Splay trees

S: splay tree

$S_i(x)$: subárvore de S enraizada em x no instante i
$s_i(x) = |S_i(x)|$

Tome $r_i(x) = \lg s_i(x)$.

$r_i(x)$ indica o potencial local no nó x.

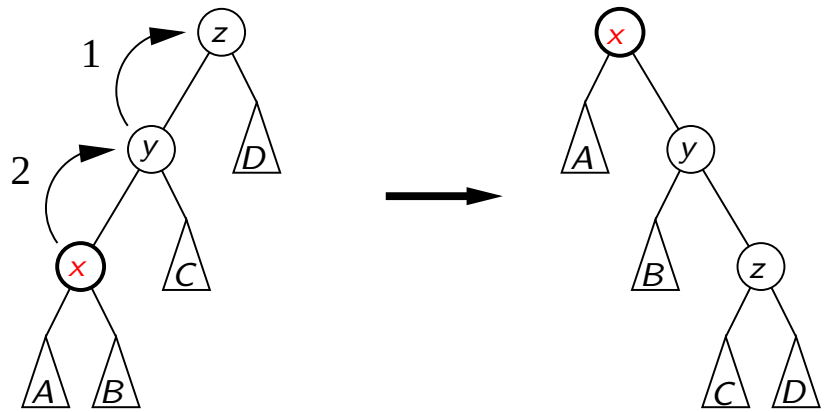
Seja $\Phi_i = \sum_x r_i(x)$.

Mostraremos que o custo amortizado por SPLAY é $\mathrm{O}(\lg n)$.

Análise amortizada dos splay steps.

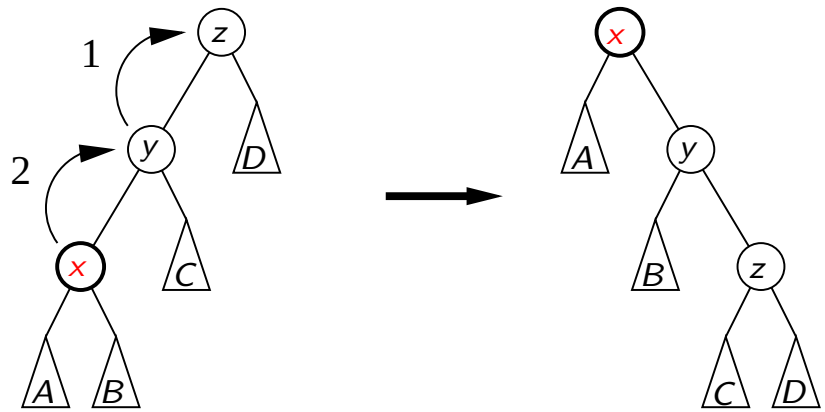
x participa de todos os splay steps de SPLAY(x, S).

Caso do rr splay step



Na aula passada, mostramos que $\hat{c}_i < 3\,(r_i(x) - r_{i-1}(x))$.

Caso do rr splay step



Na aula passada, mostramos que $\hat{c}_i < 3\,(r_i(x) - r_{i-1}(x))$.

Analogamente podemos mostrar que

$$\hat{c}_i < 3\,(r_i(x) - r_{i-1}(x))$$

para rl splay steps, lr splay steps, e ll splay steps, e que

$$\hat{c}_i \leq 3\,(r_i(x) - r_{i-1}(x)) + 1$$

para l splay steps e r splay steps.

