

Biltré

Bergur Snorrason

12. febrúar 2021

- ▶ Gefinn er listi með n tölum.

- ▶ Gefinn er listi með n tölum.
- ▶ Næst koma q fyrirspurnir, þar sem hver er af einni af tveimur gerðum:

- ▶ Gefinn er listi með n tölum.
- ▶ Næst koma q fyrirspurnir, þar sem hver er af einni af tveimur gerðum:
 - ▶ Bættu k við i -tu töluna.

- ▶ Gefinn er listi með n tölum.
- ▶ Næst koma q fyrirspurnir, þar sem hver er af einni af tveimur gerðum:
 - ▶ Bættu k við i -tu töluna.
 - ▶ Reiknaðu summu allra talna á bilinu $[i, j]$.

- ▶ Gefinn er listi með n tölum.
- ▶ Næst koma q fyrirspurnir, þar sem hver er af einni af tveimur gerðum:
 - ▶ Bættu k við i -tu töluna.
 - ▶ Reiknaðu summu allra talna á bilinu $[i, j]$.
- ▶ Einföld útfærsla á þessum fyrirspurnum gefur okkur $\mathcal{O}(\quad)$ fyrir þá fyrri og $\mathcal{O}(\quad)$ fyrir þá seinni.

- ▶ Gefinn er listi með n tölum.
- ▶ Næst koma q fyrirspurnir, þar sem hver er af einni af tveimur gerðum:
 - ▶ Bættu k við i -tu töluna.
 - ▶ Reiknaðu summu allra talna á bilinu $[i, j]$.
- ▶ Einföld útfærsla á þessum fyrirspurnum gefur okkur $\mathcal{O}(1)$ fyrir þá fyrri og $\mathcal{O}()$ fyrir þá seinni.

- ▶ Gefinn er listi með n tölum.
- ▶ Næst koma q fyrirspurnir, þar sem hver er af einni af tveimur gerðum:
 - ▶ Bættu k við i -tu töluna.
 - ▶ Reiknaðu summu allra talna á bilinu $[i, j]$.
- ▶ Einföld útfærsla á þessum fyrirspurnum gefur okkur $\mathcal{O}(1)$ fyrir þá fyrri og $\mathcal{O}(n)$ fyrir þá seinni.

- ▶ Gefinn er listi með n tölum.
- ▶ Næst koma q fyrirspurnir, þar sem hver er af einni af tveimur gerðum:
 - ▶ Bættu k við i -tu töluna.
 - ▶ Reiknaðu summu allra talna á bilinu $[i, j]$.
- ▶ Einföld útfærsla á þessum fyrirspurnum gefur okkur $\mathcal{O}(1)$ fyrir þá fyrri og $\mathcal{O}(n)$ fyrir þá seinni.
- ▶ Þar sem allar (eða langflestar) fyrirspurnir gætu verið af seinni gerðin yrði lausnin í heildin $\mathcal{O}(\quad)$.

- ▶ Gefinn er listi með n tölum.
- ▶ Næst koma q fyrirspurnir, þar sem hver er af einni af tveimur gerðum:
 - ▶ Bættu k við i -tu töluna.
 - ▶ Reiknaðu summu allra talna á bilinu $[i, j]$.
- ▶ Einföld útfærsla á þessum fyrirspurnum gefur okkur $\mathcal{O}(1)$ fyrir þá fyrri og $\mathcal{O}(n)$ fyrir þá seinni.
- ▶ Þar sem allar (eða langflestar) fyrirspurnir gætu verið af seinni gerðin yrði lausnin í heildin $\mathcal{O}(qn)$.

- ▶ Gefinn er listi með n tölum.
- ▶ Næst koma q fyrirspurnir, þar sem hver er af einni af tveimur gerðum:
 - ▶ Bættu k við i -tu töluna.
 - ▶ Reiknaðu summu allra talna á bilinu $[i, j]$.
- ▶ Einföld útfærsla á þessum fyrirspurnum gefur okkur $\mathcal{O}(1)$ fyrir þá fyrri og $\mathcal{O}(n)$ fyrir þá seinni.
- ▶ Þar sem allar (eða langflestar) fyrirspurnir gætu verið af seinni gerðin yrði lausnin í heildin $\mathcal{O}(qn)$.
- ▶ Það er þó hægt að leysa þetta dæmi hraðar.

- ▶ Gefinn er listi með n tölum.
- ▶ Næst koma q fyrirspurnir, þar sem hver er af einni af tveimur gerðum:
 - ▶ Bættu k við i -tu töluna.
 - ▶ Reiknaðu summu allra talna á bilinu $[i, j]$.
- ▶ Einföld útfærsla á þessum fyrirspurnum gefur okkur $\mathcal{O}(1)$ fyrir þá fyrri og $\mathcal{O}(n)$ fyrir þá seinni.
- ▶ Þar sem allar (eða langflestar) fyrirspurnir gætu verið af seinni gerðin yrði lausnin í heildin $\mathcal{O}(qn)$.
- ▶ Það er þó hægt að leysa þetta dæmi hraðar.
- ▶ Algengt er að nota til þess *biltré* (e. *segment tree*).

