# Talnafræði

## Leifareikningur, frumtölur o.fl.

Atli Fannar Franklín

10. mars 2019

# Efnisyfirlit

# Hvað er leifareikningur?

- Þegar við tölum um leifareikning erum við að tala um útreikninga sem eru allir gerðir á leifum einhverrar frumtölu $p$. Þá geymum við ávallt bara afgang tölunnar okkur þegar henni er deilt með $p$ í stað þess að geyma töluna alla.

# Hvað er leifareikningur?

- Þegar við tölum um leifareikning erum við að tala um útreikninga sem eru allir gerðir á leifum einhverrar frumtölu $p$. Þá geymum við ávallt bara afgang tölunnar okkur þegar henni er deilt með $p$ í stað þess að geyma töluna alla.
- Samlagning virkar með hvaða tölu $n$ sem er en til þess að margföldun virki eins og við viljum þurfum við að nota frumtölu $p$. Þetta er vegna þess að ef við vinnum með frumtölu þýðir $ab = 0$ að $a = 0$ eða $b = 0$. Ef við værum að reikna með t.d. 15 væri $5 \cdot 3 = 0$ hins vegar, sem við viljum ekki.

# Hvað er leifareikningur?

- Þegar við tölum um leifareikning erum við að tala um útreikninga sem eru allir gerðir á leifum einhverrar frumtölu $p$. Þá geymum við ávallt bara afgang tölunnar okkur þegar henni er deilt með $p$ í stað þess að geyma töluna alla.
- Samlagning virkar með hvaða tölu $n$ sem er en til þess að margföldun virki eins og við viljum þurfum við að nota frumtölu $p$. Þetta er vegna þess að ef við vinnum með frumtölu þýðir $ab = 0$ að $a = 0$ eða $b = 0$. Ef við værum að reikna með t.d. 15 væri $5 \cdot 3 = 0$ hins vegar, sem við viljum ekki.
- Það að geyma 'bara' leifina getur samt sagt okkur heilmikið um niðurstöðuna. Einnig kemur oft fyrir að svar í dæmi sé svo stórt að það sé bara beðið um að skila leifinni.

# Fleiri leifaaðgerðir

- Þegar við hefjum í veldi í leifareikningi hækkar talan ekki endilega, svo við erum oft beðin um að hefja tölu í afar stór veldi þegar við erum að framkvæma leifareikning. Hvernig má gera slíkt hratt? Það er augljóst hvernig má hefja í $n$-ta veldi í $\mathcal{O}(n)$ tíma. Hins vegar má gera það í $\mathcal{O}(\log(n))$ tíma!

# Fleiri leifaaðgerðir

- Þegar við hefjum í veldi í leifareikningi hækkar talan ekki endilega, svo við erum oft beðin um að hefja tölu í afar stór veldi þegar við erum að framkvæma leifareikning. Hvernig má gera slíkt hratt? Það er augljóst hvernig má hefja í $n$-ta veldi í $\mathcal{O}(n)$ tíma. Hins vegar má gera það í $\mathcal{O}(\log(n))$ tíma!
- Við hefjum bara í annað veldi aftur og aftur til að komast nær veldinu í $\log$ tíma, með smá auka til þess að við endum örugglega í nákvæmlega rétta veldinu!

# Fast mod pow

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

ll modpow(ll b, ll e, ll m) {
    ll res = 1;
    while(e) {
        if(e & 1) res = ((res * b) % m + m) % m;
        b = ((b * b) % m + m) % m;
        e >>= 1;
    }
    return res;
}
```

# En deiling?

- Hvernig deilum við þegar við erum að vinna með leif?

# En deiling?

- Hvernig deilum við þegar við erum að vinna með leif?
- Það er svolítið snúnara og við þurfum að skoða nokkra aðra hluti fyrst til að geta svarað þessu. Fyrst skoðum við stærsta samdeili. Vitið þið hvernig má finna stærsta samdeili í tölvu?

# En deiling?

- Hvernig deilum við þegar við erum að vinna með leif?
- Það er svolítið snúnara og við þurfum að skoða nokkra aðra hluti fyrst til að geta svarað þessu. Fyrst skoðum við stærsta samdeili. Vitið þið hvernig má finna stærsta samdeili í tölvu?
- Við notum reiknirit Evklíðs bara eins og það leggur sig!

# En deiling?

- Hvernig deilum við þegar við erum að vinna með leif?
- Það er svolítið snúnara og við þurfum að skoða nokkra aðra hluti fyrst til að geta svarað þessu. Fyrst skoðum við stærsta samdeili. Vitið þið hvernig má finna stærsta samdeili í tölvu?
- Við notum reiknirit Evklíðs bara eins og það leggur sig!
- Æfing: Sýna að þetta sé $\mathcal{O}(\log(n))$ (hint: sýna að $a + b$ minnki um fjórðung eftir tvær ítranir sama hvað)

# Reiknirit Evklíðs

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

ll gcd(ll a, ll b) {
    return b == 0 ? a : gcd(b, a % b);
}
```

