

Sammengisleit

Bergur Snorrason

17. febrúar 2023

Sammengisleit

- ▶ *Sammengisleit* (e. *union-find*) er öflug leið til að halda utan um jafngildisflokka tiltekna vensla eða, með öðrum orðum, halda utan um *sundurlæg* mengi.

Sammengisleit

- ▶ *Sammengisleit* (e. *union-find*) er öflug leið til að halda utan um jafngildisflokka tiltekna vensla eða, með öðrum orðum, halda utan um *sundurlæg* mengi.
- ▶ Við viljum:

Sammengisleit

- ▶ *Sammengisleit* (e. *union-find*) er öflug leið til að halda utan um jafngildisflokka tiltekna vensla eða, með öðrum orðum, halda utan um *sundurlæg* mengi.
- ▶ Við viljum:
 - ▶ Bera saman jafngildisflokka mismunandi staka.

Sammengisleit

- ▶ *Sammengisleit* (e. *union-find*) er öflug leið til að halda utan um jafngildisflokka tiltekna vensla eða, með öðrum orðum, halda utan um *sundurlæg* mengi.
- ▶ Við viljum:
 - ▶ Bera saman jafngildisflokka mismunandi staka.
 - ▶ Sameina jafngildisflokka.

Sammengisleit

- ▶ *Sammengisleit* (e. *union-find*) er öflug leið til að halda utan um jafngildisflokka tiltekna vensla eða, með öðrum orðum, halda utan um *sundurlæg* mengi.
- ▶ Við viljum:
 - ▶ Bera saman jafngildisflokka mismunandi staka.
 - ▶ Sameina jafngildisflokka.
- ▶ Við tölum um aðgerðirnar `find(x)` og `join(x, y)`.

- Tökum sem dæmi einstökungasafnið $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$.

- ▶ Tökum sem dæmi einstökungasafnið $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$.
- ▶ `join(1, 3)` gefur okkur $\{\{1, 3\}, \{2\}, \{4\}, \{5\}\}$.

- ▶ Tökum sem dæmi einstökungasafnið $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$.
- ▶ `join(1, 3)` gefur okkur $\{\{1, 3\}, \{2\}, \{4\}, \{5\}\}$.
- ▶ `join(2, 5)` gefur okkur $\{\{1, 3\}, \{2, 5\}, \{4\}\}$.

- ▶ Tökum sem dæmi einstökungasafnið $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$.
- ▶ `join(1, 3)` gefur okkur $\{\{1, 3\}, \{2\}, \{4\}, \{5\}\}$.
- ▶ `join(2, 5)` gefur okkur $\{\{1, 3\}, \{2, 5\}, \{4\}\}$.
- ▶ `join(2, 4)` gefur okkur $\{\{1, 3\}, \{2, 4, 5\}\}$.

- ▶ Tökum sem dæmi einstökungasafnið $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$.
- ▶ `join(1, 3)` gefur okkur $\{\{1, 3\}, \{2\}, \{4\}, \{5\}\}$.
- ▶ `join(2, 5)` gefur okkur $\{\{1, 3\}, \{2, 5\}, \{4\}\}$.
- ▶ `join(2, 4)` gefur okkur $\{\{1, 3\}, \{2, 4, 5\}\}$.
- ▶ `join(1, 4)` gefur okkur $\{\{1, 2, 3, 4, 5\}\}$.

- ▶ Tökum sem dæmi einstökungasafnið $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$.
- ▶ `join(1, 3)` gefur okkur $\{\{1, 3\}, \{2\}, \{4\}, \{5\}\}$.
- ▶ `join(2, 5)` gefur okkur $\{\{1, 3\}, \{2, 5\}, \{4\}\}$.
- ▶ `join(2, 4)` gefur okkur $\{\{1, 3\}, \{2, 4, 5\}\}$.
- ▶ `join(1, 4)` gefur okkur $\{\{1, 2, 3, 4, 5\}\}$.
- ▶ Á sérhverjum tímapunkti myndi `find(x)` skila einhverju staki sem er í sama mengi og `x`.

