Talnafræði

Frumtölur

Bergur Snorrason

17. mars 2021

- Heiltala a kallast *samsett* ef til eru heiltölur x og y, báðar stærri en 1, þannig að $a = x \cdot y$.
- Heiltala kallast *frumtala* ef hún er ekki samsett.
- Við segjum að talna runa $(a_n)_{n\in\mathbb{N}}$ sé *á endanum núll* ef til er jákvæð heiltala N þannig að $a_n = 0$ fyrir öll $n > N$.
- Látum p_n tákna n-tu minnstu jákvæðu frumtöluna.
- Þá er til, fyrir sérhverja jákvæða heiltölu a, nákvæmlega eina runa af jákvæðum heiltölum, $(e_n)_{n\in\mathbb{N}}$, sem er á endanum núll, þannig að
$$a = \prod_{n\in\mathbb{N}} p_n^{e_n}.$$
- Við köllum þessa þáttun *frumþáttun* tölunnar a.

- Við þurfum oft að ákvarða hvort tala sé frumtala.
- Oft þurfum við líka að frumþátta tölur.
- Til að ákvarða hvort tala sé frumtala er yfirleitt farið eina af þremur leiðum.
- Fyrst skoðum við hvernig þetta er gert með tæmandi leit.
- Síðan skoðum við sigti Eratosþenesar.
- Að lokum skoðum við slembið reinkirit.

- Ef n er samsett þá er til tala á milli núll og n sem deilir n.
- Köllum þá tölu a.
- Þá deilir n/a líka n.
- Einnig höfum við að $\min(a, n/a) \leq \sqrt{n}$.
- Við getum því umorðað fyrsta punkt þessara glæru sem „Ef n er samsett þá er til tala á milli núll og $\sqrt{n}$ sem deilir n".

```
int isp(ll x)
{
    ll i;
    if (x <= 1) return 0;
    for (i = 2; i*i <= x; i++) if (x%i == 0) return 0;
    return 1;
}
```

- Þetta reiknirit er $\mathcal{O}(\sqrt{n})$ því við þurfum bara að skoða jákvæðar heiltölur minni en $\sqrt{n}$.
- Ef við viljum finna allar frumtölur minni en n með þessari aðferð þarf $\mathcal{O}(n\sqrt{n})$ tíma.
- Við getum bætt þetta með sigti Eratosþenesar.

Sigti Eratosþenesar

- Við byrjum á að merkja allar tölur sem „óséðar''.
- Við merkjum síðan 0 og 1 sem „samsettar''.
- Við endurtökum svo eftirfarandi skref þangað til engin óséð tala er eftir:
  - Látum x vera minnstu „óséðu'' töluna.
  - Merkjum x sem „frumtölu''.
  - Merkjum svo allar tölur á forminu $n \cdot x$, fyrir $n > 1$ sem „samsettar''.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

		2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

		2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

		2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

	2	3	5	7	9
11		13	15	17	19
21		23	25	27	29
31		33	35	37	39
41		43	45	47	49
51		53	55	57	59
61		63	65	67	69
71		73	75	77	79
81		83	85	87	89
91		93	95	97	99

	2	3	5	7	9
11		13	15	17	19
21		23	25	27	29
31		33	35	37	39
41		43	45	47	49
51		53	55	57	59
61		63	65	67	69
71		73	75	77	79
81		83	85	87	89
91		93	95	97	99

	2	3	5	7	9
11		13	15	17	19
21		23	25	27	29
31		33	35	37	39
41		43	45	47	49
51		53	55	57	59
61		63	65	67	69
71		73	75	77	79
81		83	85	87	89
91		93	95	97	99

	2	3	5	7	
11		13		17	19
		23	25		29
31			35	37	
41		43		47	49
		53	55		59
61			65	67	
71		73		77	79
		83	85		89
91			95	97	

	2	3	5	7	
11		13		17	19
		23	25		29
31			35	37	
41		43		47	49
		53	55		59
61			65	67	
71		73		77	79
		83	85		89
91			95	97	

	2	3		5		7		
11		13				17		19
		23		25				29
31				35		37		
41		43				47		49
		53		55				59
61				65		67		
71		73				77		79
		83		85				89
91				95		97		