# Reiknirit Evklíðs

- Skoðum þetta reiknirit samt aðeins nánar. Ef $a, b \in \mathbb{Z}$ og $d = gcd(a, b)$ kallast jafnan $ax + by = d$ Bézout-jafnan. Ef við greinum nákvæmlega hvað reiknirit Evklíðs gerir þá getum við séð að við getum lesið lausn á Bézout-jöfnunni út úr því sem reiknirit Evklíðs gerir.

# Reiknirit Evklíðs

- Skoðum þetta reiknirit samt aðeins nánar. Ef $a, b \in \mathbb{Z}$ og $d = gcd(a, b)$ kallast jafnan $ax + by = d$ Bézout-jafnan. Ef við greinum nákvæmlega hvað reiknirit Evklíðs gerir þá getum við séð að við getum lesið lausn á Bézout-jöfnunni út úr því sem reiknirit Evklíðs gerir.
- Því ef við breitum forritinu að ofan bara aðeins, svo það haldi aðeins meira bókhald, þá getum við leyst þessa jöfnu frítt með.

# Útvíkkaða reiknirit Evklíðs

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

ll egcd(ll a, ll b, ll& x, ll& y) {
    if (b == 0) {
        x = 1; y = 0;
        return a;
    }
    ll d = egcd(b, a % b, x, y);
    x -= a / b * y;
    swap(x, y);
    return d;
}
```

# Deiling núna?

- Hvernig deilum við þá? Það er til ein auðveld leið sem notar litlu setningu Fermats. Hún segir að fyrir frumtölu $p$ og $a < p$ þá sé $a^{p-1} = 1 \ (mod \ p)$. En þá sést að $a^{p-2}$ sé margföldunarandhverfa. En þetta er yfirleitt ekki leiðin sem notuð er. Við notum reiknirit Evklíðs.

# Deiling núna?

- Hvernig deilum við þá? Það er til ein auðveld leið sem notar litlu setningu Fermats. Hún segir að fyrir frumtölu $p$ og $a < p$ þá sé $a^{p-1} = 1 \ (mod \ p)$. En þá sést að $a^{p-2}$ sé margföldunarandhverfa. En þetta er yfirleitt ekki leiðin sem notuð er. Við notum reiknirit Evklíðs.
- Viljum finna margföldunarandhverfu $a$ þegar við erum að vinna modulo $p$. Þá er stærsti samdeilir $a$ og $p$ jafn einum, svo útvíkkaða reiknirit Evklíðs gefur okkur lausn á $ax + py = 1$. En þessi jafna skoðuð modulo $p$ er bara $ax = 1$, svo $x$ er margföldunarandhverfa $a$ modulo $p$.

# Deiling núna?

- Hvernig deilum við þá? Það er til ein auðveld leið sem notar litlu setningu Fermats. Hún segir að fyrir frumtölu $p$ og $a < p$ þá sé $a^{p-1} = 1 \ (mod \ p)$. En þá sést að $a^{p-2}$ sé margföldunarandhverfa. En þetta er yfirleitt ekki leiðin sem notuð er. Við notum reiknirit Evklíðs.
- Viljum finna margföldunarandhverfu $a$ þegar við erum að vinna modulo $p$. Þá er stærsti samdeilir $a$ og $p$ jafn einum, svo útvíkkaða reiknirit Evklíðs gefur okkur lausn á $ax + py = 1$. En þessi jafna skoðuð modulo $p$ er bara $ax = 1$, svo $x$ er margföldunarandhverfa $a$ modulo $p$.
- Því fáum við mjög stutt og þægilegt forrit til að gera þetta með því að nota egcd fallið okkar.

# Leifamargföldunarandhverfa

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

ll mod_inv(ll a, ll m) {
    ll x, y;
    ll d = egcd(a, m, x, y);
    return (x % m + m) % m;
}
```

# Efnisyfirlit

# Frumtölutékk

- Hvernig gáum við hvort tala sé frumtala?