- ▶ Tökum sem dæmi einstökungasafnið $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$.
- ▶ `join(1, 3)` gefur okkur $\{\{1, 3\}, \{2\}, \{4\}, \{5\}\}$.
- ▶ `join(2, 5)` gefur okkur $\{\{1, 3\}, \{2, 5\}, \{4\}\}$.
- ▶ `join(2, 4)` gefur okkur $\{\{1, 3\}, \{2, 4, 5\}\}$.
- ▶ `join(1, 4)` gefur okkur $\{\{1, 2, 3, 4, 5\}\}$.
- ▶ Á sérhverjum tímapunkti myndi `find(x)` skila einhverju staki sem er í sama mengi og `x`.
- ▶ Mikilvægt er að `find(...)` skilar sama stakinu fyrir sérhvert stak í sérhverjum jafngildisflokki.

- ▶ Tökum sem dæmi einstökungasafnið $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$.
- ▶ `join(1, 3)` gefur okkur $\{\{1, 3\}, \{2\}, \{4\}, \{5\}\}$.
- ▶ `join(2, 5)` gefur okkur $\{\{1, 3\}, \{2, 5\}, \{4\}\}$.
- ▶ `join(2, 4)` gefur okkur $\{\{1, 3\}, \{2, 4, 5\}\}$.
- ▶ `join(1, 4)` gefur okkur $\{\{1, 2, 3, 4, 5\}\}$.
- ▶ Á sérhverjum tímapunkti myndi `find(x)` skila einhverju staki sem er í sama mengi og `x`.
- ▶ Mikilvægt er að `find(...)` skilar sama stakinu fyrir sérhvert stak í sérhverjum jafngildisflokki.
- ▶ Til dæmis, í þriðja punktinum myndi `find(1)` og `find(3)` þurfa að skila sama stakinu.

- ▶ Tökum sem dæmi einstökungasafnið $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$.
- ▶ `join(1, 3)` gefur okkur $\{\{1, 3\}, \{2\}, \{4\}, \{5\}\}$.
- ▶ `join(2, 5)` gefur okkur $\{\{1, 3\}, \{2, 5\}, \{4\}\}$.
- ▶ `join(2, 4)` gefur okkur $\{\{1, 3\}, \{2, 4, 5\}\}$.
- ▶ `join(1, 4)` gefur okkur $\{\{1, 2, 3, 4, 5\}\}$.
- ▶ Á sérhverjum tímapunkti myndi `find(x)` skila einhverju staki sem er í sama mengi og `x`.
- ▶ Mikilvægt er að `find(...)` skilar sama stakinu fyrir sérhvert stak í sérhverjum jafngildisflokki.
- ▶ Til dæmis, í þriðja punktinum myndi `find(1)` og `find(3)` þurfa að skila sama stakinu.
- ▶ Við köllum þetta stak *oddvita* (e. *representative*) jafngildisflokksins.

- Gerum ráð fyrir að tölurnar sem við munum vinna með séu jákvæðar og minni en n .

- ▶ Gerum ráð fyrir að tölurnar sem við munum vinna með séu jákvæðar og minni en n .
- ▶ Við munum þá gefa okkur n staka fylki p , þar sem i -ta stakið í fylkinu er upphafstillt sem i .

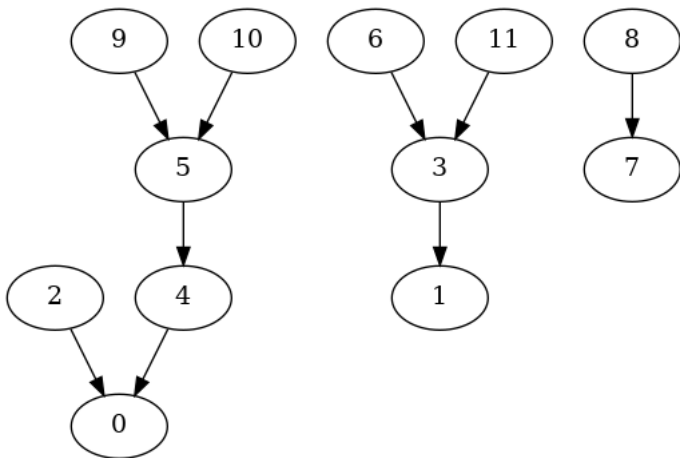
- ▶ Gerum ráð fyrir að tölurnar sem við munum vinna með séu jákvæðar og minni en n .
- ▶ Við munum þá gefa okkur n staka fylki p , þar sem i -ta stakið í fylkinu er upphafstillt sem i .
- ▶ Fylkið p mun nú geyma *foreldri* sérhvers stak.