	2	3	5	7	
11		13		17	19
		23			29
31				37	
41		43		47	49
		53			59
61				67	
71		73		77	79
		83			89
91				97	

	2	3		5		7		
11		13				17		19
		23						29
31						37		
41		43				47		49
		53						59
61						67		
71		73				77		79
		83						89
91						97		

	2	3	5	7	
11		13		17	19
		23			29
31				37	
41		43		47	49
		53			59
61				67	
71		73		77	79
		83			89
91				97	

	2	3		5		7		
11		13				17		19
		23						29
31						37		
41		43				47		
		53						59
61						67		
71		73						79
		83						89
						97		

	2	3	5	7	
11		13		17	19
		23			29
31				37	
41		43		47	
		53			59
61				67	
71		73			79
		83			89
				97	

```
int e[MAXN];
void eratos()
{
    int i, j;
    rep(i, MAXN) e[i] = 1;
    e[0] = e[1] = 0;
    rep(i, MAXN) if (e[i] == 1) for (j = 2*i; j < MAXN; j += i) e[j] = 0;
}

int isp(int x)
{
    return e[x] == 1;
}
```

- Það tekur $\mathcal{O}(n \log \log n)$ tíma að forreikna sigtið.
- Hver fyrirspurn er síðan afgreidd í $\mathcal{O}(\, 1 \,)$ tíma.

Slembin reiknirit

- Hingað til hafa reinkiritin okkar verið annað hvort rétt eða röng.
- Það er þó til flokkur reinkirit þar á milli.
- *Slembið reiknirit* er reiknirit sem skilar réttu svar með líkum p.
- Við getum þá keyrt reikniritið s sinnum, og þá er það rétt með líkum p^s (gerum ráð fyrir óhæði).
- Ef reikniritið hefur tímaflækju $\mathcal{O}(f(n))$ þá tekur það $\mathcal{O}(s \cdot f(n))$ að keyra það s sinnum.
- Ef $p = 1/2$, til dæmis, þá fæst fyrir $s = 20$ að líkurnar eru betri en 10^{-6}.

Reiknirit Millers og Rabins

- Til er slembið reiknirit, kennt við Miller og Rabin, sem ákvarðar hvort tala sé samsett.
- Í því eru líkur á röngu svar minni eða jafnar $1/4$.
- Tímaflækjan á reikniritinu er $\mathcal{O}(s \cdot \log^3 n)$ og það er rétt með líkum betri en $1/4^s$.
- Ég mun ekki fara í það afhverju reikniritið virkar.
- Það er þó gott að þekkja það nógu vel til að geta notað það.
- Útfærslan sem er gefin notar niðurstöðu Jiang og Deng (2014) til að virka alltaf fyrir nógu litlar tölur.

```
int miller_rabin(ll n)
{
    if (n%2 == 0) return n == 2;
    if (n <= 3) return n == 3;
    ll i, k, s = 0, d = n - 1,
        t[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
    while (d%2 == 0) d /= 2, s++;
    rep(k, 12) if (t[k] <= n - 2)
    {
        ll a = t[k];
        ll x = modpow(a, d, n);
        if (x == 1 || x == n - 1) continue;
        rep(i, s - 1) if ((x = bigprod(x, x, n)) == n - 1) break;
        if (i == s - 1) return 0;
    }
    return 1;
}
```

- Þegar kemur að því að frumþátta tölur hafa þessar þrjár aðferðir sér hliðstæðu.
- Skoðum aftur fyrstu aðferðina.

```
int isp(ll x)
{
    ll i;
    if (x <= 1) return 0;
    for (i = 2; i*i <= x; i++) if (x%i == 0) return 0;
    return 1;
}
```

- Þegar þetta fall skilar núll er `i` minnsti frumþáttur `x`.
- Við getum nú stytt `x` með `i` þar til `i` gengur ekki lengur upp í `x`.
- Svo höldum við áfram.

```
int factor(ll x)
{
    ll i;
    for (i = 2; i*i <= x;)
    {
        if (x%i == 0) printf("%lld ", i), x /= i;
        else i++;
    }
    printf("%lld\n", x);
    return 1;
}
```

- Tímaflækja þessarar aðferðar er $\mathcal{O}(\sqrt{n})$.