# Frumtölutékk

- Hvernig gáum við hvort tala sé frumtala?
- Við skulum skoða þrjár leiðir til þess, byrjum á þeirri 'augljósu'.

# Frumtölutékk

- Hvernig gáum við hvort tala sé frumtala?
- Við skulum skoða þrjár leiðir til þess, byrjum á þeirri 'augljósu'.
- Við deilum bara upp í töluna með smærri tölum og sjáum hvort við fáum alltaf einhvern afgang.

# Frumtölutékk

- Hvernig gáum við hvort tala sé frumtala?
- Við skulum skoða þrjár leiðir til þess, byrjum á þeirri 'augljósu'.
- Við deilum bara upp í töluna með smærri tölum og sjáum hvort við fáum alltaf einhvern afgang.
- Það er samt margt sem má gera til að spara. T.d. þarf bara að skoða deila upp í kvaðratrótina af tölunni því ef tala er ekki frumtala er einhver þáttur minna en eða jafn kvaðratrót hennar.

# Frumtölutékk

- Hvernig gáum við hvort tala sé frumtala?
- Við skulum skoða þrjár leiðir til þess, byrjum á þeirri 'augljósu'.
- Við deilum bara upp í töluna með smærri tölum og sjáum hvort við fáum alltaf einhvern afgang.
- Það er samt margt sem má gera til að spara. T.d. þarf bara að skoða deila upp í kvaðratrótina af tölunni því ef tala er ekki frumtala er einhver þáttur minna en eða jafn kvaðratrót hennar.
- Við þurfum líka (fyrir utan $2$ og $3$) bara að skoða deila á forminu $6k+1$ og $6k+5$ því $2$ deilir $6k$, $3$ deilir $6k+3$ og $2$ deilir $6k+4$.

# Frumtölutékk

- Hvernig gáum við hvort tala sé frumtala?
- Við skulum skoða þrjár leiðir til þess, byrjum á þeirri 'augljósu'.
- Við deilum bara upp í töluna með smærri tölum og sjáum hvort við fáum alltaf einhvern afgang.
- Það er samt margt sem má gera til að spara. T.d. þarf bara að skoða deila upp í kvaðratrótina af tölunni því ef tala er ekki frumtala er einhver þáttur minna en eða jafn kvaðratrót hennar.
- Við þurfum líka (fyrir utan $2$ og $3$) bara að skoða deila á forminu $6k+1$ og $6k+5$ því $2$ deilir $6k$, $3$ deilir $6k+3$ og $2$ deilir $6k+4$.
- Saman gefur þetta okkur reiknirit sem keyrir í $\mathcal{O}(\sqrt{n})$ tíma.