Biltré

- ▶ Biltré er tvíundartré sem geymir svör við vissum fyrirspurnum af seinni gerðinni.

Bilré

- ▶ Bilré er tvíundartré sem geymir svör við vissum fyrirspurnum af seinni gerðinni.
- ▶ Rótin geymir svar við fyrirspurninni $1 \dots n$ og ef nóða geymir svarið við $i \dots j$ þá geyma börn hennar svör við $i \dots m$ og $m + 1 \dots j$, þar sem m er miðja heiltölubilsins $[i, j]$.

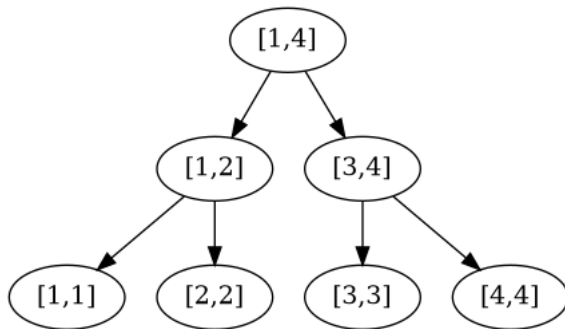
Biltré

- ▶ Biltré er tvíundartré sem geymir svör við vissum fyrirspurnum af seinni gerðinni.
- ▶ Rótin geymir svar við fyrirspurninni $1 \dots n$ og ef nóða geymir svarið við $i \dots j$ þá geyma börn hennar svör við $i \dots m$ og $m + 1 \dots j$, þar sem m er miðja heiltölubilsins $[i, j]$.
- ▶ Þær nóður sem geyma svar við fyrirspurnum af gerðinni $i \dots i$ eru lauf trésins.

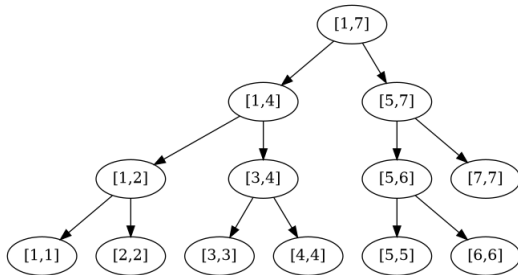
Biltré

- ▶ Biltré er tvíundartré sem geymir svör við vissum fyrirspurnum af seinni gerðinni.
- ▶ Rótin geymir svar við fyrirspurninni $1 \dots n$ og ef nóða geymir svarið við $i \dots j$ þá geyma börn hennar svör við $i \dots m$ og $m + 1 \dots j$, þar sem m er miðja heiltölubilsins $[i, j]$.
- ▶ Þær nóður sem geyma svar við fyrirspurnum af gerðinni $i \dots i$ eru lauf trésins.
- ▶ Takið eftir að lafin geyma þá gildin í listanum og aðrar nóður geyma summu barna sinna.

Mynd af biltré, $n = 4$



Mynd af biltré, $n = 7$



Notkun biltrjáa

- Gerum ráð fyrir að við höfum biltré eins og lýst er að ofan og látum H tákna hæð trésins.

Notkun biltrjáa

- ▶ Gerum ráð fyrir að við höfum biltré eins og lýst er að ofan og látum H tákna hæð trésins.
- ▶ Hvernig getum við leyst fyrirspurnirnar á glærunni á undan, og hver er tímaflækjan?

Notkun biltrjáa

- ▶ Gerum ráð fyrir að við höfum biltré eins og lýst er að ofan og látum H tákna hæð trésins.
- ▶ Hvernig getum við leyst fyrirspurnirnar á glærunni á undan, og hver er tímaflækjan?
- ▶ Fyrri fyrirspurnin er einföld.

Notkun biltrjáa

- ▶ Gerum ráð fyrir að við höfum biltré eins og lýst er að ofan og látum H tákna hæð trésins.
- ▶ Hvernig getum við leyst fyrirspurnirnar á glærunni á undan, og hver er tímaflækjan?
- ▶ Fyrri fyrirspurnin er einföld.
- ▶ Ef við eigum að breyta i -ta stakinu í k finnum við fyrst laufið sem svarar til fyrirspurnar í i , setjum svarið þar sem k og förum svo upp í rót í gegnum foreldrin og uppfærum á leiðinni gildin í þeim nóðu sem við lendum í.

Notkun biltrjáa

- ▶ Gerum ráð fyrir að við höfum biltré eins og lýst er að ofan og látum H tákna hæð trésins.
- ▶ Hvernig getum við leyst fyrirspurnirnar á glærunni á undan, og hver er tímaflækjan?
- ▶ Fyrri fyrirspurnin er einföld.
- ▶ Ef við eigum að breyta i -ta stakinu í k finnum við fyrst laufið sem svarar til fyrirspurnar í i , setjum svarið þar sem k og förum svo upp í rót í gegnum foreldrin og uppfærum á leiðinni gildin í þeim nóðu sem við lendum í.
- ▶ Þar sem við heimsækjum bara þær nóður sem eru á veginum frá rót til laufs (mest H nóður) er tímaflækjan á fyrri fyrirspurninni $\mathcal{O}(\quad)$.

