

Talningarfræði

Bergur Snorrason

19. mars 2021

- ▶ Talningarfræði er sá angi strjállar stærðfræði sem fjallar um talningar á einhverjum fyrirbærum.

- ▶ Talningarfræði er sá angi strjállar stærðfræði sem fjallar um talningar á einhverjum fyrirbærum.
- ▶ Þegar við fjölluðum um tæmandi leit kom fram að fjöldi hlutmengja í n staka mengi er 2^n .

- ▶ Talningarfræði er sá angi strjállar stærðfræði sem fjallar um talningar á einhverjum fyrirbærum.
- ▶ Þegar við fjölluðum um tæmandi leit kom fram að fjöldi hlutmengja í n staka mengi er 2^n .
- ▶ Einnig kom fram að fjöldi umraðana á menginu $\{1, 2, \dots, n\}$ er $n!$.

- ▶ Talningarfræði er sá angi strjállar stærðfræði sem fjallar um talningar á einhverjum fyrirbærum.
- ▶ Þegar við fjölluðum um tæmandi leit kom fram að fjöldi hlutmengja í n staka mengi er 2^n .
- ▶ Einnig kom fram að fjöldi umraðana á menginu $\{1, 2, \dots, n\}$ er $n!$.
- ▶ Bæði eru þetta mikilvægar niðurstöður úr talningarfræði.

- ▶ Talningarfræði er sá angi strjállar stærðfræði sem fjallar um talningar á einhverjum fyrirbærum.
- ▶ Þegar við fjölluðum um tæmandi leit kom fram að fjöldi hlutmengja í n staka mengi er 2^n .
- ▶ Einnig kom fram að fjöldi umraðana á menginu $\{1, 2, \dots, n\}$ er $n!$.
- ▶ Bæði eru þetta mikilvægar niðurstöður úr talningarfræði.
- ▶ Skerpum aðeins á grunnatriðum.

- ▶ Ef við erum með n hluti af einni gerð og m hluti af annari gerð þá getum við valið einn hlut af hvorri gerð á $n \cdot m$ vegu.

- ▶ Ef við erum með n hluti af einni gerð og m hluti af annari gerð þá getum við valið einn hlut af hvorri gerð á $n \cdot m$ vegu.
- ▶ Við þurfum í raun ekki meira en þetta.

- ▶ Ef við erum með n hluti af einni gerð og m hluti af annari gerð þá getum við valið einn hlut af hvorri gerð á $n \cdot m$ vegu.
- ▶ Við þurfum í raun ekki meira en þetta.
- ▶ Við notuðum þessa reglu til að sanna niðurstöðurnar á glærunni á undan.

- ▶ Ef við erum með n hluti af einni gerð og m hluti af annari gerð þá getum við valið einn hlut af hvorri gerð á $n \cdot m$ vegu.
- ▶ Við þurfum í raun ekki meira en þetta.
- ▶ Við notuðum þessa reglu til að sanna niðurstöðurnar á glærunni á undan.
- ▶ Í talningarfræði er oft þægilegt að hugsa um endanleg mengi og fjöldatölur þeirra.

- ▶ Ef við erum með n hluti af einni gerð og m hluti af annari gerð þá getum við valið einn hlut af hvorri gerð á $n \cdot m$ vegu.
- ▶ Við þurfum í raun ekki meira en þetta.
- ▶ Við notuðum þessa reglu til að sanna niðurstöðurnar á glærunni á undan.
- ▶ Í talningarfræði er oft þægilegt að hugsa um endanleg mengi og fjöldatölur þeirra.
- ▶ Ef A er mengi þá táknar $|A|$ fjölda staka í A .

- ▶ Ef við erum með n hluti af einni gerð og m hluti af annari gerð þá getum við valið einn hlut af hvorri gerð á $n \cdot m$ vegu.
- ▶ Við þurfum í raun ekki meira en þetta.
- ▶ Við notuðum þessa reglu til að sanna niðurstöðurnar á glærunni á undan.
- ▶ Í talningarfræði er oft þægilegt að hugsa um endanleg mengi og fjöldatölur þeirra.
- ▶ Ef A er mengi þá táknar $|A|$ fjölda staka í A .
- ▶ Við getum þá umorðað efsta punktinn sem: Ef A og B eru mengi þá er $|A \times B| = |A| \cdot |B|$.

- ▶ Við vitum að það eru 2^n hlutmengi í n staka mengi, en hvað eru mörg hlutmengi af stærð k ?

- ▶ Við vitum að það eru 2^n hlutmengi í n staka mengi, en hvað eru mörg hlutmengi af stærð k ?
- ▶ Þegar við veljum fyrst stakið höfum við um n stök að velja, síðan $n - 1$ stak og svo framvegis.

- ▶ Við vitum að það eru 2^n hlutmengi í n staka mengi, en hvað eru mörg hlutmengi af stærð k ?
- ▶ Þegar við veljum fyrst stakið höfum við um n stök að velja, síðan $n - 1$ stak og svo framvegis.
- ▶ Við fáum því $n \cdot (n - 1) \cdot \dots \cdot (n - k + 1) = \frac{n!}{k!}$ mengi.

- ▶ Við vitum að það eru 2^n hlutmengi í n staka mengi, en hvað eru mörg hlutmengi af stærð k ?
- ▶ Þegar við veljum fyrst stakið höfum við um n stök að velja, síðan $n - 1$ stak og svo framvegis.
- ▶ Við fáum því $n \cdot (n - 1) \cdot \dots \cdot (n - k + 1) = \frac{n!}{k!}$ mengi.
- ▶ Hvert mengi er þó talið $(n - k)!$ sinnum, svo loka talan er

$$\frac{n!}{(n - k)!k!}.$$

- ▶ Við vitum að það eru 2^n hlutmengi í n staka mengi, en hvað eru mörg hlutmengi af stærð k ?
- ▶ Þegar við veljum fyrst stakið höfum við um n stök að velja, síðan $n - 1$ stak og svo framvegis.
- ▶ Við fáum því $n \cdot (n - 1) \cdot \dots \cdot (n - k + 1) = \frac{n!}{k!}$ mengi.
- ▶ Hvert mengi er þó talið $(n - k)!$ sinnum, svo loka talan er

$$\frac{n!}{(n - k)!k!}.$$

- ▶ Þessi tala er táknuð með $\binom{n}{k}$.

- Tökum eftir að $|A \cup B| \neq |A| + |B|$ því það gætu verið stök í bæði A og B .