# Leifamargföldunarandhverfa

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

bool isprime(ll n) {
    if(n < 2) return false;
    if(n == 2 || n == 3) return true;
    if(n % 2 == 0 || n % 3 == 0) return false;
    for(ll i = 5; i * i <= n; i += 6) {
        if(n % i == 0) return false;
        if(n % (i + 2) == 0) return false;
    }
    return true;
}
```

# Hraðar?

- Getum við gert þetta hraðar?

# Hraðar?

- Getum við gert þetta hraðar?
- Jáá... samt ekki.

# Hraðar?

- Getum við gert þetta hraðar?
- Jáá... samt ekki.
- Við getum notað probabilistic prime testing, s.s. fall sem segir að inntakið sé *líklega* frumtala.

# Hraðar?

- Getum við gert þetta hraðar?
- Jáá... samt ekki.
- Við getum notað probabilistic prime testing, s.s. fall sem segir að inntakið sé *líklega* frumtala.
- Það hljómar kannski grunsamlega en maður getur keyrt þetta forrit nokkra tugi skipta og þá eru líkurnar á því að forritið skili rangri niðurstöðu öll skiptin svo stjarnfræðilega litlar að maður myndi nýta tíma sinn betur við að hafa áhyggjur af tölvubilana af völdum geimgeisla.

# Miller-Rabin pælingar

- Bendum fyrst á að ef $x^2 = 1 \ (mod \ p)$ þá má þátta þetta sem $(x-1)(x+1) = 0 \ (mod \ p)$ og þar sem $p$ er frumtala þýðir það að $x = \pm 1 \ (mod \ p)$.

# Miller-Rabin pælingar

- Bendum fyrst á að ef $x^2 = 1 \ (mod \ p)$ þá má þátta þetta sem $(x-1)(x+1) = 0 \ (mod \ p)$ og þar sem $p$ er frumtala þýðir það að $x = \pm 1 \ (mod \ p)$.
- Tökum nú eitthvað $p > 2$ og $a < p$. Ritum $p - 1 = 2^s d$ þ.a. $d$ sé odda. Þá með því að taka ferningsrót af jöfnunni $a^{p-1} = 1 \ (mod \ p)$ (sem er bara litla setning Fermats) þá annað hvort verður hægri hliðin einhvern tímann $-1$ og þá verðum við að stoppa, eða á endanum deilum við út öllum tvistum í veldi $a$. Því er annað hvort $a^d = 1 \ (mod \ p)$ eða $a^{2^r d} = -1 \ (mod \ p)$ fyrir eitthvað $0 \leq r \leq s-1$.

# Miller-Rabin pælingar

- Bendum fyrst á að ef $x^2 = 1 \ (mod \ p)$ þá má þátta þetta sem $(x-1)(x+1) = 0 \ (mod \ p)$ og þar sem $p$ er frumtala þýðir það að $x = \pm 1 \ (mod \ p)$.
- Tökum nú eitthvað $p > 2$ og $a < p$. Ritum $p - 1 = 2^s d$ þ.a. $d$ sé odda. Þá með því að taka ferningsrót af jöfnunni $a^{p-1} = 1 \ (mod \ p)$ (sem er bara litla setning Fermats) þá annað hvort verður hægri hliðin einhvern tímann $-1$ og þá verðum við að stoppa, eða á endanum deilum við út öllum tvistum í veldi $a$. Því er annað hvort $a^d = 1 \ (mod \ p)$ eða $a^{2^r d} = -1 \ (mod \ p)$ fyrir eitthvað $0 \leq r \leq s-1$.
- Því til að afsanna að $n$ sé frumtala reynum við að finna $a < n$ þ.a. $a^d \neq 1 \ (mod \ n)$ og $a^{2^r d} \neq -1 \ (mod \ n)$ fyrir öll $0 \leq r \leq s-1$.

# Miller-Rabin pælingar frh.

- Að finna svona $a$ hljómar langsótt, en í ljós kemur að hátt hlutfall talna virkar sem slíkt $a$ ef $n$ er ekki frumtala (að sýna þetta er aðeins of flókið til að fara í það hér).

# Miller-Rabin pælingar frh.

- Að finna svona $a$ hljómar langsótt, en í ljós kemur að hátt hlutfall talna virkar sem slíkt $a$ ef $n$ er ekki frumtala (að sýna þetta er aðeins of flókið til að fara í það hér).
- Því virkar reiknirit Miller-Rabin með því að velja handahófskennt $a$ og sjá hvort það útiloki það að $n$ sé frumtala.

# Miller-Rabin pælingar frh.

- Að finna svona $a$ hljómar langsótt, en í ljós kemur að hátt hlutfall talna virkar sem slíkt $a$ ef $n$ er ekki frumtala (að sýna þetta er aðeins of flókið til að fara í það hér).
- Því virkar reiknirit Miller-Rabin með því að velja handahófskennt $a$ og sjá hvort það útiloki það að $n$ sé frumtala.
- Því er reikniritið þannig að ef það segir $p$ er ekki frumtala, þá er það nauðsynlega satt. En ef reikniritið segir $p$ er frumtala þýðir það 'ég gat ekki útilokað það að $p$ sé frumtala, það gæti samt verið ekki frumtala'.

# Miller-Rabin pælingar frh.

- Að finna svona $a$ hljómar langsótt, en í ljós kemur að hátt hlutfall talna virkar sem slíkt $a$ ef $n$ er ekki frumtala (að sýna þetta er aðeins of flókið til að fara í það hér).
- Því virkar reiknirit Miller-Rabin með því að velja handahófskennt $a$ og sjá hvort það útiloki það að $n$ sé frumtala.
- Því er reikniritið þannig að ef það segir $p$ er ekki frumtala, þá er það nauðsynlega satt. En ef reikniritið segir $p$ er frumtala þýðir það 'ég gat ekki útilokað það að $p$ sé frumtala, það gæti samt verið ekki frumtala'.
- En ef við prófum mörg $a$ þá eru líkurnar mjög svo okkur í hag. Láta má forritið taka inn breytu `it` sem segir hversu oft hún á að reyna og er þá hægt að stilla það eftir því hversu paranoid maður er. Þetta keyrir þá í $\mathcal{O}(it \log(n)^3)$ tíma fyrir stórar tölur $n$ (s.s. þegar við erum farin að nota bigInt).

# Miller-Rabin útfærsla