- ▶ Gerum ráð fyrir að tölurnar sem við munum vinna með séu jákvæðar og minni en n .
- ▶ Við munum þá gefa okkur n staka fylki p , þar sem i -ta stakið í fylkinu er upphafstillt sem i .
- ▶ Fylkið p mun nú geyma *foreldri* sérhvers stak.
- ▶ Foreldrin mynda keðjur, sem svara til jafngildisflokka.

- ▶ Gerum ráð fyrir að tölurnar sem við munum vinna með séu jákvæðar og minni en n .
- ▶ Við munum þá gefa okkur n staka fylki p , þar sem i -ta stakið í fylkinu er upphafstillt sem i .
- ▶ Fylkið p mun nú geyma *foreldri* sérhvers stak.
- ▶ Foreldrin mynda keðjur, sem svara til jafngildisflokka.
- ▶ Sérhver keðja endar í einhverju staki, sem munu vera oddvitar jafngildisflokka.

- Keðjurnar sem fást með $\{\{0, 2, 4, 5, 9, 10\}, \{1, 3, 6, 11\}, \{7, 8\}\}$ gætu til dæmis verið gefnar með $p = [0, 1, 0, 1, 0, 4, 3, 7, 7, 5, 5, 3]$.

- Keðjurnar sem fást með $\{\{0, 2, 4, 5, 9, 10\}, \{1, 3, 6, 11\}, \{7, 8\}\}$ gætu til dæmis verið gefnar með $p = [0, 1, 0, 1, 0, 4, 3, 7, 7, 5, 5, 3]$.



- ▶ Til að fá oddvita flokks tiltekings staks er hægt að fara endurkvæmt upp keðjuna.

- ▶ Til að fá oddvita flokks tiltekings staks er hægt að fara endurkvæmt upp keðjuna.
- ▶ Til að sameina flokka nægir að breyta foreldri oddvita annars flokksins yfir í eitthvert stak hins flokksins.

- ▶ Til að fá oddvita flokks tiltekins staks er hægt að fara endurkvæmt upp keðjuna.
- ▶ Til að sameina flokka nægir að breyta foreldri oddvita annars flokksins yfir í eitthvert stak hins flokksins.
- ▶ Báðar þessar aðgerðir er auðvelt að útfæra.

Frumstæð sammengisleit

```
4 int p[MAXN];
5 int find(int x)
6 {
7     if (p[x] == x) return x;
8     return find(p[x]);
9 }
10
11 void join(int x, int y)
12 {
13     p[find(x)] = find(y);
14 }
15
16 void init(int n)
17 {
18     for (int i = 0; i < n; i++) p[i] = i;
19 }
```

Ekki nota frumstæða sammengisleit

- ▶ Við sjáum nú að tímaflækja `find(...)` er línuleg í lengd keðjunnar, svo þar sem lengd keðjunnar getur verið í versta falli n þá er `find(...)` með tímaflækjuna $\mathcal{O}()$.

Ekki nota frumstæða sammengisleit

- ▶ Við sjáum nú að tímaflækja `find(...)` er línuleg í lengd keðjunnar, svo þar sem lengd keðjunnar getur verið í versta falli n þá er `find(...)` með tímaflækjuna $\mathcal{O}(n)$.