Notkun biltrjáa

- ▶ Gerum ráð fyrir að við höfum biltré eins og lýst er að ofan og látum H tákna hæð trésins.
- ▶ Hvernig getum við leyst fyrirspurnirnar á glærunni á undan, og hver er tímaflækjan?
- ▶ Fyrri fyrirspurnin er einföld.
- ▶ Ef við eigum að breyta i -ta stakinu í k finnum við fyrst laufið sem svarar til fyrirspurnar í i , setjum svarið þar sem k og förum svo upp í rót í gegnum foreldrin og uppfærum á leiðinni gildin í þeim nóðu sem við lendum í.
- ▶ Þar sem við heimsækjum bara þær nóður sem eru á veginum frá rót til laufs (mest H nóður) er tímaflækjan á fyrri fyrirspurninni $\mathcal{O}(H)$.

Biltré í C

```
23
24 void update_rec(int i, int j, int x, int y, int e) // Hjálparfall.
25 { // Við erum að leita að laufinu [x, x] og erum í [i, j].
26     if (i == j) p[e] += y;
27     else
28     {
29         int m = (i + j)/2;
30         if (x <= m) update_rec(i, m, x, y, LEFT(e));
31         else update_rec(m + 1, j, x, y, RIGHT(e));
32         p[e] = p[LEFT(e)] + p[RIGHT(e)];
33     }
34 }
35 void update(int x, int y)
36 { // Bætum y við x-ta stakið.
37     return update_rec(0, n - 1, x, y, 1);
```

- ▶ Seinni fyrirspurnin er ögn flóknari.

- ▶ Seinni fyrirspurnin er ögn flóknari.
- ▶ Auðveldast er að ímynda sér að við förum niður tréð og leitum að hvorum endapunktinum fyrir sig.

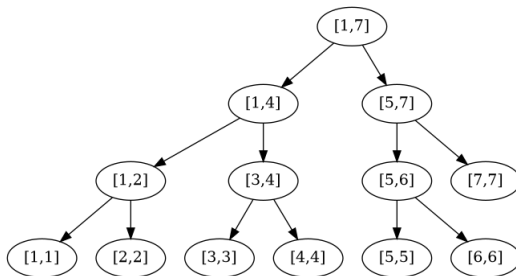
- ▶ Seinni fyrirspurnin er ögn flóknari.
- ▶ Auðveldast er að ímynda sér að við förum niður tréð og leitum að hvorum endapunktinum fyrir sig.
- ▶ Á leiðinni upp getum við svo pússlað saman svarinu, eftir því hvort við erum að skoða hægri eða vinstri endapunktinn.

- ▶ Seinni fyrirspurnin er ögn flóknari.
- ▶ Auðveldast er að ímynda sér að við förum niður tréð og leitum að hvorum endapunktinum fyrir sig.
- ▶ Á leiðinni upp getum við svo pússlað saman svarinu, eftir því hvort við erum að skoða hægri eða vinstri endapunktinn.
- ▶ Til dæmis, ef við erum að leita að vinstri endapunkti x og komum upp í bil $[i, j]$ þá bætum við gildinu í nóðu $[i, m]$ við það sem við höfum reiknað hingað til ef $x \in [m + 1, j]$, en annars bætum við engu við (því x er vinstri endapunkturinn).

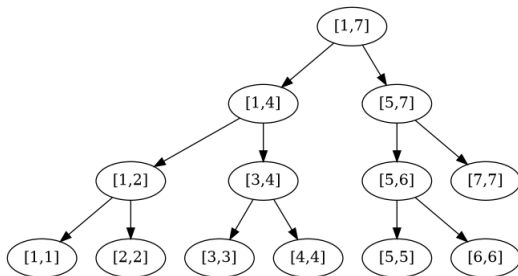
- ▶ Seinni fyrirspurnin er ögn flóknari.
- ▶ Auðveldast er að ímynda sér að við förum niður tréð og leitum að hvorum endapunktinum fyrir sig.
- ▶ Á leiðinni upp getum við svo pússlað saman svarinu, eftir því hvort við erum að skoða hægri eða vinstri endapunktinn.
- ▶ Til dæmis, ef við erum að leita að vinstri endapunkti x og komum upp í bil $[i, j]$ þá bætum við gildinu í nóðu $[i, m]$ við það sem við höfum reiknað hingað til ef $x \in [m + 1, j]$, en annars bætum við engu við (því x er vinstri endapunkturinn).
- ▶ Við göngum svona upp þar til við lendum í bili sem inniheldur hinn endapunktinn.