Análise amortizada do splay

Se m é o número de splay steps no SPLAY(x, S)
e n o número de nós na árvore,

$$\begin{aligned}
\sum_{i=1}^{m} \hat{c}_i &\leq 3 \sum_{i=1}^{m} (r_i(x) - r_{i-1}(x)) + 1 \\
&= 3(r_m(x) - r_0(x)) + 1 \\
&\leq 3 \lg n + 1.
\end{aligned}$$

Análise amortizada do splay

Se m é o número de splay steps no SPLAY(x, S)
e n o número de nós na árvore,

$$\begin{aligned}\sum_{i=1}^{m} \hat{c}_i &\leq 3\sum_{i=1}^{m}(r_i(x) - r_{i-1}(x)) + 1 \\ &= 3(r_m(x) - r_0(x)) + 1 \\ &\leq 3 \lg n + 1.\end{aligned}$$

Então vale que o custo do SPLAY(x, S) é

$$\sum_{i=1}^{m} c_i = \sum_{i=1}^{m} \hat{c}_i - \Phi_m + \Phi_0 \leq 3 \lg n + 1 - \Phi_m + \Phi_0.$$

Análise amortizada do splay

Se m é o número de splay steps no SPLAY(x, S)
e n o número de nós na árvore,

$$\begin{aligned}\sum_{i=1}^{m} \hat{c}_i &\leq 3\sum_{i=1}^{m}(r_i(x) - r_{i-1}(x)) + 1 \\ &= 3(r_m(x) - r_0(x)) + 1 \\ &\leq 3 \lg n + 1.\end{aligned}$$

Então vale que o custo do SPLAY(x, S) é

$$\sum_{i=1}^{m} c_i = \sum_{i=1}^{m} \hat{c}_i - \Phi_m + \Phi_0 \leq 3 \lg n + 1 - \Phi_m + \Phi_0.$$

Lembre-se que $\Phi_i = \sum_x r_i(x)$, portanto $\Phi_i \geq 0$ para todo i.

Análise amortizada do splay

Se m é o número de splay steps no SPLAY(x, S)
e n o número de nós na árvore,

$$\begin{aligned}\sum_{i=1}^{m} \hat{c}_i &\leq 3\sum_{i=1}^{m}(r_i(x) - r_{i-1}(x)) + 1 \\ &= 3(r_m(x) - r_0(x)) + 1 \\ &\leq 3 \lg n + 1.\end{aligned}$$

Então vale que o custo do SPLAY(x, S) é

$$\sum_{i=1}^{m} c_i = \sum_{i=1}^{m} \hat{c}_i - \Phi_m + \Phi_0 \leq 3 \lg n + 1 - \Phi_m + \Phi_0.$$

Lembre-se que $\Phi_i = \sum_x r_i(x)$, portanto $\Phi_i \geq 0$ para todo i.

Mas... quanto vale Φ_0?

Inserções em splay trees

A inserção em splay trees é igual à inserção em ABB,
seguida de uma chamada a SPLAY no elemento inserido.

Seja y_1 o pai de x após a inserção, e
y_i o pai de y_{i-1} para $i = 1, \ldots, k$, onde y_k é a raiz da árvore.

Inserções em splay trees

A inserção em splay trees é igual à inserção em ABB, seguida de uma chamada a SPLAY no elemento inserido.

Seja y_1 o pai de x após a inserção, e
y_i o pai de y_{i-1} para $i = 1, \ldots, k$, onde y_k é a raiz da árvore.

Considere a variação do potencial causado pela inserção.

r: potencial local antes da inserção
r': potencial local depois da inserção

$$\Delta\Phi = \sum_{j=1}^{k} (r'(y_j) - r(y_j))$$

Inserções em splay trees

A inserção em splay trees é igual à inserção em ABB,
seguida de uma chamada a SPLAY no elemento inserido.

Seja y_1 o pai de x após a inserção, e
y_i o pai de y_{i-1} para $i = 1, \ldots, k$, onde y_k é a raiz da árvore.

Considere a variação do potencial causado pela inserção.

r: potencial local antes da inserção
r': potencial local depois da inserção

$$\Delta\Phi = \sum_{j=1}^{k}(r'(y_j) - r(y_j)) = \sum_{j=1}^{k}(\lg(s(y_j)+1) - \lg(s(y_j)))$$

Inserções em splay trees

A inserção em splay trees é igual à inserção em ABB, seguida de uma chamada a SPLAY no elemento inserido.