- ▶ Tökum eftir að $|A \cup B| \neq |A| + |B|$ því það gætu verið stök í bæði A og B .
- ▶ Ef svo er getum við einfaldlega fjarlæggt þau stök sem eru tvítalin, og fáum

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

- ▶ Tökum eftir að $|A \cup B| \neq |A| + |B|$ því það gætu verið stök í bæði A og B .
- ▶ Ef svo er getum við einfaldlega fjarlæggt þau stök sem eru tvítalin, og fáum

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

- ▶ Gerum nú ráð fyrir að

$$|A_1 \cup \dots \cup A_n| = \sum_{\substack{J \subset \{1, \dots, n\} \\ J \neq \emptyset}} (-1)^{|J|-1} \left| \bigcap_{j \in J} A_j \right|.$$

- ▶ Tökum eftir að $|A \cup B| \neq |A| + |B|$ því það gætu verið stök í bæði A og B .
- ▶ Ef svo er getum við einfaldlega fjarlæggt þau stök sem eru tvítalin, og fáum

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

- ▶ Gerum nú ráð fyrir að

$$|A_1 \cup \dots \cup A_n| = \sum_{\substack{J \subset \{1, \dots, n\} \\ J \neq \emptyset}} (-1)^{|J|-1} \left| \bigcap_{j \in J} A_j \right|.$$

- ▶ Þá fæst

$$\begin{aligned}
|A_1 \cup \dots \cup A_{n+1}| &= |A_1 \cup \dots \cup A_n| + |A_{n+1}| - |(A_1 \cup \dots \cup A_n) \cap A_{n+1}| \\
&= \sum_{\substack{J \subset \{1, \dots, n\} \\ J \neq \emptyset}} (-1)^{|J|-1} \left| \bigcap_{j \in J} A_j \right| + |A_{n+1}| - |(A_1 \cap A_{n+1}) \cup \dots \cup (A_n \cap A_{n+1})| \\
&= \sum_{\substack{J \subset \{1, \dots, n\} \\ J \neq \emptyset}} (-1)^{|J|-1} \left| \bigcap_{j \in J} A_j \right| + |A_{n+1}| + \sum_{\substack{J \subset \{1, \dots, n\} \\ J \neq \emptyset}} (-1)^{|J|} \left| \bigcap_{j \in J} (A_j \cap A_{n+1}) \right| \\
&= \sum_{\substack{J \subset \{1, \dots, n\} \\ J \neq \emptyset}} (-1)^{|J|-1} \left| \bigcap_{j \in J} A_j \right| + |A_{n+1}| + \sum_{\substack{J \subset \{1, \dots, n\} \\ J \neq \emptyset}} (-1)^{|J|} \left| \bigcap_{j \in J \cup \{n+1\}} A_j \right| \\
&= \sum_{\substack{J \subset \{1, \dots, n\} \\ J \neq \emptyset}} (-1)^{|J|-1} \left| \bigcap_{j \in J} A_j \right| + \sum_{J \subset \{1, \dots, n\}} (-1)^{|J|} \left| \bigcap_{j \in J \cup \{n+1\}} A_j \right| \\
&= \sum_{\substack{J \subset \{1, \dots, n+1\} \\ J \neq \emptyset \\ n+1 \notin J}} (-1)^{|J|-1} \left| \bigcap_{j \in J} A_j \right| + \sum_{\substack{J \subset \{1, \dots, n+1\} \\ n+1 \in J}} (-1)^{|J|-1} \left| \bigcap_{j \in J} A_j \right| \\
&= \sum_{\substack{J \subset \{1, \dots, n, n+1\} \\ J \neq \emptyset}} (-1)^{|J|-1} \left| \bigcap_{j \in J} A_j \right|
\end{aligned}$$

- Við höfum því sýnt með þreppun að

$$|A_1 \cup \dots \cup A_n| = \sum_{\substack{J \subset \{1, \dots, n\} \\ J \neq \emptyset}} (-1)^{|J|-1} \left| \bigcap_{j \in J} A_j \right|.$$

- ▶ Við höfum því sýnt með þreppun að

$$|A_1 \cup \dots \cup A_n| = \sum_{\substack{J \subset \{1, \dots, n\} \\ J \neq \emptyset}} (-1)^{|J|-1} \left| \bigcap_{j \in J} A_j \right|.$$

- ▶ Vera má að þessi jafna komi spánkst fyrir sjónir en er í raun að fjarlægja þau stök sem eru tvítekin, bæta aftur við stökum sem eru þrítekin, fjarlægja aftur stök sem eru fjórtekin og svo framvegis.

- ▶ Við höfum því sýnt með þrepun að

$$|A_1 \cup \dots \cup A_n| = \sum_{\substack{J \subset \{1, \dots, n\} \\ J \neq \emptyset}} (-1)^{|J|-1} \left| \bigcap_{j \in J} A_j \right|.$$

- ▶ Vera má að þessi jafna komi spánkst fyrir sjónir en er í raun að fjarlægja þau stök sem eru tvítekin, bæta aftur við stökum sem eru þrítekin, fjarlægja aftur stök sem eru fjórtekin og svo framvegis.
- ▶ Þessi jafna er kölluð *lögmálið um fjöldatölu sammengja* (e. *Inclusion-Exclusion principle*).

- ▶ Sjáum nú hagnýtingu á þessari jöfnu.

- ▶ Sjáum nú hagnýtingu á þessari jöfnu.
- ▶ Munum að gagntækj vörpun $\sigma: \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$ kallast *umröðun* (e. *permutation*).

- ▶ Sjáum nú hagnýtingu á þessari jöfnu.
- ▶ Munum að gagntækj vörpun $\sigma: \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$ kallast *umröðun* (e. *permutation*).
- ▶ Ef við festum n þá höfum við sýnt að til séu $n!$ umraðanir.

- ▶ Sjáum nú hagnýtingu á þessari jöfnu.
- ▶ Munum að gagntækj vörpun $\sigma: \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$ kallast *umröðun* (e. *permutation*).
- ▶ Ef við festum n þá höfum við sýnt að til séu $n!$ umraðanir.
- ▶ Næst segjum við að $k \in \{1, 2, \dots, n\}$ sé *fastapunktur* σ (e. *fixed point of σ*) ef $\sigma(k) = k$.

- ▶ Sjáum nú hagnýtingu á þessari jöfnu.
- ▶ Munum að gagntækj vörpun $\sigma: \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$ kallast *umröðun* (e. *permutation*).
- ▶ Ef við festum n þá höfum við sýnt að til séu $n!$ umraðanir.
- ▶ Næst segjum við að $k \in \{1, 2, \dots, n\}$ sé *fastapunktur* σ (e. *fixed point of σ*) ef $\sigma(k) = k$.
- ▶ Með öðrum orðum hefur umröðunin ekki áhrif á þennan punkt.

- ▶ Sjáum nú hagnýtingu á þessari jöfnu.
- ▶ Munum að gagntækj vörpun $\sigma: \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$ kallast *umröðun* (e. *permutation*).
- ▶ Ef við festum n þá höfum við sýnt að til séu $n!$ umraðanir.
- ▶ Næst segjum við að $k \in \{1, 2, \dots, n\}$ sé *fastapunktur* σ (e. *fixed point of σ*) ef $\sigma(k) = k$.
- ▶ Með öðrum orðum hefur umröðunin ekki áhrif á þennan punkt.
- ▶ Hversu margar umraðanir hafa engan fastapunkt?

► Skoðum fyrst þegar $n = 4$:

1 2 3 4	2 1 4 3	3 1 2 4	4 1 2 3
1 2 4 3	2 1 3 4	3 1 4 2	4 1 3 2
1 3 2 4	2 3 1 4	3 2 1 4	4 2 1 3
1 3 4 2	2 3 4 1	3 2 4 1	4 2 3 1
1 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
1 4 3 2	2 4 3 1	3 4 2 1	4 3 2 1

► Skoðum fyrst þegar $n = 4$:

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ \wedge & \wedge & \wedge & \wedge \end{array}$$

2 1 4 3

$$\begin{array}{cccc} 3 & 1 & 2 & 4 \\ & & & \wedge \end{array}$$

4 1 2 3

$$\begin{array}{cccc} 1 & 2 & 4 & 3 \\ \wedge & \wedge & & \end{array}$$
$$\begin{array}{cccc} 2 & 1 & 3 & 4 \\ & & \wedge & \wedge \end{array}$$