- ▶ Seinni fyrirspurnin er ögn flóknari.
- ▶ Auðveldast er að ímynda sér að við förum niður tréð og leitum að hvorum endapunktinum fyrir sig.
- ▶ Á leiðinni upp getum við svo pússlað saman svarinu, eftir því hvort við erum að skoða hægri eða vinstri endapunktinn.
- ▶ Til dæmis, ef við erum að leita að vinstri endapunkti x og komum upp í bil $[i, j]$ þá bætum við gildinu í nóðu $[i, m]$ við það sem við höfum reiknað hingað til ef $x \in [m + 1, j]$, en annars bætum við engu við (því x er vinstri endapunkturinn).
- ▶ Við göngum svona upp þar til við lendum í bili sem inniheldur hinn endapunktinn.
- ▶ Með sömu rökum og áðan er tímaflækjan $\mathcal{O}(\quad)$.

- ▶ Seinni fyrirspurnin er ögn flóknari.
- ▶ Auðveldast er að ímynda sér að við förum niður tréð og leitum að hvorum endapunktinum fyrir sig.
- ▶ Á leiðinni upp getum við svo pússlað saman svarinu, eftir því hvort við erum að skoða hægri eða vinstri endapunktinn.
- ▶ Til dæmis, ef við erum að leita að vinstri endapunkti x og komum upp í bil $[i, j]$ þá bætum við gildinu í nóðu $[i, m]$ við það sem við höfum reiknað hingað til ef $x \in [m + 1, j]$, en annars bætum við engu við (því x er vinstri endapunkturinn).
- ▶ Við göngum svona upp þar til við lendum í bili sem inniheldur hinn endapunktinn.
- ▶ Með sömu rökum og áðan er tímaflækjan $\mathcal{O}(H)$.

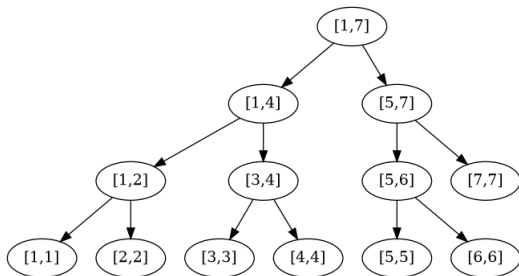


- ▶ Látum $f(i, j)$ tákna svar við fyrirspurninni í j og skoðum nokkur dæmi.

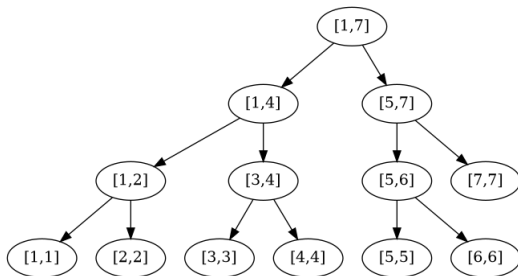


► Látum $f(i, j)$ tákna svar við fyrirspurninni í j og skoðum nokkur dæmi.

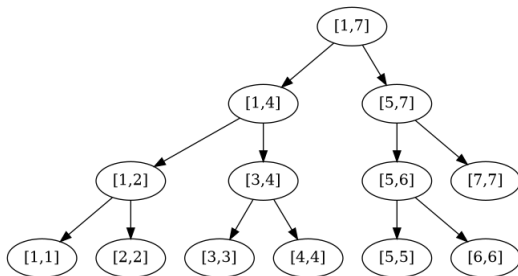
► $f(1, 3) = f(1, 2) + f(3, 3)$.



- ▶ Látum $f(i, j)$ tákna svar við fyrirspurninni í j og skoðum nokkur dæmi.
 - ▶ $f(1, 3) = f(1, 2) + f(3, 3)$.
 - ▶ $f(2, 5) = f(2, 2) + f(3, 4) + f(5, 5)$.



- ▶ Látum $f(i, j)$ tákna svar við fyrirspurninni í j og skoðum nokkur dæmi.
 - ▶ $f(1, 3) = f(1, 2) + f(3, 3)$.
 - ▶ $f(2, 5) = f(2, 2) + f(3, 4) + f(5, 5)$.
 - ▶ $f(1, 6) = f(1, 4) + f(5, 6)$.



► Látum $f(i, j)$ tákna svar við fyrirspurninni í j og skoðum nokkur dæmi.

- $f(1, 3) = f(1, 2) + f(3, 3)$.
- $f(2, 5) = f(2, 2) + f(3, 4) + f(5, 5)$.
- $f(1, 6) = f(1, 4) + f(5, 6)$.
- $f(3, 6) = f(3, 4) + f(5, 6)$.

Biltré í C

```
9
10 int query_rec(int i, int j, int x, int y, int e) // Hjálparfall.
11 { // Við erum að leita að bili [x, y] og erum í [i, j].
12     if (x == i && y == j) return p[e];
13     int m = (i + j)/2;
14     if (x <= m && y <= m) return query_rec(i, m, x, y, LEFT(e));
15     if (x > m && y > m) return query_rec(m + 1, j, x, y, RIGHT(e));
16     return query_rec(i, m, x, m, LEFT(e))
17         + query_rec(m + 1, j, m + 1, y, RIGHT(e));
18 }
19 int query(int x, int y)
20 { // Finnum summuna yfir [x, y].
21     return query_rec(0, n - 1, x, y, 1);
```

Tímaflækja biltrjáa

- ▶ Þar sem lengd hvers bils sem nóða svarar til helmingast þegar farið er niður tréð er $\mathcal{O}(H) = \mathcal{O}(\quad)$.