Seja y_1 o pai de x após a inserção, e
y_i o pai de y_{i-1} para $i = 1, \ldots, k$, onde y_k é a raiz da árvore.

Considere a variação do potencial causado pela inserção.

r: potencial local antes da inserção
r': potencial local depois da inserção

$$\Delta\Phi = \sum_{j=1}^{k}(r'(y_j) - r(y_j)) = \sum_{j=1}^{k}(\lg(s(y_j)+1) - \lg(s(y_j)))$$

$$= \sum_{j=1}^{k} \lg \frac{s(y_j)+1}{s(y_j)}$$

Inserções em splay trees

A inserção em splay trees é igual à inserção em ABB, seguida de uma chamada a SPLAY no elemento inserido.

Seja y_1 o pai de x após a inserção, e
y_i o pai de y_{i-1} para $i = 1, \ldots, k$, onde y_k é a raiz da árvore.

Considere a variação do potencial causado pela inserção.

r: potencial local antes da inserção
r': potencial local depois da inserção

$$\begin{aligned} \Delta\Phi &= \sum_{j=1}^{k} (r'(y_j) - r(y_j)) = \sum_{j=1}^{k} (\lg(s(y_j)+1) - \lg(s(y_j))) \\ &= \sum_{j=1}^{k} \lg \frac{s(y_j)+1}{s(y_j)} = \lg \left(\prod_{j=1}^{k} \frac{s(y_j)+1}{s(y_j)} \right) \end{aligned}$$

Inserções em splay trees

Seja y_1 o pai de x após a inserção, e y_i o pai de y_{i-1} para $i = 1, \ldots, k$, onde y_k é a raiz da árvore.

Considere a variação do potencial causado pela inserção.

r: potencial local antes da inserção
r': potencial local depois da inserção

$$\Delta\Phi \;=\; \sum_{j=1}^{k}(r'(y_j) - r(y_j)) \;=\; \lg\left(\prod_{j=1}^{k}\frac{s(y_j)+1}{s(y_j)}\right)$$

Inserções em splay trees

Seja y_1 o pai de x após a inserção, e y_i o pai de y_{i-1} para $i = 1, \ldots, k$, onde y_k é a raiz da árvore.

Considere a variação do potencial causado pela inserção.

r: potencial local antes da inserção
r': potencial local depois da inserção

$$\begin{aligned}
\Delta\Phi &= \sum_{j=1}^{k} (r'(y_j) - r(y_j)) = \lg \left(\prod_{j=1}^{k} \frac{s(y_j) + 1}{s(y_j)} \right) \\
&\leq \lg \left(\frac{s(y_2)}{s(y_1)} \cdot \frac{s(y_3)}{s(y_2)} \cdots \frac{s(y_k)}{s(y_{k-1})} \cdot \frac{s(y_k)+1}{s(y_k)} \right) \quad (\text{pois } s(y_j)+1 \leq s(y_{j+1}))
\end{aligned}$$

Inserções em splay trees

Seja y_1 o pai de x após a inserção, e y_i o pai de y_{i-1} para $i = 1, \ldots, k$, onde y_k é a raiz da árvore.

Considere a variação do potencial causado pela inserção.

r: potencial local antes da inserção
r': potencial local depois da inserção

$$\begin{aligned}
\Delta\Phi &= \sum_{j=1}^{k}(r'(y_j) - r(y_j)) = \lg\left(\prod_{j=1}^{k}\frac{s(y_j)+1}{s(y_j)}\right) \\
&\leq \lg\left(\frac{s(y_2)}{s(y_1)}\cdot\frac{s(y_3)}{s(y_2)}\cdots\frac{s(y_k)}{s(y_{k-1})}\cdot\frac{s(y_k)+1}{s(y_k)}\right) \quad (\text{pois } s(y_j)+1 \leq s(y_{j+1})) \\
&= \lg\left(\frac{s(y_k)+1}{s(y_1)}\right)
\end{aligned}$$

Inserções em splay trees

Seja y_1 o pai de x após a inserção, e y_i o pai de y_{i-1} para $i = 1, \ldots, k$, onde y_k é a raiz da árvore.