```
bool isprobablyprime(ll n, int it) {
    if(n % 2 == 0) return n == 2;
    if(n <= 3) return n == 3;
    ll d = n - 1, r = 0;
    while(d % 2 == 0) d >>= 1, r++;
    for(int i = 0; i < it; ++i) {
        ll a = (n - 3) * rand() / RAND_MAX + 2;
        ll x = modpow(a, d, n);
        if(x == 1 || x == n - 1) continue;
        for(ll j = 0; j < r - 1; ++j) {
            x = (x * x % n + n) % n;
            if(x == n - 1) continue;
        }
        return false;
    }
    return true;
}
```

# Magnafsláttur

- En ef við viljum finna allar frumtölur undir $n$? Getum við fengið einhverskonar magnafslátt þá?

# Magnafsláttur

- En ef við viljum finna allar frumtölur undir $n$? Getum við fengið einhverskonar magnafslátt þá?
- Já! Við notum sáldur Eratosthenes!

# Magnafsláttur

- En ef við viljum finna allar frumtölur undir $n$? Getum við fengið einhverskonar magnafslátt þá?
- Já! Við notum sáldur Eratosthenes!
- Það er mjög einfalt reiknirit. Við byrjum í 2 og krossum út allar sléttar tölur upp að $n$. Svo förum við áfram að næstu tölu sem er ekki búið að krossa út og krossum út öll margfeldi af henni, rinse and repeat. Þegar við erum komin upp að $n$ er búið að krossa út allt nema frumtölurnar.

# Magnafsláttur

- En ef við viljum finna allar frumtölur undir $n$? Getum við fengið einhverskonar magnafslátt þá?
- Já! Við notum sáldur Eratosthenes!
- Það er mjög einfalt reiknirit. Við byrjum í 2 og krossum út allar sléttar tölur upp að $n$. Svo förum við áfram að næstu tölu sem er ekki búið að krossa út og krossum út öll margfeldi af henni, rinse and repeat. Þegar við erum komin upp að $n$ er búið að krossa út allt nema frumtölurnar.
- Svo gerum við þetta aðeins hraðara með því að fara bara upp í rótina af $n$, krossa bara út tölur þar sem frumþátturinn okkar getur verið minnsti frumþátturinn o.s.frv. Þegar við förum að hellast yfir í biglnt verður tímaflækjan á þessu $\mathcal{O}(n \log(n) \log(\log(n)))$.

# Eratosthenes útfærsla

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

vector<bool> eratosthenes(ll n) {
    vector<bool> res(n + 1, true);
    res[0] = res[1] = false;
    for(ll i = 2; i * i <= n; ++i) {
        if(res[i]) {
            for(ll j = i * i; j <= n; j += i) {
                res[j] = false;
            }
        }
    }
    return res;
}
```

# Frumþáttun

- Sáldur Eratosthenes er mjög gott til að frumþátta tölur því þá geymir maður bara frumþátt tölunnar í stað bool og er þá tala frumtala ef hún er sjálf geymd í sætinu sínu. Þá má frumþátta með því að deila út tölunni í sæti þess í sáldrinu aftur og aftur þar til maður fær út ás.

# Frumþáttun

- Sáldur Eratosthenes er mjög gott til að frumþátta tölur því þá geymir maður bara frumþátt tölunnar í stað bool og er þá tala frumtala ef hún er sjálf geymd í sætinu sínu. Þá má frumþátta með því að deila út tölunni í sæti þess í sáldrinu aftur og aftur þar til maður fær út ás.
- En er til einhver hraðari leið til að frumþátta $n$?

# Frumþáttun

- Sáldur Eratosthenes er mjög gott til að frumþátta tölur því þá geymir maður bara frumþátt tölunnar í stað bool og er þá tala frumtala ef hún er sjálf geymd í sætinu sínu. Þá má frumþátta með því að deila út tölunni í sæti þess í sáldrinu aftur og aftur þar til maður fær út ás.
- En er til einhver hraðari leið til að frumþátta $n$?
- Aftur er svarið jáááaa, samt ekki.