3 1 4 2

$$\begin{array}{cccc} 4 & 1 & 3 & 2 \\ & & \wedge & \end{array}$$
$$\begin{array}{cccc} 1 & 3 & 2 & 4 \\ \wedge & & & \wedge \end{array}$$
$$\begin{array}{cccc} 2 & 3 & 1 & 4 \\ & & & \wedge \end{array}$$
$$\begin{array}{cccc} 3 & 2 & 1 & 4 \\ & \wedge & & \wedge \end{array}$$
$$\begin{array}{cccc} 4 & 2 & 1 & 3 \\ & \wedge & & \end{array}$$
$$\begin{matrix} 1 & 3 & 4 & 2 \\ \wedge \end{matrix}$$

2 3 4 1

$$\begin{array}{cccc} 3 & 2 & 4 & 1 \\ & \wedge & & \end{array}$$
$$\begin{array}{cccc} 4 & 2 & 3 & 1 \\ & \wedge & \wedge & \\ & & & \end{array}$$
$$\begin{matrix} 1 & 4 & 2 & 3 \\ \wedge \end{matrix}$$

2 4 1 3

3 4 1 2

4 3 1 2

$$\begin{array}{cccc} 1 & 4 & 3 & 2 \\ \wedge & & \wedge & \end{array}$$
$$\begin{array}{cccc} 2 & 4 & 3 & 1 \\ & & \wedge & \end{array}$$

3 4 2 1

4 3 2 1

- Skoðum fyrst þegar $n = 4$:

2 1 4 3

4 1 2 3

3 1 4 2

2 3 4 1

2 4 1 3

3 4 1 2

4 3 1 2

3 4 2 1

4 3 2 1

- ▶ Við munum ferkar telja hversu margar umraðanir hafa fastapunkt.

- ▶ Við munum ferkar telja hversu margar umraðanir hafa fastapunkt.
- ▶ Þetta er algengt að gera í talningarfræði.

- ▶ Við munum ferkar telja hversu margar umraðanir hafa fastapunkt.
- ▶ Þetta er algenzt að gera í talningarfræði.
- ▶ Látum nú A_j tákna mengi þeirra umraðana þar sem j er fastapunktur.

- ▶ Við munum ferkar telja hversu margar umraðanir hafa fastapunkt.
- ▶ Þetta er algenzt að gera í talningarfræði.
- ▶ Látum nú A_j tákna mengi þeirra umraðana þar sem j er fastapunktur.
- ▶ Þá er $\bigcap_{j \in J} A_j$ mengi þeirra umraðana þar sem allir punktar J eru fastapunktur.

- ▶ Við munum ferkar telja hversu margar umraðanir hafa fastapunkt.
- ▶ Þetta er algenzt að gera í talningarfræði.
- ▶ Látum nú A_j tákna mengi þeirra umraðana þar sem j er fastapunktur.
- ▶ Þá er $\bigcap_{j \in J} A_j$ mengi þeirra umraðana þar sem allir punktar J eru fastapunktur.
- ▶ Ef við festum k punkta í umröðuninni getum við raðað restinni á $(n - k)!$ marga vegu.

- ▶ Við munum ferkar telja hversu margar umraðanir hafa fastapunkt.
- ▶ Þetta er algenzt að gera í talningarfræði.
- ▶ Látum nú A_j tákna mengi þeirra umraðana þar sem j er fastapunktur.
- ▶ Þá er $\bigcap_{j \in J} A_j$ mengi þeirra umraðana þar sem allir punktar J eru fastapunktur.
- ▶ Ef við festum k punkta í umröðuninni getum við raðað restinni á $(n - k)!$ marga vegu.
- ▶ Svo $|A_j| = (n - 1)!$ og $\left| \bigcap_{j \in J} A_j \right| = (n - |J|)!$.

- ▶ Við munum ferkar telja hversu margar umraðanir hafa fastapunkt.
- ▶ Þetta er algenzt að gera í talningarfræði.
- ▶ Látum nú A_j tákna mengi þeirra umraðana þar sem j er fastapunktur.
- ▶ Þá er $\bigcap_{j \in J} A_j$ mengi þeirra umraðana þar sem allir punktar J eru fastapunktur.
- ▶ Ef við festum k punkta í umröðuninni getum við raðað restinni á $(n - k)!$ marga vegu.
- ▶ Svo $|A_j| = (n - 1)!$ og $\left| \bigcap_{j \in J} A_j \right| = (n - |J|)!$.
- ▶ Takið eftir að seinni stærðin er bara háð fjölda staka í menginu J .

- ▶ Við munum ferkar telja hversu margar umraðanir hafa fastapunkt.
- ▶ Þetta er algenzt að gera í talningarfræði.
- ▶ Látum nú A_j tákna mengi þeirra umraðana þar sem j er fastapunktur.
- ▶ Þá er $\bigcap_{j \in J} A_j$ mengi þeirra umraðana þar sem allir punktar J eru fastapunktur.
- ▶ Ef við festum k punkta í umröðuninni getum við raðað restinni á $(n - k)!$ marga vegu.
- ▶ Svo $|A_j| = (n - 1)!$ og $\left| \bigcap_{j \in J} A_j \right| = (n - |J|)!$.
- ▶ Takið eftir að seinni stærðin er bara háð fjölda staka í menginu J .
- ▶ Við vitum einnig að fjöldi hlutmengja $\{1, \dots, n\}$ með k stök er $\binom{n}{k}$.

- Við fáum loks að fjöldi umraðana með einhvern fastapunkt er

$$\begin{aligned}|A_1 \cup \dots \cup A_n| &= \sum_{\substack{J \subset \{1, \dots, n\} \\ J \neq \emptyset}} (-1)^{|J|-1} \left| \bigcap_{j \in J} A_j \right| \\&= \sum_{j=1}^n (-1)^{j+1} \binom{n}{j} (n-j)! \\&= \sum_{j=1}^n (-1)^{j+1} \frac{n!}{(n-j)!j!} (n-j)! \\&= \sum_{j=1}^n (-1)^{j+1} \frac{n!}{j!} \\&= n! \sum_{j=1}^n \frac{(-1)^{j+1}}{j!}.\end{aligned}$$

- Fjöldi umraðana með engan fasta punkt er því

$$\begin{aligned} n! - n! \sum_{j=1}^n \frac{(-1)^{j+1}}{j!} &= n! \left(1 + \sum_{j=1}^n \frac{(-1)^j}{j!} \right) \\ &= n! \left(\frac{(-1)^0}{0!} + \sum_{j=1}^n \frac{(-1)^j}{j!} \right) \\ &= n! \sum_{j=0}^n \frac{(-1)^j}{j!}. \end{aligned}$$

- Fjöldi umraðana með engan fasta punkt er því

$$\begin{aligned} n! - n! \sum_{j=1}^n \frac{(-1)^{j+1}}{j!} &= n! \left(1 + \sum_{j=1}^n \frac{(-1)^j}{j!} \right) \\ &= n! \left(\frac{(-1)^0}{0!} + \sum_{j=1}^n \frac{(-1)^j}{j!} \right) \\ &= n! \sum_{j=0}^n \frac{(-1)^j}{j!}. \end{aligned}$$

- Algengt er að kalla þessa tölu $!n$.

- Fjöldi umraðana með engan fasta punkt er því

$$\begin{aligned} n! - n! \sum_{j=1}^n \frac{(-1)^{j+1}}{j!} &= n! \left(1 + \sum_{j=1}^n \frac{(-1)^j}{j!} \right) \\ &= n! \left(\frac{(-1)^0}{0!} + \sum_{j=1}^n \frac{(-1)^j}{j!} \right) \\ &= n! \sum_{j=0}^n \frac{(-1)^j}{j!}. \end{aligned}$$

- Algengt er að kalla þessa tölu $!n$.
- Takið eftir að $!n/n!$ er n -ta hlutsumma veldaraðar e^x , fyrir $x = -1$.

