Sammengisleit

Bergur Snorrason

15. febrúar 2021

Sammengisleit

- *Sammengisleit* (e. *union-find*) er öflug leið til að halda utan um jafngildisflokka tiltekna vensla, eða m.ö.o. halda utan um *sundurlæg* mengi.

Sammengisleit

- *Sammengisleit* (e. *union-find*) er öflug leið til að halda utan um jafngildisflokka tiltekna vensla, eða m.ö.o. halda utan um *sundurlæg* mengi.
- Við viljum:

Sammengisleit

- *Sammengisleit* (e. *union-find*) er öflug leið til að halda utan um jafngildisflokka tiltekna vensla, eða m.ö.o. 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 m.ö.o. 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 m.ö.o. 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 *ráðherra* (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.

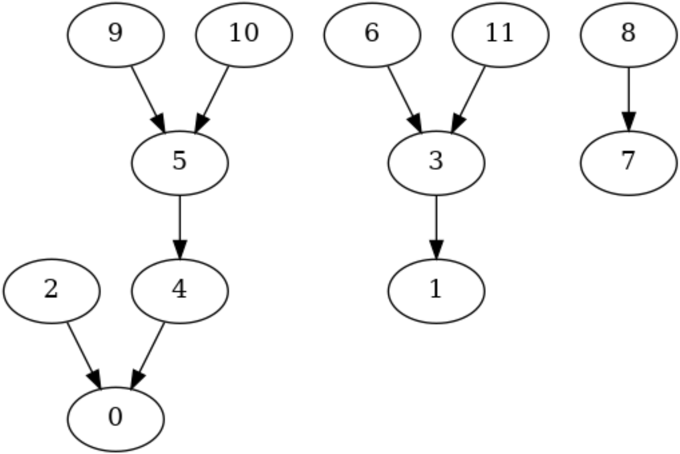
- Gerum ráð fyrir að tölurnar sem við munum vinna með séu jákvæðar og minni en n.
- Við munum þá gefa okkur n staka fylki p, þar sem i-ta stakið í fylkinu er upphafstillt sem i.
- Fylkið p mun nú geyma *foreldri* sérhvers stak.
- Foreldrin mynda keðjur.

- 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.
- Keðjurnar eru jafngildisflokkarnir.

- 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.
- Keðjurnar eru jafngildisflokkarnir.
- Sérhver keðja endar í einhverju staki, sem munu vera ráðherra jafngildisflokksins.

- 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 ráðherra flokks tiltekins staks er hægt að fara endurkvæmt upp keðjuna.

- Til að fá ráðherra flokks tiltekins staks er hægt að fara endurkvæmt upp keðjuna.
- Til að sameina flokka nægir að breyta foreldri ráðherra annars flokksins yfir í eitthvert stak hins flokksins (sér í lagi ráðherrann).

- Til að fá ráðherra flokks tiltekins staks er hægt að fara endurkvæmt upp keðjuna.
- Til að sameina flokka nægir að breyta foreldri ráðherra annars flokksins yfir í eitthvert stak hins flokksins (sér í lagi ráðherrann).
- Báðar þessar aðgerðir er auðvelt að útfæra.

Frumstæð sammengisleit

```
#define MAXN 1000000
int p[MAXN];

int find(int x)
{
    if (p[x] == x) return x;
    return find(p[x]);
}

void join(int x, int y)
{
    p[find(x)] = find(y);
}

int main()
{
    int i, n = MAXN;
    for (i = 0; i < n; i++) p[i] = i;
    ...
}
```

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 $\mathcal{O}(\quad)$.

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 $\mathcal{O}(n)$.

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 smækka keðjurnar.

Keðjuþjöppuð sammengisleit

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

Keðjuþjöppuð sammengisleit

- Lykilatriðið í bætingunni er að smækka 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 ráðherra flokks `x`, í hverju skrefi endurkvæmninnar.

Keðjuþjöppuð sammengisleit

- Lykilatriðið í bætingunni er að smækka 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 ráðherra flokks `x`, í hverju skrefi endurkvæmninnar.
- Þetta köllum við *keðjuþjöppun* (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 ráðherrann fyrir stök 6, 7 og 8, þó við heimsóttum þau ekki í endurkvæmninni.