# Frumþáttun

- Sáldur Eratosthenes er mjög gott til að frumþátta tölur því þá geymir maður bara frumþátt tölunnar í stað bool og er þá tala frumtala ef hún er sjálf geymd í sætinu sínu. Þá má frumþátta með því að deila út tölunni í sæti þess í sáldrinu aftur og aftur þar til maður fær út ás.
- En er til einhver hraðari leið til að frumþátta $n$?
- Aftur er svarið jáááaa, samt ekki.
- Við notum afmælisdagaþversögnina! Ef við höfum $n$ möguleg gildi er viðbúið að við þurfum að taka tilviljanakennt gildi $\mathcal{O}(\sqrt{n})$ sinnum til að fá endurtekningu. Við munum nýta þetta til að finna frumþátt í $n$. En fyrst smá hliðarspor.

# Floyd rásarleit

- Ef við höfum fall $f$ sem tekur inn heiltölur, hvernig finnum við hvort $f(x+n) = f(x)$ fyrir eitthvað $n$ (fyrir öll $x$ stærri en eitthvað $k$)?

# Floyd rásarleit

- Ef við höfum fall $f$ sem tekur inn heiltölur, hvernig finnum við hvort $f(x+n) = f(x)$ fyrir eitthvað $n$ (fyrir öll $x$ stærri en eitthvað $k$)?
- Þetta er oft nytsamlegt að geta gert hratt og til þess notum við rásarleitarreiknirit Floyds, einnig þekkt sem skjaldböku- og hérareikniritið.

# Floyd rásarleit

- Ef við höfum fall $f$ sem tekur inn heiltölur, hvernig finnum við hvort $f(x+n) = f(x)$ fyrir eitthvað $n$ (fyrir öll $x$ stærri en eitthvað $k$)?
- Þetta er oft nytsamlegt að geta gert hratt og til þess notum við rásarleitarreiknirit Floyds, einnig þekkt sem skjaldböku- og hérareikniritið.
- Trikkið er að $i$ er margfeldi af rásarlengd $f$ þ.þ.a.a. $f(i) = f(2i)$. Því dugar að skoða þá jöfnu til að finna rásarlengd $f$. Þegar það er búið getum við farið til baka til að hvar rásin byrjar og útfrá því er auðvelt að finna raunverulega rásarlengd.

# Floyd útfærsla

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef pair<int,int> ii;

ii floyd(int (*f)(int), int x0) {
    int t = (*f)(x0), h = (*f)((*f)(x0));
    while(t != h) {
        t = (*f)(t);
        h = (*f)((*f)(h));
    }
    int mu = 0; t = x0;
    while(t != h) {
        t = (*f)(t);
        h = (*f)(h);
        mu++;
    }
    int lam = 1; h = (*f)(t);
    while(t != h) {
        h = (*f)(h);
        lam++;
    }
    // lengd rásar, upphafspunktur rásarhegðunar
    return ii(lam, mu);
}
```

# Pollard rho þáttun

- Hugmyndin er nú sú að við látum $g(x) = x^2 + 1 \ (mod \ n)$ og búum til runu $x, g(x), g(g(x)), \ldots$ þar sem $x$ er valið bara einhvern veginn. Köllum þessar tölur $x_1, x_2, \ldots$. Við reiknum svona gildi og tékkum hvort við fáum einhvern tímann $\gcd(x_i - x_j, n) > 1$. Þá er runan farin að endurtaka sig ekki bara modulo $n$ heldur modulo $d$ þar sem $d$ deilir $n$. Þá gefur þessi stærsti samdeilir okkur tölu sem deilir $n$.

# Pollard rho þáttun

- Hugmyndin er nú sú að við látum $g(x) = x^2 + 1 \ (mod \ n)$ og búum til runu $x, g(x), g(g(x)), \ldots$ þar sem $x$ er valið bara einhvern veginn. Köllum þessar tölur $x_1, x_2, \ldots$. Við reiknum svona gildi og tékkum hvort við fáum einhvern tímann $\gcd(x_i - x_j, n) > 1$. Þá er runan farin að endurtaka sig ekki bara modulo $n$ heldur modulo $d$ þar sem $d$ deilir $n$. Þá gefur þessi stærsti samdeilir okkur tölu sem deilir $n$.
- Ef $\gcd(x_i - x_j, n) = 1$ fyrir öll gildin sem við skoðum er annað hvort $n$ frumtala eða okkur tókst bara ekki að finna frumþátt í $n$. Því prófum við nokkur upphafsgildi fyrir $x$ yfirleitt og gefumst upp ef ekkert þeirra virkar. Þetta reiknirit er almennt bara notað á biglnt því það borgar sig ekki fyrir en þá, en við sýnum þetta hér fyrir long long bara til að sýna einhverja útfærslu.

# Pollard rho útfærsla