- Fjöldi umraðana með engan fasta punkt er því

$$\begin{aligned} n! - n! \sum_{j=1}^n \frac{(-1)^{j+1}}{j!} &= n! \left(1 + \sum_{j=1}^n \frac{(-1)^j}{j!} \right) \\ &= n! \left(\frac{(-1)^0}{0!} + \sum_{j=1}^n \frac{(-1)^j}{j!} \right) \\ &= n! \sum_{j=0}^n \frac{(-1)^j}{j!}. \end{aligned}$$

- Algengt er að kalla þessa tölu $!n$.
- Takið eftir að $!n/n!$ er n -ta hlutsumma veldaraðar e^x , fyrir $x = -1$.
- Svo $!n/n!$ er að stefna á e^{-1} , þegar n stefnir á ∞ .

- Oft er hentugt að geta reiknað

$$\binom{n}{k} \bmod m.$$

fyrir jákvæðar heiltölur $k \leq n < m$.

- Oft er hentugt að geta reiknað

$$\binom{n}{k} \bmod m.$$

fyrir jákvæðar heiltölur $k \leq n < m$.

- Munið að

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}.$$

- Oft er hentugt að geta reiknað

$$\binom{n}{k} \bmod m.$$

fyrir jákvæðar heiltölur $k \leq n < m$.

- Munið að

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}.$$

- Ein leið til að gera þetta er að finna fyrst margföldunarandhverfur $(n-k)!$ og $k!$ með tilliti til m .

- Oft er hentugt að geta reiknað

$$\binom{n}{k} \bmod m.$$

fyrir jákvæðar heiltölur $k \leq n < m$.

- Munið að

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}.$$

- Ein leið til að gera þetta er að finna fyrst margföldunarandhverfur $(n-k)!$ og $k!$ með tilliti til m .
- Við reiknum svo $n! \cdot ((n-k)!)^{-1} \cdot (k!)^{-1} \bmod m$.

- Oft er hentugt að geta reiknað

$$\binom{n}{k} \bmod m.$$

fyrir jákvæðar heiltölur $k \leq n < m$.

- Munið að

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}.$$

- Ein leið til að gera þetta er að finna fyrst margföldunarandhverfur $(n-k)!$ og $k!$ með tilliti til m .
- Við reiknum svo $n! \cdot ((n-k)!)^{-1} \cdot (k!)^{-1} \bmod m$.
- Hér þarf að passa sig að margföldunarandhverfan sé til.

- ▶ Oft er hentugt að geta reiknað

$$\binom{n}{k} \bmod m.$$

fyrir jákvæðar heiltölur $k \leq n < m$.

- ▶ Munið að

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}.$$

- ▶ Ein leið til að gera þetta er að finna fyrst margföldunarandhverfur $(n-k)!$ og $k!$ með tilliti til m .
- ▶ Við reiknum svo $n! \cdot ((n-k)!)^{-1} \cdot (k!)^{-1} \bmod m$.
- ▶ Hér þarf að passa sig að margföldunarandhverfan sé til.
- ▶ Helst þarf m að vera frumtala.

```

27 ll f[MAXN], fm[MAXN];
28 ll prepare_nck(ll m)
29 {
30     ll i;
31     f[0] = 1;
32     rep(i, MAXN) if (i != 0) f[i] = (f[i - 1]*i)%m;
33     rep(i, MAXN) fm[i] = mulinv(f[i], m);
34 }
35
36 ll nck(ll n, ll k, ll m)
37 {
38     return (((f[n]*fm[n - k])%m)*fm[k])%m;
39 }

```

- ▶ Ef við erum með fast n og m og viljum reikna fyrir q mismunandi gildi á k þá er tímaflækjan á þessari að ferð $O(\quad)$.

- ▶ Ef við erum með fast n og m og viljum reikna fyrir q mismunandi gildi á k þá er tímaflækjan á þessari að ferð $\mathcal{O}(n \log m + q)$.

- ▶ Önnur leið til að reikna $\binom{n}{k}$ mod m byggir á kvikri bestun.

- ▶ Önnur leið til að reikna $\binom{n}{k} \bmod m$ byggir á kvikri bestun.
- ▶ Sjáum fyrst að ef $n > 1$ og $1 < k < n$ þá

- ▶ Önnur leið til að reikna $\binom{n}{k}$ mod m byggir á kvikri bestun.
- ▶ Sjáum fyrst að ef $n > 1$ og $1 < k < n$ þá

$$\begin{aligned}
 \binom{n-1}{k-1} + \binom{n-1}{k} &= \frac{(n-1)!}{(n-k)!(k-1)!} + \frac{(n-1)!}{(n-k-1)!k!} \\
 &= \frac{k(n-1)!}{(n-k)!k!} + \frac{(n-k)(n-1)!}{(n-k)!k!} \\
 &= \frac{k(n-1)! + n(n-1)! - k(n-1)!}{(n-k)!k!} \\
 &= \frac{n(n-1)!}{(n-k)!k!} \\
 &= \frac{n!}{(n-k)!k!} \\
 &= \binom{n}{k}.
 \end{aligned}$$

► Látum því

$$f(n, k) = \begin{cases} 0, & \text{ef } n < 1 \\ 0, & \text{ef } k < 0 \\ 0, & \text{ef } k > n \\ 1, & \text{ef } k = 0 \text{ eða } k = n \\ f(n-1, k-1) + f(n-1, k) & \text{annars.} \end{cases}$$

- ▶ Látum því

$$f(n, k) = \begin{cases} 0, & \text{ef } n < 1 \\ 0, & \text{ef } k < 0 \\ 0, & \text{ef } k > n \\ 1, & \text{ef } k = 0 \text{ eða } k = n \\ f(n-1, k-1) + f(n-1, k) & \text{annars.} \end{cases}$$

- ▶ Við höfum svo að $f(n, k) = \binom{n}{k}$.

- ▶ Látum því

$$f(n, k) = \begin{cases} 0, & \text{ef } n < 1 \\ 0, & \text{ef } k < 0 \\ 0, & \text{ef } k > n \\ 1, & \text{ef } k = 0 \text{ eða } k = n \\ f(n-1, k-1) + f(n-1, k) & \text{annars.} \end{cases}$$

- ▶ Við höfum svo að $f(n, k) = \binom{n}{k}$.
- ▶ Útfærslan notar ofansækna kvika bestun.

```
6  || d[MAXN][MAXN], m;
7  || dp_lookup(|| x, || y)
8  {
9      if (d[x][y] != -1) return d[x][y];
10     if (x < 0 || y < 0 || y > x) return 0;
11     if (y == 0 || y == x) return 1;
12     return d[x][y] = (dp_lookup(x - 1, y - 1) + dp_lookup(x - 1, y))%m;
13 }
```

- ▶ Sjáum að það eru n^2 stöður.

- ▶ Sjáum að það eru n^2 stöður.
- ▶ Við reiknum hverja stöðu í $\mathcal{O}(\quad)$ tíma, svo við getum svarað q fyrirspurnum í $\mathcal{O}(\quad)$ tíma.

- ▶ Sjáum að það eru n^2 stöður.
- ▶ Við reiknum hverja stöðu í $\mathcal{O}(1)$ tíma, svo við getum svarað q fyrirspurnum í $\mathcal{O}(\quad)$ tíma.