Considere a variação do potencial causado pela inserção.

r: potencial local antes da inserção
r': potencial local depois da inserção

$$\begin{aligned}
\Delta\Phi &= \sum_{j=1}^{k}(r'(y_j) - r(y_j)) = \lg\left(\prod_{j=1}^{k}\frac{s(y_j)+1}{s(y_j)}\right) \\
&\leq \lg\left(\frac{s(y_2)}{s(y_1)}\cdot\frac{s(y_3)}{s(y_2)}\cdots\frac{s(y_k)}{s(y_{k-1})}\cdot\frac{s(y_k)+1}{s(y_k)}\right) \quad (\text{pois } s(y_j)+1 \leq s(y_{j+1})) \\
&= \lg\left(\frac{s(y_k)+1}{s(y_1)}\right) \\
&\leq \lg(n+1).
\end{aligned}$$

Inserções em splay trees

Seja y_1 o pai de x após a inserção, e y_i o pai de y_{i-1} para $i = 1, \ldots, k$, onde y_k é a raiz da árvore.

Considere a variação do potencial causado pela inserção.

r: potencial local antes da inserção
r': potencial local depois da inserção

Mostramos que $\Delta\Phi \leq \lg(n+1) \leq \lg n + 1$.

Inserções em splay trees

Seja y_1 o pai de x após a inserção, e y_i o pai de y_{i-1} para $i = 1, \ldots, k$, onde y_k é a raiz da árvore.

Considere a variação do potencial causado pela inserção.

r: potencial local antes da inserção
r': potencial local depois da inserção

Mostramos que $\Delta\Phi \leq \lg(n+1) \leq \lg n + 1$.

Então $\hat{c} \leq 0 + \lg n + 1 = \lg n + 1$. (inserção não faz rotações)

Inserções em splay trees

Seja y_1 o pai de x após a inserção, e y_i o pai de y_{i-1} para $i = 1, \ldots, k$, onde y_k é a raiz da árvore.

Considere a variação do potencial causado pela inserção.

r: potencial local antes da inserção
r': potencial local depois da inserção

Mostramos que $\Delta\Phi \leq \lg(n+1) \leq \lg n + 1$.

Então $\hat{c} \leq 0 + \lg n + 1 = \lg n + 1$. (inserção não faz rotações)

Ou seja, tanto para o SPLAY quanto para a inserção, vale que

$$\hat{c} \leq 3 \lg n + 1.$$

Concluindo a análise

Lembre-se que $\Phi_i \geq 0$ para todo i e agora $\Phi_0 = 0$.
(Começamos da árvore vazia.)

Concluindo a análise

Lembre-se que $\Phi_i \geq 0$ para todo i e agora $\Phi_0 = 0$.
(Começamos da árvore vazia.)

Então o custo total de m operações (inserções e buscas) é

$$\begin{aligned}
\sum_{i=1}^{m} c_i &= \sum_{i=1}^{m} \hat{c}_i - \Phi_m + \Phi_0 \\
&\leq \sum_{i=1}^{m} (3 \lg n_i + 1) - \Phi_m + \Phi_0 \\
&\leq 3m \lg m + m - 0 \\
&\leq 4m \lg m.
\end{aligned}$$

Concluindo a análise

Lembre-se que $\Phi_i \geq 0$ para todo i e agora $\Phi_0 = 0$.
(Começamos da árvore vazia.)

Então o custo total de m operações (inserções e buscas) é

$$\begin{aligned}
\sum_{i=1}^{m} c_i &= \sum_{i=1}^{m} \hat{c}_i - \Phi_m + \Phi_0 \\
&\leq \sum_{i=1}^{m} (3 \lg n_i + 1) - \Phi_m + \Phi_0 \\
&\leq 3m \lg m + m - 0 \\
&\leq 4m \lg m.
\end{aligned}$$

Portanto o custo amortizado por operação é $\mathrm{O}(\lg m)$.