Ekki nota frumstæða sammengisleit

- ▶ Við sjáum nú að tímaflækja `find(...)` er línuleg í lengd keðjunnar, svo þar sem lengd keðjunnar getur verið í versta falli n þá er `find(...)` með tímaflækjuna $\mathcal{O}(n)$.
- ▶ Fallið `join(...)` gerir lítið annað en að kalla tvisvar á `find(...)` svo það er $\mathcal{O}(\quad)$.

Ekki nota frumstæða sammengisleit

- ▶ Við sjáum nú að tímaflækja `find(...)` er línuleg í lengd keðjunnar, svo þar sem lengd keðjunnar getur verið í versta falli n þá er `find(...)` með tímaflækjuna $\mathcal{O}(n)$.
- ▶ Fallið `join(...)` gerir lítið annað en að kalla tvisvar á `find(...)` svo það er $\mathcal{O}(n)$.

Ekki nota frumstæða sammengisleit

- ▶ Við sjáum nú að tímaflækja `find(...)` er línuleg í lengd keðjunnar, svo þar sem lengd keðjunnar getur verið í versta falli n þá er `find(...)` með tímaflækjuna $\mathcal{O}(n)$.
- ▶ Fallið `join(...)` gerir lítið annað en að kalla tvisvar á `find(...)` svo það er $\mathcal{O}(n)$.
- ▶ Við myndum því aðeins ná að svara n fyrirpsurnum ef $n \leq 10^4$.

Ekki nota frumstæða sammengisleit

- ▶ Við sjáum nú að tímaflækja `find(...)` er línuleg í lengd keðjunnar, svo þar sem lengd keðjunnar getur verið í versta falli n þá er `find(...)` með tímaflækjuna $\mathcal{O}(n)$.
- ▶ Fallið `join(...)` gerir lítið annað en að kalla tvisvar á `find(...)` svo það er $\mathcal{O}(n)$.
- ▶ Við myndum því aðeins ná að svara n fyrirpsurnum ef $n \leq 10^4$.
- ▶ Því er ekki ráðlagt að nota þessa frumstæðu útfærslu.

Ekki nota frumstæða sammengisleit

- ▶ Við sjáum nú að tímaflækja `find(...)` er línuleg í lengd keðjunnar, svo þar sem lengd keðjunnar getur verið í versta falli n þá er `find(...)` með tímaflækjuna $\mathcal{O}(n)$.
- ▶ Fallið `join(...)` gerir lítið annað en að kalla tvisvar á `find(...)` svo það er $\mathcal{O}(n)$.
- ▶ Við myndum því aðeins ná að svara n fyrirpsurnum ef $n \leq 10^4$.
- ▶ Því er ekki ráðlagt að nota þessa frumstæðu útfærslu.
- ▶ Hana má þó bæta.

Keðjuþjöppuð sammengisleit

- ▶ Lykilatriðið í bætingunni er að stytta keðjurnar.

Keðjubjöppuð sammengisleit

- ▶ Lykilatriðið í bætingunni er að stytta keðjurnar.
- ▶ Í hvert sinn sem við köllum á `find(...)` þá fletjum við keðjun sem við heimsækjum.

Keðjuþjöppuð sammengisleit

- ▶ Lykilatriðið í bætingunni er að stytta keðjurnar.
- ▶ Í hvert sinn sem við köllum á `find(...)` þá fletjum við keðjun sem við heimsækjum.
- ▶ Þetta er gert með því að setja `p[x]` sem oddvita flokks `x`, í hverju skrefi endurkvæmninnar.

Keðjubjöppuð sammengisleit

- ▶ Lykilatriðið í bætingunni er að stytta keðjurnar.
- ▶ Í hvert sinn sem við köllum á `find(...)` þá fletjum við keðjun sem við heimsækjum.
- ▶ Þetta er gert með því að setja `p[x]` sem oddvita flokks `x`, í hverju skrefi endurkvæmninnar.
- ▶ Þetta köllum við *keðjubjöppun* (e. *path compression*).

- ▶ Gefum okkur $p = [0, 0, 1, 2, 3, 4, 5, 6, 7]$.