- ▶ Sjáum að það eru n^2 stöður.
- ▶ Við reiknum hverja stöðu í $\mathcal{O}(1)$ tíma, svo við getum svarað q fyrirspurnum í $\mathcal{O}(n^2 + q)$ tíma.

- ▶ Sjáum að það eru n^2 stöður.
- ▶ Við reiknum hverja stöðu í $\mathcal{O}(1)$ tíma, svo við getum svarað q fyrirspurnum í $\mathcal{O}(n^2 + q)$ tíma.
- ▶ Það sem forritið okkar er í rauninni að gera er að reikna gildin í þríhyrningin Pascals.

- ▶ Sjáum að það eru n^2 stöður.
- ▶ Við reiknum hverja stöðu í $\mathcal{O}(1)$ tíma, svo við getum svarað q fyrirspurnum í $\mathcal{O}(n^2 + q)$ tíma.
- ▶ Það sem forritið okkar er í rauninni að gera er að reikna gildin í þríhyrningin Pascals.
- ▶ Þekkt er að k -ta talan í n -tu línu þríhyrnings Pascals er $\binom{n}{k}$.

Þríhyrningur Pascals

				1					
			1		1				
		1		2		1			
	1		3		3		1		
	1	4		6		4		1	
	1	5	10		10	5		1	
	1	6	15	20		15	6		1
	1	7	21	35	35	21	7		1
1	8	28	56	70	56	28	8		1

Fjöldi umhverfinga í umröðun

- ▶ Látum σ vera umröðun á $\{1, \dots, n\}$.

Fjöldi umhverfinga í umröðun

- ▶ Látum σ vera umröðun á $\{1, \dots, n\}$.
- ▶ Þá kallast par $(i, j) \in \{1, \dots, n\} \times \{1, \dots, n\}$ þannig að $i < j$ og $\sigma(i) > \sigma(j)$, *umhverfing* (e. *inversion*) í σ .

Fjöldi umhverfinga í umröðun

- ▶ Látum σ vera umröðun á $\{1, \dots, n\}$.
- ▶ Þá kallast par $(i, j) \in \{1, \dots, n\} \times \{1, \dots, n\}$ þannig að $i < j$ og $\sigma(i) > \sigma(j)$, *umhverfing* (e. *inversion*) í σ .
- ▶ Látum a tákna n staka lista þannig að j -ta stak listans sé $\sigma(j)$.

Fjöldi umhverfinga í umröðun

- ▶ Látum σ vera umröðun á $\{1, \dots, n\}$.
- ▶ Þá kallast par $(i, j) \in \{1, \dots, n\} \times \{1, \dots, n\}$ þannig að $i < j$ og $\sigma(i) > \sigma(j)$, *umhverfing* (e. *inversion*) í σ .
- ▶ Látum a tákna n staka lista þannig að j -ta stak listans sé $\sigma(j)$.
- ▶ Svona táknum við iðulega umraðanir þegar við útfærum lausnir.

Fjöldi umhverfinga í umröðun

- ▶ Látum σ vera umröðun á $\{1, \dots, n\}$.
- ▶ Þá kallast par $(i, j) \in \{1, \dots, n\} \times \{1, \dots, n\}$ þannig að $i < j$ og $\sigma(i) > \sigma(j)$, *umhverfing* (e. *inversion*) í σ .
- ▶ Látum a tákna n staka lista þannig að j -ta stak listans sé $\sigma(j)$.
- ▶ Svona táknum við iðulega umraðanir þegar við útfærum lausnir.
- ▶ Gerum nú ráð fyrir að eina leiðin okkar til að breyta a er að skipta á aðlægum stökum.

Fjöldi umhverfinga í umröðun

- ▶ Látum σ vera umröðun á $\{1, \dots, n\}$.
- ▶ Þá kallast par $(i, j) \in \{1, \dots, n\} \times \{1, \dots, n\}$ þannig að $i < j$ og $\sigma(i) > \sigma(j)$, *umhverfing* (e. *inversion*) í σ .
- ▶ Látum a tákna n staka lista þannig að j -ta stak listans sé $\sigma(j)$.
- ▶ Svona táknum við iðulega umraðanir þegar við útfærum lausnir.
- ▶ Gerum nú ráð fyrir að eina leiðin okkar til að breyta a er að skipta á aðlægum stökum.
- ▶ Hvað tekur það minnst margar aðgerðir að raða listanum?

Fjöldi umhverfinga í umröðun

- ▶ Látum σ vera umröðun á $\{1, \dots, n\}$.
- ▶ Þá kallast par $(i, j) \in \{1, \dots, n\} \times \{1, \dots, n\}$ þannig að $i < j$ og $\sigma(i) > \sigma(j)$, *umhverfing* (e. *inversion*) í σ .
- ▶ Látum a tákna n staka lista þannig að j -ta stak listans sé $\sigma(j)$.
- ▶ Svona táknum við iðulega umraðanir þegar við útfærum lausnir.
- ▶ Gerum nú ráð fyrir að eina leiðin okkar til að breyta a er að skipta á aðlægum stökum.
- ▶ Hvað tekur það minnst margar aðgerðir að raða listanum?
- ▶ Það vill svo til að fjöldi umhverfinga í umröðununni er einmitt fjöldi aðgerða sem þarf til að raða listanum.

Fjöldi umhverfinga í umröðun

- ▶ Látum σ vera umröðun á $\{1, \dots, n\}$.
- ▶ Þá kallast par $(i, j) \in \{1, \dots, n\} \times \{1, \dots, n\}$ þannig að $i < j$ og $\sigma(i) > \sigma(j)$, *umhverfing* (e. *inversion*) í σ .
- ▶ Látum a tákna n staka lista þannig að j -ta stak listans sé $\sigma(j)$.
- ▶ Svona táknum við iðulega umraðanir þegar við útfærum lausnir.
- ▶ Gerum nú ráð fyrir að eina leiðin okkar til að breyta a er að skipta á aðlægum stökum.
- ▶ Hvað tekur það minnst margar aðgerðir að raða listanum?
- ▶ Það vill svo til að fjöldi umhverfinga í umröðununni er einmitt fjöldi aðgerða sem þarf til að raða listanum.
- ▶ Við getum notað okkur þetta til að finna fjölda umhverfinga.

Fjöldi umhverfinga í umröðun

- ▶ Ein leið til að gera þetta er að færa fyrst minnsta stakið fremst, síðan næst minnsta stakið næst fremst og svo framvegis.

Fjöldi umhverfinga í umröðun

- ▶ Ein leið til að gera þetta er að færa fyrst minnsta stakið fremst, síðan næst minnsta stakið næst fremst og svo framvegis.
- ▶ Þetta tekur $\mathcal{O}(n^2)$ tíma.

Fjöldi umhverfinga í umröðun

- ▶ Ein leið til að gera þetta er að færa fyrst minnsta stakið fremst, síðan næst minnsta stakið næst fremst og svo framvegis.
- ▶ Þetta tekur $\mathcal{O}(n^2)$ tíma.

Fjöldi umhverfinga í umröðun

- ▶ Ein leið til að gera þetta er að færa fyrst minnsta stakið fremst, síðan næst minnsta stakið næst fremst og svo framvegis.
- ▶ Þetta tekur $\mathcal{O}(n^2)$ tíma.
- ▶ Þetta er því of hægt of $n > 10^4$.