Análise do Union-Find

CLRS cap 21

Algoritmo de Kruskal

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

```
KRUSKAL (G, c)
1  A ← ∅
2  sejam e_1, ..., e_m as arestas de G ordenadas por c
3  para cada u ∈ V(G) faça MakeSet(u)
4  para i ← 1 até m faça
5      sejam u e v as pontas de e_i
6      se FindSet(u) ≠ FindSet(v)
7          então A ← A ∪ {e_i}
8                Union(u, v)
9  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)
1 $\quad \text{pai}[x] \leftarrow x$

Implementação 1 do union-find

```
Make-Set (x)
1   pai[x] ← x

Find (x)
1   r ← x
2   enquanto pai[r] ≠ r faça
3       r ← pai[r]
4   devolva r
```

Implementação 1 do union-find

```
Make-Set (x)
1   pai[x] ← x

Find (x)
1   r ← x
2   enquanto pai[r] ≠ r faça
3       r ← pai[r]
4   devolva r

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

Implementação 1 do union-find

```
Make-Set (x)
1   pai[x] ← x

Find (x)
1   r ← x
2   enquanto pai[r] ≠ r faça
3       r ← pai[r]
4   devolva r

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

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

Implementação 2

Heurística dos tamanhos

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

Implementação 2

Heurística dos tamanhos

Make-Set (x)

1 $\quad \text{pai}[x] \leftarrow x$

2 $\quad \text{rank}[x] \leftarrow 0$ $\quad \triangleright$ altura da árvore

Find (x): o mesmo de antes

Implementação 2

Heurística dos tamanhos

Make-Set (x)

```
1  pai[x] ← x
2  rank[x] ← 0                                  ▷ altura da árvore
```

Find (x): o mesmo de antes

Union (x, y) ▷ x e y representantes distintos

```
1  se rank[x] ≥ rank[y]
2      então pai[y] ← x
3            se rank[x] = rank[y]
4                então rank[x] ← rank[x] + 1
5      senão pai[x] ← y
```

Implementação 2

Heurística dos tamanhos

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

Find (x): o mesmo de antes

Union (x, y) $\quad \triangleright$ x e y representantes distintos
1 $\quad$ **se** $\text{rank}[x] \geq \text{rank}[y]$
2 $\quad\quad$ **então** $\text{pai}[y] \leftarrow x$
3 $\quad\quad\quad$ **se** $\text{rank}[x] = \text{rank}[y]$
4 $\quad\quad\quad\quad$ **então** $\text{rank}[x] \leftarrow \text{rank}[x] + 1$
5 $\quad\quad$ **senão** $\text{pai}[x] \leftarrow y$

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

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:

$$O(\lg^* n),$$

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:

$$O(\lg^* n),$$

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)
1   pai[x] ← x
2   rank[x] ← 0

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

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

Union-Find

```
Union (x, y)
1   x' ← Find (x)
2   y' ← Find (y)
3   se x' ≠ y'
4       então Link (x', y')
```

```
Link (x, y)                                  ▷ x e y representantes distintos
1   se rank[x] ≥ rank[y]
2       então  pai[y] ← x
3              se rank[x] = rank[y]
4                  então rank[x] ← rank[x] + 1
5       senão pai[x] ← y
```

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?

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.
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 $\mathrm{E}[X] < 2$ 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 $\mathrm{E}[X] < 2$ 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 $\mathrm{E}[X] < 2$ 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 $\mathrm{E}[X] < 2$ 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 $\mathrm{E}[X] < 2$ 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 $\Pr\{X_s = 1\} \leq \dfrac{1}{n}$ 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 $\mathrm{E}[X] < 2$ 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

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

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 $s = r \bmod n$.

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.