- ▶ Gefum okkur $p = [0, 0, 1, 2, 3, 4, 5, 6, 7]$.
- ▶ Ljóst er að `find(5)` skilar 0.

- ▶ Gefum okkur $p = [0, 0, 1, 2, 3, 4, 5, 6, 7]$.
- ▶ Ljóst er að `find(5)` skilar 0.
- ▶ Ef við notum frumstæða sammengisleit breytist p ekki neitt þegar kallað er á `find(...)` en með keðjuþjappaðri sammengisleit þjappast keðjan frá og með 5 og því fæst $p = [0, 0, 0, 0, 0, 0, 5, 6, 7]$.

- ▶ Gefum okkur $p = [0, 0, 1, 2, 3, 4, 5, 6, 7]$.
- ▶ Ljóst er að `find(5)` skilar 0.
- ▶ Ef við notum frumstæða sammengisleit breytist p ekki neitt þegar kallað er á `find(...)` en með keðjuþjappaðri sammengisleit þjappast keðjan frá og með 5 og því fæst $p = [0, 0, 0, 0, 0, 0, 5, 6, 7]$.
- ▶ Takið eftir að nú er líka styttra í oddvitann fyrir stök 6, 7 og 8, þó við heimsóttum þau ekki í endurkvæmninni.

Keðjubjöppað sammengisleit

```
4 int p[MAXN];
5 int find(int x)
6 {
7     if (p[x] == x) return x;
8     return p[x] = find(p[x]);
9 }
10
11 void join(int x, int y)
12 {
13     p[find(x)] = find(y);
14 }
15
16 void init(int n)
17 {
18     for (int i = 0; i < n; i++) p[i] = i;
19 }
```

- ▶ Þetta er þó ekki eina bætingin sem er í boði.

- ▶ Þetta er þó ekki eina bætingin sem er í boði.
- ▶ Við getum einnig sameinað á skilvirkari máta með *stærðarmiðaðri sameiningu*.

Stærðarmiðuð sameining (e. *union by size*)

- Þegar við sameinum keðjur þarf að velja hvor odvitinn verður ennþá odviti.

Stærðarmiðuð sameining (e. *union by size*)

- ▶ Þegar við sameinum keðjur þarf að velja hvor oddvitinn verður ennþá odviti.
- ▶ Við getum valið oddvitann sem hefur fleiri stök í sinni keðju.

Stærðarmiðuð sameining (e. union by size)

- ▶ Þegar við sameinum keðjur þarf að velja hvor oddvitinn verður ennþá odviti.
- ▶ Við getum valið oddvitann sem hefur fleiri stök í sinni keðju.

```
4 int p[MAXN];
5 int find(int x)
6 {
7     return p[x] < 0 ? x : (p[x] = find(p[x]));
8 }
9
10 void join(int x, int y)
11 {
12     int rx = find(x), ry = find(y);
13     if (rx == ry) return;
14     if (p[rx] > p[ry]) p[ry] += p[rx], p[rx] = ry;
15     else p[rx] += p[ry], p[ry] = rx;
16 }
```

Stærðarmiðuð sameining (e. *union by size*)

- ▶ Þegar við sameinum keðjur þarf að velja hvor oddvitinn verður ennþá odviti.
- ▶ Við getum valið oddvitann sem hefur fleiri stök í sinni keðju.

```
4 int p[MAXN];
5 int find(int x)
6 {
7     return p[x] < 0 ? x : (p[x] = find(p[x]));
8 }
9
10 void join(int x, int y)
11 {
12     int rx = find(x), ry = find(y);
13     if (rx == ry) return;
14     if (p[rx] > p[ry]) p[ry] += p[rx], p[rx] = ry;
15     else p[rx] += p[ry], p[ry] = rx;
16 }
```

- ▶ Í þessari út færslu geymir oddvitinn neikvæða tölu, en önnur stök vísa ennþá upp keðjuna.