Fjöldi umhverfinga í umröðun

- ▶ Ein leið til að gera þetta er að færa fyrst minnsta stakið fremst, síðan næst minnsta stakið næst fremst og svo framvegis.
- ▶ Þetta tekur $\mathcal{O}(n^2)$ tíma.
- ▶ Þetta er því of hægt of $n > 10^4$.
- ▶ Við getum þó notað biltré til að gera þetta hraðara.

Fjöldi umhverfinga í umröðun

- ▶ Ein leið til að gera þetta er að færa fyrst minnsta stakið fremst, síðan næst minnsta stakið næst fremst og svo framvegis.
- ▶ Þetta tekur $\mathcal{O}(n^2)$ tíma.
- ▶ Þetta er því of hægt of $n > 10^4$.
- ▶ Við getum þó notað biltré til að gera þetta hraðara.
- ▶ Tökum eftir að þegar við höfum sett stak á sinn stað getum við hætt að hugsa um það.

Fjöldi umhverfinga í umröðun

- ▶ Ein leið til að gera þetta er að færa fyrst minnsta stakið fremst, síðan næst minnsta stakið næst fremst og svo framvegis.
- ▶ Þetta tekur $\mathcal{O}(n^2)$ tíma.
- ▶ Þetta er því of hægt of $n > 10^4$.
- ▶ Við getum þó notað biltré til að gera þetta hraðara.
- ▶ Tökum eftir að þegar við höfum sett stak á sinn stað getum við hætt að hugsa um það.
- ▶ Við þurfum í raun að sagt til um hversu mörg stök í listanum eru fyrir framan það stak sem við viljum færa (við teljum ekki þau stök sem eru komin á sinn stað).

Fjöldi umhverfinga í umröðun

- ▶ Ein leið til að gera þetta er að færa fyrst minnsta stakið fremst, síðan næst minnsta stakið næst fremst og svo framvegis.
- ▶ Þetta tekur $\mathcal{O}(n^2)$ tíma.
- ▶ Þetta er því of hægt of $n > 10^4$.
- ▶ Við getum þó notað biltré til að gera þetta hraðara.
- ▶ Tökum eftir að þegar við höfum sett stak á sinn stað getum við hætt að hugsa um það.
- ▶ Við þurfum í raun að sagt til um hversu mörg stök í listanum eru fyrir framan það stak sem við viljum færa (við teljum ekki þau stök sem eru komin á sinn stað).
- ▶ Við byrjum því með 1 í hverju staki í biltrénu okkar.

Fjöldi umhverfinga í umröðun

- ▶ Ein leið til að gera þetta er að færa fyrst minnsta stakið fremst, síðan næst minnsta stakið næst fremst og svo framvegis.
- ▶ Þetta tekur $\mathcal{O}(n^2)$ tíma.
- ▶ Þetta er því of hægt of $n > 10^4$.
- ▶ Við getum þó notað biltré til að gera þetta hraðara.
- ▶ Tökum eftir að þegar við höfum sett stak á sinn stað getum við hætt að hugsa um það.
- ▶ Við þurfum í raun að sagt til um hversu mörg stök í listanum eru fyrir framan það stak sem við viljum færa (við teljum ekki þau stök sem eru komin á sinn stað).
- ▶ Við byrjum því með 1 í hverju staki í biltrénu okkar.
- ▶ Þegar við höfum fært stak á sinn stað setjum við tilheyrandi gildi sem 0.

Fjöldi umhverfinga í umröðun

- ▶ Ein leið til að gera þetta er að færa fyrst minnsta stakið fremst, síðan næst minnsta stakið næst fremst og svo framvegis.
- ▶ Þetta tekur $\mathcal{O}(n^2)$ tíma.
- ▶ Þetta er því of hægt of $n > 10^4$.
- ▶ Við getum þó notað biltré til að gera þetta hraðara.
- ▶ Tökum eftir að þegar við höfum sett stak á sinn stað getum við hætt að hugsa um það.
- ▶ Við þurfum í raun að sagt til um hversu mörg stök í listanum eru fyrir framan það stak sem við viljum færa (við teljum ekki þau stök sem eru komin á sinn stað).
- ▶ Við byrjum því með 1 í hverju staki í biltrénu okkar.
- ▶ Þegar við höfum fært stak á sinn stað setjum við tilheyrandi gildi sem 0.
- ▶ Summa fyrst $j - 1$ stakanna í trénu er því fjöldi staka sem j -ta stakið í listanum þarf að sipta á til að komast á sinn stað.

Hæga aðferðin

4 1 6 5 7 3 2

svar: 0

Hæga aðferðin

```
4 1 6 5 7 3 2
```

```
  ^
```

```
svar: 0
```

Hæga aðferðin

```
1 4 6 5 7 3 2
```

```
^
```

```
svar: 1
```

Hæga aðferðin

```
1 4 6 5 7 3 2
           ^
```

```
svar: 1
```

Hæga aðferðin

```
1 4 6 5 7 2 3
```

^

```
svar: 2
```

Hæga aðferðin

1 4 6 5 2 7 3

^

svar: 3

Hæga aðferðin

```
1 4 6 2 5 7 3
```

```
  ^
```

```
svar: 4
```

Hæga aðferðin

```
1 4 2 6 5 7 3
```

```
  ^
```

```
svar: 5
```


Hæga aðferðin

```
1 2 4 6 5 7 3
```

```
^
```

```
svar: 6
```

Hæga aðferðin

```
1 2 4 6 5 7 3
          ^
```

```
svar: 6
```

Hæga aðferðin

```
1 2 4 6 5 3 7
      ^
```

```
svar: 7
```

Hæga aðferðin

```
1 2 4 6 3 5 7
      ^
```

```
svar: 8
```

Hæga aðferðin

```
1 2 4 3 6 5 7
```

```
^
```

```
svar: 9
```

Hæga aðferðin

```
1 2 3 4 6 5 7
    ^
```

```
svar: 10
```

Hæga aðferðin

1 2 3 4 6 5 7

^

svar: 10

Hæga aðferðin

```
1 2 3 4 6 5 7
      ^
```

```
svar: 10
```


Hæga aðferðin

```
1 2 3 4 5 6 7
      ^
```

```
svar: 11
```

Hæga aðferðin

```
1 2 3 4 5 6 7
      ^
```

```
svar: 11
```

Hæga aðferðin

```
1 2 3 4 5 6 7
      ^
```

```
svar: 11
```

Hæga aðferðin

```
1 2 3 4 5 6 7
```

```
svar: 11
```

Hraða aðferðin

```
listinn: 4 1 6 5 7 3 2
```

```
biltred: 1 1 1 1 1 1 1
```

```
svar: 0
```

Hraða aðferðin

```
listinn: 4 1 6 5 7 3 2
```

^

```
biltred: 1 1 1 1 1 1 1
```

```
svar: 0
```

Hraða aðferðin

```
listinn: 4 1 6 5 7 3 2
          ^
biltred: 1 1 1 1 1 1 1
          | |
          svar: 0
```

Hraða aðferðin

```
listinn: 4 x 6 5 7 3 2
```

```
biltred: 1 0 1 1 1 1
```

```
svar: 1
```


Hraða aðferðin

```
listinn: 4 x 6 5 7 3 2  
          ^
```

```
biltred: 1 0 1 1 1 1
```

```
svar: 1
```

Hraða aðferðin

```
listinn: 4 x 6 5 7 3 2
              ^
biltred: 1 0 1 1 1 1 1
          |           |
          svar:  1
```

Hraða aðferðin

```
listinn: 4 x 6 5 7 3 x
```

```
biltred: 1 0 1 1 1 1 0
```

```
svar: 6
```