Tímaflækja biltrjáa

- ▶ Þar sem lengd hvers bils sem nóða svarar til helmingast þegar farið er niður tréð er $\mathcal{O}(H) = \mathcal{O}(\log n)$.

Tímaflækja biltrjáa

- ▶ Þar sem lengd hvers bils sem nóða svarar til helmingast þegar farið er niður tréð er $\mathcal{O}(H) = \mathcal{O}(\log n)$.
- ▶ Við erum því komin með lausn á upprunalega dæminu sem er $\mathcal{O}(\quad)$.

Tímaflækja biltrjáa

- ▶ Þar sem lengd hvers bils sem nóða svarar til helmingast þegar farið er niður tréð er $\mathcal{O}(H) = \mathcal{O}(\log n)$.
- ▶ Við erum því komin með lausn á upprunalega dæminu sem er $\mathcal{O}(q \cdot \log n)$.

Tímaflækja biltrjáa

- ▶ Þar sem lengd hvers bils sem nóða svarar til helmingast þegar farið er niður tréð er $\mathcal{O}(H) = \mathcal{O}(\log n)$.
- ▶ Við erum því komin með lausn á upprunalega dæminu sem er $\mathcal{O}(q \cdot \log n)$.
- ▶ Þetta væri nógu hratt ef, til dæmis, $n = q = 10^6$.

Annað dæmi

- Fyrsta lína inntaksins inniheldur tvær tölur, n og m , báðar jákvæðar heiltölur minni en 10^5 .

Annað dæmi

- ▶ Fyrsta lína inntaksins inniheldur tvær tölur, n og m , báðar jákvæðar heiltölur minni en 10^5 .
- ▶ Næsta lína inniheldur n heiltölur, á milli -10^9 og 10^9 .

Annað dæmi

- ▶ Fyrsta lína inntaksins inniheldur tvær tölur, n og m , báðar jákvæðar heiltölur minni en 10^5 .
- ▶ Næsta lína inniheldur n heiltölur, á milli -10^9 og 10^9 .
- ▶ Næstu m línur innihalda fyrirspurnir, af tveimur gerðum.

Annað dæmi

- ▶ Fyrsta lína inntaksins inniheldur tvær tölur, n og m , báðar jákvæðar heiltölur minni en 10^5 .
- ▶ Næsta lína inniheldur n heiltölur, á milli -10^9 og 10^9 .
- ▶ Næstu m línur innihalda fyrirspurnir, af tveimur gerðum.
- ▶ Fyrri gerðin hefst á 1 og inniheldur svo tvær tölur, x og y . Hér á að setja x -tu töluna sem y .

Annað dæmi

- ▶ Fyrsta lína inntaksins inniheldur tvær tölur, n og m , báðar jákvæðar heiltölur minni en 10^5 .
- ▶ Næsta lína inniheldur n heiltölur, á milli -10^9 og 10^9 .
- ▶ Næstu m línur innihalda fyrirspurnir, af tveimur gerðum.
- ▶ Fyrri gerðin hefst á 1 og inniheldur svo tvær tölur, x og y . Hér á að setja x -tu töluna sem y .
- ▶ Seinni gerðin hefst á 2 og inniheldur svo tvær tölur, x og y . Hér á að prenta út stærstu töluna á hlutbilinu $[x, y]$ í talnalistanum.

Annað dæmi

- ▶ Fyrsta lína inntaksins inniheldur tvær tölur, n og m , báðar jákvæðar heiltölur minni en 10^5 .
- ▶ Næsta lína inniheldur n heiltölur, á milli -10^9 og 10^9 .
- ▶ Næstu m línur innihalda fyrirspurnir, af tveimur gerðum.
- ▶ Fyrri gerðin hefst á 1 og inniheldur svo tvær tölur, x og y . Hér á að setja x -tu töluna sem y .
- ▶ Seinni gerðin hefst á 2 og inniheldur svo tvær tölur, x og y . Hér á að prenta út stærstu töluna á hlutbilinu $[x, y]$ í talnalistanum.
- ▶ Hvernig leysum við þetta?

- ▶ Við getum leyst þetta með biltrjám.

- ▶ Við getum leyst þetta með biltrjám.
- ▶ Í stað þess að láta nóður (sem eru ekki lauf) geyma summu barna sinna, þá geyma þær stærra stak barna sinna.

Lausn

```
12 { // Við erum að leita að bili [x, y] og erum í [i, j].
13   if (x == i && y == j) return p[e];
14   int m = (i + j)/2;
15   if (x <= m && y <= m) return query_rec(i, m, x, y, LEFT(e));
16   if (x > m && y > m) return query_rec(m + 1, j, x, y, RIGHT(e));
17   return max(query_rec(i, m, x, m, LEFT(e)),
18             query_rec(m + 1, j, m + 1, y, RIGHT(e)));
19 }
20 int query(int x, int y)
21 { // Finnum stærsta gildið á [x, y].
22   return query_rec(0, n - 1, x, y, 1);
23 }
24
25 void update_rec(int i, int j, int x, int y, int e) // Hjálparfall.
26 { // Við erum að leita að laufinu [x, x] og erum í [i, j].
27   if (i == j) p[e] = y;
28   else
29   {
30     int m = (i + j)/2;
31     if (x <= m) update_rec(i, m, x, y, LEFT(e));
32     else update_rec(m + 1, j, x, y, RIGHT(e));
33     p[e] = max(p[LEFT(e)], p[RIGHT(e)]);
34   }
35 }
36 void update(int x, int y)
37 { // Látum x-ta stakið vera y.
38   return update_rec(0, n - 1, x, y, 1);
39 }
```

- ▶ Leysa má ýmis dæmi af þessari gerð, með biltrjám.

