a,b} \in \mathcal{H}$ são tais que $h_{a,b}(k) = h_{a,b}(\ell)$.

Se $(ak + b) \bmod p = (a\ell + b) \bmod p$,
então $a(k - \ell) = 0 \bmod p$, o que implica que $a = 0$ ou $k = \ell$.

Exemplo de coleção universal de hashing

Para todo a em $\mathbb{Z}_p^*$ e b em $\mathbb{Z}_p$, seja

$$h_{a,b}(k) \;=\; ((ak+b) \bmod p) \bmod n, \quad \text{para todo } k \text{ em } U.$$

A coleção $\mathcal{H} \;=\; \{h_{a,b} : a \in \mathbb{Z}_p^* \text{ e } \; b \in \mathbb{Z}_p\}$ é universal.

Esboço da prova: Sejam $k, \ell \in U$ tais que $k \neq \ell$.

Queremos determinar quantas $h_{a,b} \in \mathcal{H}$ são tais que $h_{a,b}(k) = h_{a,b}(\ell)$.

Se $(ak+b) \bmod p = (a\ell+b) \bmod p$,
então $a(k-\ell) = 0 \bmod p$, o que implica que $a = 0$ ou $k = \ell$.

Como $a \neq 0$ e $k \neq \ell$, $\quad r = (ak+b) \bmod p \neq (a\ell+b) \bmod p = s$.

Exemplo de coleção universal de hashing

Para todo a em $\mathbb{Z}_p^*$ e b em $\mathbb{Z}_p$, seja

$$h_{a,b}(k) \;=\; ((ak+b) \bmod p) \bmod n, \quad \text{para todo } k \text{ em } U.$$

A coleção $\mathcal{H} \;=\; \{h_{a,b} : a \in \mathbb{Z}_p^* \text{ e } b \in \mathbb{Z}_p\}$ é universal.

Esboço da prova: Sejam $k, \ell \in U$ tais que $k \neq \ell$.

Queremos determinar quantas $h_{a,b} \in \mathcal{H}$ são tais que $h_{a,b}(k) = h_{a,b}(\ell)$.

Se $(ak+b) \bmod p = (a\ell+b) \bmod p$,
então $a(k-\ell) = 0 \bmod p$, o que implica que $a = 0$ ou $k = \ell$.

Como $a \neq 0$ e $k \neq \ell$, $\quad r = (ak+b) \bmod p \neq (a\ell+b) \bmod p = s$.

Ademais, cada par (a, b) está associado a um par distinto (r, s), com $r \neq s$, já que $a = (r-s)(k-\ell)^{-1} \bmod p$ e $b = (r-ak) \bmod p$.

Exemplo de coleção universal de hashing

Para todo a em $\mathbb{Z}_p^*$ e b em $\mathbb{Z}_p$, seja

$$h_{a,b}(k) = ((ak + b) \bmod p) \bmod n, \quad \text{para todo } k \text{ em } U.$$

A coleção $\mathcal{H} = \{h_{a,b} : a \in \mathbb{Z}_p^* \text{ e } b \in \mathbb{Z}_p\}$ é universal.

Esboço da prova: Sejam $k, \ell \in U$ tais que $k \neq \ell$.

Queremos determinar quantas $h_{a,b} \in \mathcal{H}$ são tais que $h_{a,b}(k) = h_{a,b}(\ell)$.

Como $a \neq 0$ e $k \neq \ell$, $\quad r = (ak + b) \bmod p \neq (a\ell + b) \bmod p = s$.

Exemplo de coleção universal de hashing

Para todo a em $\mathbb{Z}_p^*$ e b em $\mathbb{Z}_p$, seja

$$h_{a,b}(k) \;=\; ((ak + b) \bmod p) \bmod n, \qquad \text{para todo } k \text{ em } U.$$

A coleção $\mathcal{H} \;=\; \{h_{a,b} : a \in \mathbb{Z}_p^* \text{ e } \; b \in \mathbb{Z}_p\}$ é universal.

Esboço da prova: Sejam $k, \ell \in U$ tais que $k \neq \ell$.

Queremos determinar quantas $h_{a,b} \in \mathcal{H}$ são tais que $h_{a,b}(k) = h_{a,b}(\ell)$.

Como $a \neq 0$ e $k \neq \ell$, $\quad r = (ak + b) \bmod p \neq (a\ell + b) \bmod p = s$.

Para cada r em $\mathbb{Z}_p$, são $p - 1$ valores possíveis para s em $\mathbb{Z}_p \setminus \{r\}$.
Destes, não mais que $p/n - 1 \leq (p-1)/n$ são tais que $s = r \bmod n$.

Exemplo de coleção universal de hashing

Para todo a em $\mathbb{Z}_p^*$ e b em $\mathbb{Z}_p$, seja

$$h_{a,b}(k) \;=\; ((ak+b) \bmod p) \bmod n, \quad \text{para todo } k \text{ em } U.$$

A coleção $\mathcal{H} \;=\; \{h_{a,b} : a \in \mathbb{Z}_p^* \text{ e } \; b \in \mathbb{Z}_p\}$ é universal.

Esboço da prova: Sejam $k, \ell \in U$ tais que $k \neq \ell$.

Queremos determinar quantas $h_{a,b} \in \mathcal{H}$ são tais que $h_{a,b}(k) = h_{a,b}(\ell)$.

Como $a \neq 0$ e $k \neq \ell$, $\quad r = (ak+b) \bmod p \neq (a\ell+b) \bmod p = s$.

Para cada r em $\mathbb{Z}_p$, são $p-1$ valores possíveis para s em $\mathbb{Z}_p \setminus \{r\}$.
Destes, não mais que $p/n - 1 \leq (p-1)/n$ são tais que $s = r \bmod n$.

Ou seja, $h_{a,b}(k) = h_{a,b}(\ell)$
para não mais que $p(p-1)/n = |\mathcal{H}|/n$ das funções $h_{a,b}$ de $\mathcal{H}$. $\square$

Fácil de usar!

Se o conjunto $U = \{0, 1, \ldots, N\}$.

Fácil de usar!

Se o conjunto $U = \{0, 1, \ldots, N\}$.

Fixe no programa um primo $p \geq N$.

Fácil de usar!

Se o conjunto $U = \{0, 1, \ldots, N\}$.

Fixe no programa um primo $p \geq N$.

Ao inicializar o hashing, escolha aleatoriamente
um inteiro a em $\{1, \ldots, p-1\}$ e um inteiro b em $\{0, \ldots, p-1\}$.

Fácil de usar!

Se o conjunto $U = \{0, 1, \ldots, N\}$.

Fixe no programa um primo $p \geq N$.

Ao inicializar o hashing, escolha aleatoriamente
um inteiro a em $\{1, \ldots, p-1\}$ e um inteiro b em $\{0, \ldots, p-1\}$.

Use a função de hashing $h_{a,b}(k) \ = \ ((ak + b) \bmod p) \bmod n$,
onde n é o número de pontos na coleção.