Hraða aðferðin

```
listinn: 4 x 6 5 7 3 x  
          ^
```

```
biltred: 1 0 1 1 1 1 0
```

```
svar: 6
```

Hraða aðferðin

```
listinn: 4 x 6 5 7 3 x
           ^
biltred: 1 0 1 1 1 1 0
         |         |
         svar: 6
```

Hraða aðferðin

```
listinn: 4 x 6 5 7 x x
```

```
biltred: 1 0 1 1 1 0 0
```

```
svar: 10
```

Hraða aðferðin

```
listinn: 4 x 6 5 7 x x  
        ^
```

```
biltred: 1 0 1 1 1 0 0
```

```
svar: 10
```

Hraða aðferðin

```
listinn: 4 x 6 5 7 x x
```

^

```
biltred: 1 0 1 1 1 0 0
```

|

```
svar: 10
```


Hraða aðferðin

```
listinn: x x 6 5 7 x x
```

```
biltred: 0 0 1 1 1 0 0
```

```
svar: 10
```

Hraða aðferðin

```
listinn: x x 6 5 7 x x  
          ^
```

```
biltred: 0 0 1 1 1 0 0
```

```
svar: 10
```

Hraða aðferðin

```
listinn: x x 6 5 7 x x
```

^

```
biltred: 0 0 1 1 1 0 0
```

```
|      |
```

```
svar: 10
```

Hraða aðferðin

```
listinn: x x 6 x 7 x x
```

```
biltred: 0 0 1 0 1 0 0
```

```
svar: 11
```

Hraða aðferðin

```
listinn: x x 6 x 7 x x  
          ^
```

```
biltred: 0 0 1 0 1 0 0
```

```
svar: 11
```

Hraða aðferðin

```
listinn: x x 6 x 7 x x  
          ^
```

```
biltred: 0 0 1 0 1 0 0  
         |  |
```

```
svar: 11
```

Hraða aðferðin

```
listinn: x x x x 7 x x
```

```
biltred: 0 0 0 0 1 0 0
```

```
svar: 11
```

Hraða aðferðin

```
listinn: x x x x 7 x x  
           ^
```

```
biltred: 0 0 0 0 1 0 0
```

```
svar: 11
```


Hraða aðferðin

```
listinn: x x x x 7 x x
           ^
biltred: 0 0 0 0 1 0 0
          |       |
          svar: 11
```

Hraða aðferðin

```
listinn: x x x x x x x
```

```
biltred: 0 0 0 0 0 0 0
```

```
svar: 11
```

```

38 ll invnum(ll* a, ll n)
39 { // Finnur fjölda umhverfinga i umroduninni sem svarar til listans |a|.
40   // G.r.f. ad |a| inni haldi 0, 1, ..., n - 1.
41   ll b[n], i, r = 0;
42   pn = n;
43   rep(i, MAXN*5) p[i] = 0;
44   rep(i, n) update(i, 1);
45   rep(i, n) b[a[i]] = i;
46   rep(i, n) r += b[i] != 0 ? query(0, b[i] - 1) : 0, update(b[i], -1);
47   return r;
48 }

```

- ▶ Hver aðgerð á biltrénu er framkvæmd í $\mathcal{O}(\quad)$ tíma.

- ▶ Hver aðgerð á biltrénu er framkvæmd í $\mathcal{O}(\log n)$ tíma.

- ▶ Hver aðgerð á biltrénu er framkvæmd í $\mathcal{O}(\log n)$ tíma.
- ▶ Við finnum því fjölda umhverfinga í $\mathcal{O}(\quad)$ tíma.

- ▶ Hver aðgerð á biltrénu er framkvæmd í $\mathcal{O}(\log n)$ tíma.
- ▶ Við finnum því fjölda umhverfinga í $\mathcal{O}(n \log n)$ tíma.
- ▶ Svo við getum auðveldlega fundið fjölda umhverfinga fyrir $n < 10^6$.

- ▶ Hver aðgerð á biltrénu er framkvæmd í $\mathcal{O}(\log n)$ tíma.
- ▶ Við finnum því fjölda umhverfinga í $\mathcal{O}(n \log n)$ tíma.
- ▶ Svo við getum auðveldlega fundið fjölda umhverfinga fyrir $n < 10^6$.
- ▶ Takið þó eftir að fjöldinn gæti orðið $n \cdot (n - 1)/2$, svo það þarf alltaf að nota long long.

- ▶ Hvað gerum við þó ef tölurnar sem eru gefnar eru ekki $\{1, 2, \dots, n\}$ (líkt og í dæminu *Ultra-QuickSort*)?

- ▶ Hvað gerum við þó ef tölurnar sem eru gefnar eru ekki $\{1, 2, \dots, n\}$ (líkt og í dæminu *Ultra-QuickSort*)?
- ▶ Þá virkar oft að skipta minnstu tölunni út fyrir 1, næst minnstu fyrir 2 og svo framvegis.

- ▶ Hvað gerum við þó ef tölurnar sem eru gefnar eru ekki $\{1, 2, \dots, n\}$ (líkt og í dæminu *Ultra-QuickSort*)?
- ▶ Þá virkar oft að skipta minnstu tölunni út fyrir 1, næst minnstu fyrir 2 og svo framvegis.
- ▶ Ef það eru endurtekiningar þarf að passa að öllum eins tölum sé breytt í eins tölur.

- ▶ Hvað gerum við þó ef tölurnar sem eru gefnar eru ekki $\{1, 2, \dots, n\}$ (líkt og í dæminu *Ultra-QuickSort*)?
- ▶ Þá virkar oft að skipta minnstu tölunni út fyrir 1, næst minnstu fyrir 2 og svo framvegis.
- ▶ Ef það eru endurtekiningar þarf að passa að öllum eins tölum sé breytt í eins tölur.
- ▶ Þetta má gera með því að raða.

- ▶ Hvað gerum við þó ef tölurnar sem eru gefnar eru ekki $\{1, 2, \dots, n\}$ (líkt og í dæminu *Ultra-QuickSort*)?
- ▶ Þá virkar oft að skipta minnstu tölunni út fyrir 1, næst minnstu fyrir 2 og svo framvegis.
- ▶ Ef það eru endurtekiningar þarf að passa að öllum eins tölum sé breytt í eins tölur.
- ▶ Þetta má gera með því að raða.
- ▶ Við röðum fyrst tvenndunum (a_j, j) , þar sem a_j táknar j -ta stakið í listanum okkar, eftir fyrsta stakinu.

- ▶ Hvað gerum við þó ef tölurnar sem eru gefnar eru ekki $\{1, 2, \dots, n\}$ (líkt og í dæminu *Ultra-QuickSort*)?
- ▶ Þá virkar oft að skipta minnstu tölunni út fyrir 1, næst minnstu fyrir 2 og svo framvegis.
- ▶ Ef það eru endurtekiningar þarf að passa að öllum eins tölum sé breytt í eins tölur.
- ▶ Þetta má gera með því að raða.
- ▶ Við röðum fyrst tvenndunum (a_j, j) , þar sem a_j táknar j -ta stakið í listanum okkar, eftir fyrsta stakinu.
- ▶ Við getum þá labbað í gegn og breytt öllum tölunum.