- ▶ Leysa má ýmis dæmi af þessari gerð, með biltrjám.
- ▶ Þessi dæmi eru yfirleitt kölluð *punkt-uppfærslur*, *bil-fyrirspurnir* (e. *point-update*, *range-query*).

- ▶ Leysa má ýmis dæmi af þessari gerð, með biltrjám.
- ▶ Þessi dæmi eru yfirleitt kölluð *punkt-uppfærslur*, *bil-fyrirspurnir* (e. *point-update*, *range-query*).
- ▶ Algengt er að sýna næst hvernig nota megir biltré til að leysa *bil-uppfærslur*, *punkt-fyrirspurnir* (e. *range-update*, *point-query*).

- ▶ Leysa má ýmis dæmi af þessari gerð, með biltrjám.
- ▶ Þessi dæmi eru yfirleitt kölluð *punkt-uppfærslur*, *bil-fyrirspurnir* (e. *point-update*, *range-query*).
- ▶ Algengt er að sýna næst hvernig nota megir biltré til að leysa *bil-uppfærslur*, *punkt-fyrirspurnir* (e. *range-update*, *point-query*).
- ▶ Þetta er, í grófum dráttum, gert með því að snúa trjánnum við.

- ▶ Leysa má ýmis dæmi af þessari gerð, með biltrjám.
- ▶ Þessi dæmi eru yfirleitt kölluð *punkt-uppfærslur*, *bil-fyrirspurnir* (e. *point-update*, *range-query*).
- ▶ Algengt er að sýna næst hvernig nota megir biltré til að leysa *bil-uppfærslur*, *punkt-fyrirspurnir* (e. *range-update*, *point-query*).
- ▶ Þetta er, í grófum dráttum, gert með því að snúa trjánunni við.
- ▶ Við munum ekki skoða þetta.

- ▶ Leysa má ýmis dæmi af þessari gerð, með biltrjám.
- ▶ Þessi dæmi eru yfirleitt kölluð *punkt-uppfærslur*, *bil-fyrirspurnir* (e. *point-update*, *range-query*).
- ▶ Algengt er að sýna næst hvernig nota megir biltré til að leysa *bil-uppfærslur*, *punkt-fyrirspurnir* (e. *range-update*, *point-query*).
- ▶ Þetta er, í grófum dráttum, gert með því að snúa trjánunum við.
- ▶ Við munum ekki skoða þetta.
- ▶ Við tökum frekar fyrir *lygn biltré*.

- ▶ Leysa má ýmis dæmi af þessari gerð, með biltrjám.
- ▶ Þessi dæmi eru yfirleitt kölluð *punkt-uppfærslur*, *bil-fyrirspurnir* (e. *point-update*, *range-query*).
- ▶ Algengt er að sýna næst hvernig nota megir biltré til að leysa *bil-uppfærslur*, *punkt-fyrirspurnir* (e. *range-update*, *point-query*).
- ▶ Þetta er, í grófum dráttum, gert með því að snúa trjánnum við.
- ▶ Við munum ekki skoða þetta.
- ▶ Við tökum frekar fyrir *lygn biltré*.
- ▶ Þau leyfa okkur að leysa *bil-uppfærslur*, *bil-fyrirspurnir* (e. *range-update*, *range-query*).

Lygn dreifing

- ▶ Sem beinagrind munum við nota biltrjáa utfærsluna sem við notuðum til að leysa fyrsta dæmið.

Lygn dreifing

- ▶ Sem beinagrind munum við nota biltrjáa utfærsluna sem við notuðum til að leysa fyrsta dæmið.
- ▶ Við munum nú láta fyrri fyrirspurnina, $i \leq j \leq k$, þýða “Bættu k við allar tölur á bilinu $[i, j]$ ”.

Lygn dreifing

- ▶ Sem beinagrind munum við nota biltrjáa utfærsluna sem við notuðum til að leysa fyrsta dæmið.
- ▶ Við munum nú láta fyrri fyrirspurnina, $i \leq j \leq k$, þýða “Bættu k við allar tölur á bilinu $[i, j]$ ”.
- ▶ Uppfærslan er framkvæmd á svipaðan hátt og fyrirspurnirnar eru.