- Í stað þess að láta sigti Eratosþenesar geyma hvort tala sé samsett eða ekki getum við látið það geyma minnsta frumþátt tölunnar.
- Takið eftir að n er frumtala þá og því aðeins að minnsti frumþáttur hennar sé n.
- Til að þátta tölur förum við endurkvæmt í gegn líkt og þegar við útfærðum sammengisleit.

```
int e[MAXN];
void eratos()
{
    int i, j;
    rep(i, MAXN) e[i] = 0;
    e[0] = e[1] = -1;
    rep(i, MAXN) if (e[i] == 0)
        for (j = i; j < MAXN; j += i) if (e[j] == 0) e[j] = i;
}

void factor(int x)
{
    if (x < 2) return;
    printf("%d ", e[x]);
    factor(x/e[x]);
}

int isp(int x)
{
    return e[x] == x;
}
```

- Þessi útgáfa er ekki verri en hin á neinn veg, og því er sniðugt að nota hana alltaf.
- Hún hefur sömu tímaflækjur: $\mathcal{O}(n \log \log n)$ tíma að forreikna og `isp(...)` fyrirspurnin tekur $\mathcal{O}(1)$ tíma.
- Nýja `factor(...)` fyrirspurnin tekur $\mathcal{O}(\log n)$ tíma því hún þarf að heimsækja hvern frumþátt tölunnar.

- Reiknirit Pollards er slembið reiknirit sem byggir á rásaleit til að finna þátt í samsettri tölu.
- Reikniritið finnur þátt í samsettri tölu n í $\mathcal{O}(\sqrt{a})$ tíma, þar sem a er minnsti frumþáttur n.
- Nú gildir að $a \leq \sqrt{n}$ og því tekur reikniritið $\mathcal{O}(\sqrt[4]{n})$ tíma.
- Þetta er því töluverð bæting.
- Við þurfum samt fyrst úr skugga um að n sé frumtala.
- Við megum þó ekki nota tæmandi leit til þess því þá bætist tímaflækjan ekkert.
- Líkt og með reiknirit Millers og Rabins þá mun ég ekki fara í smáatriði hér.

```
ll rho(ll n)
{ // skilar vonandi thaetti i |n|
    ll s[8] = {2, 3, 4, 5, 7, 11, 13, 1031}, i, j, a, x, y, d;
    for (a = 1;; a++) rep(j, 8)
    {
        x = y = s[j], d = 1;
        while (d == 1)
        {
            x = (bigprod(x, x, n) + a)%n;
            y = (bigprod(y, y, n) + a)%n;
            y = (bigprod(y, y, n) + a)%n;
            d = gcd(llabs(x - y), n);
        }
        if (d != n) return d;
    }
}
```

```
void pollard_rho(ll n)
{ // notar rho(...) ad ofan til ad thatta |n| og setur thaettina i |a|
    c = 0;
    ll i, s[200], ss = 0, p[6] = {2, 3, 5, 7, 11, 13};
    rep(i, 6) while (n%p[i] == 0) n /= p[i], a[c++] = p[i];
    s[ss++] = n;
    if (n == 1) return;
    while (ss > 0)
    {
        ll k = s[--ss];
        if (miller_rabin(k)) a[c++] = k;
        else
        {
            ll r = rho(k);
            s[ss++] = r;
            s[ss++] = k/r;
        }
    }
}
```

- Eftirfarandi jöfnur eru merkilegar hagnýtingar á frumþáttun.
- Látum $n = p_1^{e_1} \cdot \ldots \cdot p_m^{e_m}$, þar sem $p_1, \ldots, p_m$ eru frumtölur og $e_1, \ldots, e_m$ eru heiltölur stærri en 1.
- Við fáum þá eftirfarandi föll:
  - Fjöldi deila n:
$$d(n) = \prod_{k=1}^{r} (e_k + 1).$$
  - Summa deila n:
$$\sigma(n) = \prod_{k=1}^{r} \frac{p_k^{e_k+1} - 1}{p_k - 1}.$$
  - Fjöldi jákvæðra heiltalna $k < n$, þannig að $\gcd(n, k) = 1$:
$$\phi(n) = n \prod_{k=1}^{r} (1 - 1/p_k).$$
- Einnig gefur setning kennd við Euler alhæfingu á litlu setningu Fermats:
$$a^{\phi(m)} = 1 \mod m.$$
ef $\gcd(a, m) = 1$.