```
ll rho(ll n) {
    vector<ll> seed = {2, 3, 4, 5, 7, 11, 13, 1031};
    for(ll s : seed) {
        ll x = s, y = x, d = 1;
        while(d == 1) {
            x = ((x * x + 1) % n + n) % n;
            y = ((y * y + 1) % n + n) % n;
            y = ((y * y + 1) % n + n) % n;
            d = gcd(abs(x - y), n);
        }
        if(d == n) continue;
        return d;
    }
    return -1;
}
```

# Efnisyfirlit

# Talnafræðiföll

- Nú með þessum aðferður hér á undan getum við frumþáttað tölu $n = p_1^{e_1} \dots p_r^{e_r}$. En hvað má nýta þetta í?

# Talnafræðiföll

- Nú með þessum aðferður hér á undan getum við frumþáttað tölu $n = p_1^{e_1} \dots p_r^{e_r}$. En hvað má nýta þetta í?
- Þetta má nýta í föll eins og:

# Talnafræðiföll

- Nú með þessum aðferður hér á undan getum við frumþáttað tölu $n = p_1^{e_1} \dots p_r^{e_r}$. En hvað má nýta þetta í?
- Þetta má nýta í föll eins og:

$$\text{Fjöldi deila } n : d(n) = \prod_{i=1}^{r} (e_i + 1)$$

# Talnafræðiföll

- Nú með þessum aðferður hér á undan getum við frumþáttað tölu $n = p_1^{e_1} \dots p_r^{e_r}$. En hvað má nýta þetta í?
- Þetta má nýta í föll eins og:

$$\text{Fjöldi deila } n : d(n) = \prod_{i=1}^{r} (e_i + 1)$$

$$\text{Summa deila } n : \sigma(n) = \prod_{i=1}^{r} \frac{p_i^{e_i+1} - 1}{p_i - 1}$$

$$\text{Fjöldi talna minni en } n \text{ ósamþátta } n : \varphi(n) = n \prod_{i=1}^{r} \left(1 - \frac{1}{p_i}\right)$$

# Kínverska leifasetningin

- Hvað ef við höfum jöfnuhneppi á forminu $a_i x = b_i \ (mod \ m_i)$ þar sem $i = 1, \ldots, n$? Hvernig leysum við slíkt?

# Kínverska leifasetningin

- Hvað ef við höfum jöfnuhneppi á forminu $a_i x = b_i \ (mod \ m_i)$ þar sem $i = 1, \ldots, n$? Hvernig leysum við slíkt?
- Í slíkt notum við kínversku leifasetninguna!

# Kínverska leifasetningin

- Hvað ef við höfum jöfnuhneppi á forminu $a_i x = b_i \ (mod\ m_i)$ þar sem $i = 1, \ldots, n$? Hvernig leysum við slíkt?
- Í slíkt notum við kínversku leifasetninguna!
- Það að reikna lausnina er samt rosalega ljótt og leiðinlegt. Við látum því duga að segja að ef $m_i$-in eru ósamþátta tvö og tvö, þá *er til* ótvírætt ákvörðuð lausn, og það dugar okkur yfirleitt. Mjög sjaldan þarf maður að reikna út raunverulegu lausnina.

# Fast Fourier Transform

- Þetta, eins og flæðireikniritin förum við ekki í.

# Fast Fourier Transform

- Þetta, eins og flæðireikniritin förum við ekki í.
- Hins vegar, eins og flæðireikniritin, eru mörg þyngri keppnisforritunardæmi sem byggja á þessu.

# Fast Fourier Transform

- Þetta, eins og flæðireikniritin förum við ekki í.
- Hins vegar, eins og flæðireikniritin, eru mörg þyngri keppnisforritunardæmi sem byggja á þessu.
- Því mælum við eindregið með að áhugasamir kynni sér þetta. Þetta er svolítið þungt efni, en þið getið rætt það við okkur ef þið viljið aðstoð við að átta ykkur á því.