Lygn dreifing

- ▶ Sem beinagrind munum við nota biltrjáa utfærsluna sem við notuðum til að leysa fyrsta dæmið.
- ▶ Við munum nú láta fyrri fyrirspurnina, $i \leq j \leq k$, þýða “Bættu k við allar tölur á bilinu $[i, j]$ ”.
- ▶ Uppfærslan er framkvæmd á svipaðan hátt og fyrirspurnirnar eru.
- ▶ Við geymum í annari hrúgu þær uppfærslur sem við eigum eftir að framkvæma.

Lygn dreifing

- ▶ Sem beinagrind munum við nota biltrjáa utfærsluna sem við notuðum til að leysa fyrsta dæmið.
- ▶ Við munum nú láta fyrri fyrirspurnina, i j k , þýða “Bættu k við allar tölur á bilinu $[i, j]$ ”.
- ▶ Uppfærslan er framkvæmd á svipaðan hátt og fyrirspurnirnar eru.
- ▶ Við geymum í annari hrúgu þær uppfærslur sem við eigum eftir að framkvæma.
- ▶ Í hverri endurkvæmni (bæði uppfærslum og fyrirspurnum) dreifum við uppfærslunum í nóðunni niður á við.

Lygn dreifing

- ▶ Sem beinagrind munum við nota biltrjáa utfærsluna sem við notuðum til að leysa fyrsta dæmið.
- ▶ Við munum nú láta fyrri fyrirspurnina, i j k , þýða “Bættu k við allar tölur á bilinu $[i, j]$ ”.
- ▶ Uppfærslan er framkvæmd á svipaðan hátt og fyrirspurnirnar eru.
- ▶ Við geymum í annari hrúgu þær uppfærslur sem við eigum eftir að framkvæma.
- ▶ Í hverri endurkvæmni (bæði uppfærslum og fyrirspurnum) dreifum við uppfærslunum í nóðunni niður á við.
- ▶ Þetta kallast *lygn dreifing* (e. *lazy propagation*), því við framkvæmum hana bara þegar nauðsyn krefur.

Lygn dreifing

- ▶ Sem beinagrind munum við nota biltrjáa utfærsluna sem við notuðum til að leysa fyrsta dæmið.
- ▶ Við munum nú láta fyrri fyrirspurnina, i j k , þýða “Bættu k við allar tölur á bilinu $[i, j]$ ”.
- ▶ Uppfærslan er framkvæmd á svipaðan hátt og fyrirspurnirnar eru.
- ▶ Við geymum í annari hrúgu þær uppfærslur sem við eigum eftir að framkvæma.
- ▶ Í hverri endurkvæmni (bæði uppfærslum og fyrirspurnum) dreifum við uppfærslunum í nóðunni niður á við.
- ▶ Þetta kallast *lygn dreifing* (e. *lazy propagation*), því við framkvæmum hana bara þegar nauðsyn krefur.
- ▶ Ef biltré hefur lygna dreifingu köllum við það *lygnt biltré* (e. *segment tree with lazy propagation*).

- ▶ Látum $i < j$ vera heiltölur þannig að bilið $[i, j]$ svara til nóðu í bilþréi og m vera miðpunkt heiltölu bilsins $[i, j]$.

- ▶ Látum $i < j$ vera heiltölur þannig að bilið $[i, j]$ svara til nóðu í bilþréi og m vera miðpunkt heiltölu bilsins $[i, j]$.
- ▶ Gerum ráð fyrir að við eigum eftir að framkvæm uppfærslu í j k.

- ▶ Látum $i < j$ vera heiltölur þannig að bilið $[i, j]$ svara til nóðu í bilþréi og m vera miðpunkt heiltölu bilsins $[i, j]$.
- ▶ Gerum ráð fyrir að við eigum eftir að framkvæm uppfærslu $i \ j \ k$.
- ▶ Næst þegar við köllum á `query_rec(i, j, ...)` eða `update_rec(i, j, ...)` þá munum við dreifa uppfærslunni $i \ j \ k$.

- ▶ Látum $i < j$ vera heiltölur þannig að bilið $[i, j]$ svara til nóðu í bilþréi og m vera miðpunkt heiltölu bilsins $[i, j]$.
- ▶ Gerum ráð fyrir að við eigum eftir að framkvæm uppfærslu í j k.
- ▶ Næst þegar við köllum á `query_rec(i, j, ...)` eða `update_rec(i, j, ...)` þá munum við dreifa uppfærslunni í j k.
- ▶ Eftir dreifinguna munum við ekki eiga eftir uppfærslu á bilinu $[i, j]$, en við munum eiga eftir uppfærslurnar í m k og $(m + 1)$ j k.

- ▶ Látum $i < j$ vera heiltölur þannig að bilið $[i, j]$ svara til nóðu í bilþréi og m vera miðpunkt heiltölu bilsins $[i, j]$.
- ▶ Gerum ráð fyrir að við eigum eftir að framkvæm uppfærslu $i \ j \ k$.
- ▶ Næst þegar við köllum á `query_rec(i, j, ...)` eða `update_rec(i, j, ...)` þá munum við dreifa uppfærslunni $i \ j \ k$.
- ▶ Eftir dreifinguna munum við ekki eiga eftir uppfærslu á bilinu $[i, j]$, en við munum eiga eftir uppfærslurnar $i \ m \ k$ og $(m + 1) \ j \ k$.
- ▶ Þegar við dreifum uppfærslunni $i \ i \ k$ þá nægir að uppfæra tilheyrandi lauf í biltrénu.