Stærðarmiðuð sameining (e. union by size)

- ▶ Þegar við sameinum keðjur þarf að velja hvor oddvitinn verður ennþá odviti.
- ▶ Við getum valið oddvitann sem hefur fleiri stök í sinni keðju.

```
4 int p[MAXN];
5 int find(int x)
6 {
7     return p[x] < 0 ? x : (p[x] = find(p[x]));
8 }
9
10 void join(int x, int y)
11 {
12     int rx = find(x), ry = find(y);
13     if (rx == ry) return;
14     if (p[rx] > p[ry]) p[ry] += p[rx], p[rx] = ry;
15     else p[rx] += p[ry], p[ry] = rx;
16 }
```

- ▶ Í þessari út færslu geymir oddvitinn neikvæða tölu, en önnur stök vísa ennþá upp keðjuna.
- ▶ Þessi neikvæða tala svarar til fjölda staka í þeim keðjum sem enda í oddvitanum.

Stærðarmiðuð sameining (e. union by size)

- ▶ Þegar við sameinum keðjur þarf að velja hvor oddvitinn verður ennþá odviti.
- ▶ Við getum valið oddvitann sem hefur fleiri stök í sinni keðju.

```
4 int p[MAXN];
5 int find(int x)
6 {
7     return p[x] < 0 ? x : (p[x] = find(p[x]));
8 }
9
10 void join(int x, int y)
11 {
12     int rx = find(x), ry = find(y);
13     if (rx == ry) return;
14     if (p[rx] > p[ry]) p[ry] += p[rx], p[rx] = ry;
15     else p[rx] += p[ry], p[ry] = rx;
16 }
```

- ▶ Í þessari út færslu geymir oddvitinn neikvæða tölu, en önnur stök vísa ennþá upp keðjuna.
- ▶ Þessi neikvæða tala svarar til fjölda staka í þeim keðjum sem enda í oddvitanum.
- ▶ Svo `-p[find(x)]` er fjöldi staka í jafngildisflokki `x`.

Keðjubjöppað og stærðarmiðuð sammengisleit

```
4 int p[MAXN];
5 int find(int x)
6 {
7     return p[x] < 0 ? x : (p[x] = find(p[x]));
8 }
9
10 void join(int x, int y)
11 {
12     int rx = find(x), ry = find(y);
13     if (rx == ry) return;
14     if (p[rx] > p[ry]) p[ry] += p[rx], p[rx] = ry;
15     else p[rx] += p[ry], p[ry] = rx;
16 }
17
18 void init(int n)
19 {
20     for (int i = 0; i < n; i++) p[i] = -1;
21 }
22
23 void size(int x)
24 {
25     return -p[find(x)];
26 }
```

- ▶ Það er flóknara að lýsa tímaflækju keðjuþjappaðrar sammengisleitar með stærðarmiðaðri sameiningu.

- ▶ Það er flóknara að lýsa tímaflækju keðjuþjappaðrar sammengisleitar með stærðarmiðaðri sameiningu.
- ▶ Á heildina litið (e. *amortized*) er tímaflækja hverrar aðgerðar $\mathcal{O}(\alpha(n))$, þar sem α er andhverfa Ackermann fallsins.

- ▶ Það er flóknara að lýsa tímaflækju keðjuþjappaðrar sammengisleitar með stærðarmiðaðri sameiningu.
- ▶ Á heildina litið (e. *amortized*) er tímaflækja hverrar aðgerðar $\mathcal{O}(\alpha(n))$, þar sem α er andhverfa Ackermann fallsins.
- ▶ Fyrir þau n sem við fáumst við er $\alpha(n)$ nánast fast.

- ▶ Það er flóknara að lýsa tímaflækju keðjuþjappaðrar sammengisleitar með stærðarmiðaðri sameiningu.
- ▶ Á heildina litið (e. *amortized*) er tímaflækja hverrar aðgerðar $\mathcal{O}(\alpha(n))$, þar sem α er andhverfa Ackermann fallsins.
- ▶ Fyrir þau n sem við fáumst við er $\alpha(n)$ nánast fast.
- ▶ Við ímyndum okkur því alltaf að hver aðgerð sammengisleitar sé $\mathcal{O}(1)$.