Keðjuþjöppað sammengisleit

```
#define MAXN 1000000
int p[MAXN];

int find(int x)
{
    if (p[x] == x) return x;
    return p[x] = find(p[x]);
}

void join(int x, int y)
{
    p[find(x)] = find(y);
}

int main()
{
    int i, n = MAXN;
    for (i = 0; i < n; i++) p[i] = i;
    ...
}
```

- Það er flóknara að lýsa tímaflækju keðjuþjappaðrar sammengisleitar.

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

- Það er flóknara að lýsa tímaflækju keðjuþjappaðrar sammengisleitar.
- *Á heildina litið* (e. *amortized*) er tímaflækjan er $\mathcal{O}(\alpha(n))$, 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.
- *Á heildina litið* (e. *amortized*) er tímaflækjan er $\mathcal{O}(\alpha(n))$, 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 sammengisleit hafi tímaflækju $\mathcal{O}(\ n\ )$.

Stærðarmiðuð sameining (e. *union by size*)

- Þegar við sameinum keðjur þarf að velja hvor ráðherrann verður ennþá ráðherra.

Stærðarmiðuð sameining (e. *union by size*)

- Þegar við sameinum keðjur þarf að velja hvor ráðherrann verður ennþá ráðherra.
- Við getum þá valið ráðherrann sem hefur fleiri stök í sinni keðju.

Stærðarmiðuð sameining (e. *union by size*)

- Þegar við sameinum keðjur þarf að velja hvor ráðherrann verður ennþá ráðherra.
- Við getum þá valið ráðherrann sem hefur fleiri stök í sinni keðju.

```
int p[MAXN]; // = [-1, -1, ..., -1]
int find(int x)
{
    return p[x] < 0 ? x : (p[x] = find(p[x]));
}
void join(int x, int y)
{
    int rx = find(x), ry = find(y);
    if (rx == ry) return;
    if (p[rx] > p[ry]) p[ry] += p[rx], p[rx] = ry;
    else p[rx] += p[ry], p[ry] = rx;
}
```

Stærðarmiðuð sameining (e. *union by size*)

- Þegar við sameinum keðjur þarf að velja hvor ráðherrann verður ennþá ráðherra.
- Við getum þá valið ráðherrann sem hefur fleiri stök í sinni keðju.

```
int p[MAXN]; // = [-1, -1, ..., -1]
int find(int x)
{
    return p[x] < 0 ? x : (p[x] = find(p[x]));
}
void join(int x, int y)
{
    int rx = find(x), ry = find(y);
    if (rx == ry) return;
    if (p[rx] > p[ry]) p[ry] += p[rx], p[rx] = ry;
    else p[rx] += p[ry], p[ry] = rx;
}
```

- Í þessari út færlsu geymir ráðherrann neikvæða tölu, en önnur stök vísa ennþá upp keðjuna.

Stærðarmiðuð sameining (e. *union by size*)

- Þegar við sameinum keðjur þarf að velja hvor ráðherrann verður ennþá ráðherra.
- Við getum þá valið ráðherrann sem hefur fleiri stök í sinni keðju.

```
int p[MAXN]; // = [-1, -1, ..., -1]
int find(int x)
{
    return p[x] < 0 ? x : (p[x] = find(p[x]));
}
void join(int x, int y)
{
    int rx = find(x), ry = find(y);
    if (rx == ry) return;
    if (p[rx] > p[ry]) p[ry] += p[rx], p[rx] = ry;
    else p[rx] += p[ry], p[ry] = rx;
}
```

- Í þessari út færlsu geymir ráðherrann neikvæða tölu, en önnur stök vísa ennþá upp keðjuna.
- Þessi tala svarar til fjölda staka í þeim keðjum sem enda í ráðherranum.

Stærðarmiðuð sameining (e. *union by size*)

- Þegar við sameinum keðjur þarf að velja hvor ráðherrann verður ennþá ráðherra.
- Við getum þá valið ráðherrann sem hefur fleiri stök í sinni keðju.

```
int p[MAXN]; // = [-1, -1, ..., -1]
int find(int x)
{
    return p[x] < 0 ? x : (p[x] = find(p[x]));
}
void join(int x, int y)
{
    int rx = find(x), ry = find(y);
    if (rx == ry) return;
    if (p[rx] > p[ry]) p[ry] += p[rx], p[rx] = ry;
    else p[rx] += p[ry], p[ry] = rx;
}
```

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