- ▶ Áðan var sagt “laufin geyma þá gildin í listanum og aðrar nóður geyma summu barna sinna”.

- ▶ Áðan var sagt “laufin geyma þá gildin í listanum og aðrar nóður geyma summu barna sinna”.
- ▶ Þetta gildir ekki fyrir lygn biltré.

- ▶ Áðan var sagt “laufin geyma þá gildin í listanum og aðrar nóður geyma summu barna sinna”.
- ▶ Þetta gildir ekki fyrir lygn biltré.
- ▶ Nóður lygna biltrjáa þurfa að geyma summu barna sinna, ásamt því að geyma þá summu sem fengist eftir allar óframkvæmdar útfærslur afkomenda hennar.

- ▶ Áðan var sagt “laufin geyma þá gildin í listanum og aðrar nóður geyma summu barna sinna”.
- ▶ Þetta gildir ekki fyrir lygn biltré.
- ▶ Nóður lygna biltrjáa þurfa að geyma summu barna sinna, ásamt því að geyma þá summu sem fengist eftir allar óframkvæmdar útfærslur afkomenda hennar.
- ▶ Þegar við ferðumst í gegnum tréð til að finna hvert við eigum að setja uppfærsluna uppfærum við tréð jafn óðum.

- ▶ Áðan var sagt “laufin geyma þá gildin í listanum og aðrar nóður geyma summu barna sinna”.
- ▶ Þetta gildir ekki fyrir lygn biltré.
- ▶ Nóður lygna biltrjáa þurfa að geyma summu barna sinna, ásamt því að geyma þá summu sem fengist eftir allar óframkvæmdar útfærslur afkomenda hennar.
- ▶ Þegar við ferðumst í gegnum tréð til að finna hvert við eigum að setja uppfærsluna uppfærum við tréð jafn óðum.
- ▶ Til dæmis, ef við viljum framkvæma uppfærsluna í j k þá þurfum við að bæta $k \cdot (j - i - 1)$ við rót biltrésins, því rótin geymir summu allra stakana.

Lyngt biltré

```
11 { // Fall sem dreifir lygnum uppfærslum niður um eina hæð.
12   p[e] += (y - x + 1)*o[e];
13   if (x != y) o[LEFT(e)] += o[e], o[RIGHT(e)] += o[e];
14   o[e] = 0;
15 }
16 int query_rec(int i, int j, int x, int y, int e)
17 { // Við erum að leita að bili [x, y] og erum í [i, j].
18   prop(i, j, e);
19   if (x == i && y == j) return p[e];
20   int m = (i + j)/2;
21   if (x <= m && y <= m) return query_rec(i, m, x, y, LEFT(e));
22   else if (x > m && y > m) return query_rec(m + 1, j, x, y, RIGHT(e));
23   return query_rec(i, m, x, m, LEFT(e))
24         + query_rec(m + 1, j, m + 1, y, RIGHT(e));
25 }
26 int query(int x, int y)
27 { // Finnum summuna yfir [x, y].
28   return query_rec(0, n - 1, x, y, 1);
29 }
30 void update_rec(int i, int j, int x, int y, int z, int e)
31 { // Við erum að leita að bili [x, y] og erum í [i, j].
32   prop(i, j, e);
33   if (x == i && y == j) { o[e] = z; return; }
34   int m = (i + j)/2;
35   p[e] += (y - x + 1)*z;
36   if (x <= m && y <= m) update_rec(i, m, x, y, z, LEFT(e));
37   else if (x > m && y > m) update_rec(m + 1, j, x, y, z, RIGHT(e));
38   else update_rec(i, m, x, m, z, LEFT(e)),
39         update_rec(m + 1, j, m + 1, y, z, RIGHT(e));
40 }
41 void update(int x, int y, int z)
42 { // Bætum z við stökin á bilinu [x, y]
43   update_rec(0, n - 1, x, y, z, 1);
44 }
```

- ▶ Nú hefur `query(...)` sömu tímaflækju og í hefðbundnum biltrjám, það er að segja $\mathcal{O}(\quad)$.

- ▶ Nú hefur `query(...)` sömu tímaflækju og í hefðbundnum biltrjám, það er að segja $\mathcal{O}(\log n)$.

- ▶ Nú hefur `query(...)` sömu tímaflækju og í hefðbundnum biltrjám, það er að segja $\mathcal{O}(\log n)$.
- ▶ Loks fæst (með sömu rökum og gefa að `query(...)` er $\mathcal{O}(\log n)$) að `update(...)` er $\mathcal{O}(\quad)$.

- ▶ Nú hefur `query(...)` sömu tímaflækju og í hefðbundnum biltrjám, það er að segja $\mathcal{O}(\log n)$.
- ▶ Loks fæst (með sömu rökum og gefa að `query(...)` er $\mathcal{O}(\log n)$) að `update(...)` er $\mathcal{O}(\log n)$.