- ▶ Hvað gerum við þó ef tölurnar sem eru gefnar eru ekki $\{1, 2, \dots, n\}$ (líkt og í dæminu *Ultra-QuickSort*)?
- ▶ Þá virkar oft að skipta minnstu tölunni út fyrir 1, næst minnstu fyrir 2 og svo framvegis.
- ▶ Ef það eru endurtekiningar þarf að passa að öllum eins tölum sé breytt í eins tölur.
- ▶ Þetta má gera með því að raða.
- ▶ Við röðum fyrst tvenndunum (a_j, j) , þar sem a_j táknar j -ta stakið í listanum okkar, eftir fyrsta stakinu.
- ▶ Við getum þá labbað í gegn og breytt öllum tölunum.
- ▶ Að lokum röðum tvenndunum aftur eftir seinna stakinu.

```

4 typedef struct {int x, y;} ii;
5
6 int cmpx(const void* p1, const void* p2) {return ((ii*)p1)->x - ((ii*)p2)->x;}
7 int cmpy(const void* p1, const void* p2) {return ((ii*)p1)->y - ((ii*)p2)->y;}
8
9 void compress(int* a, int n)
10 {
11     int i, j, x, k;
12     ii b[n];
13     rep(i, n) b[i].x = a[i], b[i].y = i;
14     qsort(b, n, sizeof(b[0]), cmpx);
15     for (i = k = 0; i < n; i = j, k++)
16         for (j = i, x = b[i].x; b[j].x == x; j++)
17             b[j].x = k;
18     qsort(b, n, sizeof(b[0]), cmpy);
19     rep(i, n) a[i] = b[i].x;
20 }

```


- ▶ Reikniritið keyrir í $\mathcal{O}(\quad)$ tíma því við þurfum að raða.

- ▶ Reikniritið keyrir í $\mathcal{O}(n \log n)$ tíma því við þurfum að raða.

- ▶ Mörg talningarfræði dæmi má smækka á þægilegan máta.

- ▶ Mörg talningarfræði dæmi má smækka á þægilegan máta.
- ▶ Ef við látum, til dæmis, $f(n)$ tákna fjölda hlutmengja í mengin $\{1, 2, \dots, n\}$ þá höfum við að

$$f(n) = \begin{cases} 1, & \text{ef } n = 0 \\ 2 \cdot f(n-1), & \text{annars.} \end{cases}$$

- ▶ Mörg talningarfræði dæmi má smækka á þægilegan máta.
- ▶ Ef við látum, til dæmis, $f(n)$ tákna fjölda hlutmengja í mengin $\{1, 2, \dots, n\}$ þá höfum við að

$$f(n) = \begin{cases} 1, & \text{ef } n = 0 \\ 2 \cdot f(n-1), & \text{annars.} \end{cases}$$

- ▶ Þetta gildir því það eru $f(n-1)$ hlutmengi sem innihalda n og $f(n-1)$ hlutmengi sem innihalda ekki n .

- ▶ Mörg talningarfræði dæmi má smækka á þægilegan máta.
- ▶ Ef við látum, til dæmis, $f(n)$ tákna fjölda hlutmengja í mengin $\{1, 2, \dots, n\}$ þá höfum við að

$$f(n) = \begin{cases} 1, & \text{ef } n = 0 \\ 2 \cdot f(n-1), & \text{annars.} \end{cases}$$

- ▶ Þetta gildir því það eru $f(n-1)$ hlutmengi sem innihalda n og $f(n-1)$ hlutmengi sem innihalda ekki n .
- ▶ Ef við getum smækkað dæmin á þennan máta má svo nota til að leysa þau.

- ▶ Mörg talningarfræði dæmi má smækka á þægilegan máta.
- ▶ Ef við látum, til dæmis, $f(n)$ tákna fjölda hlutmengja í mengin $\{1, 2, \dots, n\}$ þá höfum við að

$$f(n) = \begin{cases} 1, & \text{ef } n = 0 \\ 2 \cdot f(n-1), & \text{annars.} \end{cases}$$

- ▶ Þetta gildir því það eru $f(n-1)$ hlutmengi sem innihalda n og $f(n-1)$ hlutmengi sem innihalda ekki n .
- ▶ Ef við getum smækkað dæmin á þennan máta má svo nota kvika bestun til að leysa þau.

- ▶ Mörg talningarfræði dæmi má smækka á þægilegan máta.
- ▶ Ef við látum, til dæmis, $f(n)$ tákna fjölda hlutmengja í mengin $\{1, 2, \dots, n\}$ þá höfum við að

$$f(n) = \begin{cases} 1, & \text{ef } n = 0 \\ 2 \cdot f(n-1), & \text{annars.} \end{cases}$$

- ▶ Þetta gildir því það eru $f(n-1)$ hlutmengi sem innihalda n og $f(n-1)$ hlutmengi sem innihalda ekki n .
- ▶ Ef við getum smækkað dæmin á þennan máta má svo nota kvika bestun til að leysa þau.
- ▶ Oft þarf að bæta við vídd til að halda utan um önnur gögn.

- ▶ Mörg talningarfræði dæmi má smækka á þægilegan máta.
- ▶ Ef við látum, til dæmis, $f(n)$ tákna fjölda hlutmengja í mengin $\{1, 2, \dots, n\}$ þá höfum við að

$$f(n) = \begin{cases} 1, & \text{ef } n = 0 \\ 2 \cdot f(n-1), & \text{annars.} \end{cases}$$

- ▶ Þetta gildir því það eru $f(n-1)$ hlutmengi sem innihalda n og $f(n-1)$ hlutmengi sem innihalda ekki n .
- ▶ Ef við getum smækkað dæmin á þennan máta má svo nota kvika bestun til að leysa þau.
- ▶ Oft þarf að bæta við vídd til að halda utan um önnur gögn.
- ▶ Tökum dæmi.

- Gerum ráð fyrir að þú sért með n spilastokka.

- ▶ Gerum ráð fyrir að þú sért með n spilastokka.
- ▶ Hver stokkur inniheldur k spil, númeruð frá 1 og upp í k .

- ▶ Gerum ráð fyrir að þú sért með n spilastokka.
- ▶ Hver stokkur inniheldur k spil, númeruð frá 1 og upp í k .
- ▶ Á hversu marga vegu getur þú valið eitt spil úr hverjum stokk þannig að summa spilanna sé nákvæmlega m ?

- ▶ Gerum ráð fyrir að þú sért með n spilastokka.
- ▶ Hver stokkur inniheldur k spil, númeruð frá 1 og upp í k .
- ▶ Á hversu marga vegu getur þú valið eitt spil úr hverjum stökk þannig að summa spilanna sé nákvæmlega m ?
- ▶ Þar sem þessi tala getur verið mjög stór skila reikna hana mod $10^9 + 7$.

- ▶ Festum fyrsta spilið sem x .

- ▶ Festum fyrsta spilið sem x .
- ▶ Við eigum þá eftir $n - 1$ stökk og viljum fá summuna $m - x$ úr þeim.

- ▶ Festum fyrsta spilið sem x .
- ▶ Við eigum þá eftir $n - 1$ stökk og viljum fá summuna $m - x$ úr þeim.
- ▶ Með öðrum orðum höfum við smækkað dæmið.

- ▶ Festum fyrsta spilið sem x .
- ▶ Við eigum þá eftir $n - 1$ stökk og viljum fá summuna $m - x$ úr þeim.
- ▶ Með öðrum orðum höfum við smækkað dæmið.
- ▶ Við skilgreinum því

$$f(x, y) = \begin{cases} 1, & \text{ef } n = 0 \text{ og } y = 0 \\ 0, & \text{ef } n = 0 \text{ og } y \neq 0 \\ \sum_{j=1}^k f(x-1, y-j), & \text{annars.} \end{cases}$$

- ▶ Festum fyrsta spilið sem x .
- ▶ Við eigum þá eftir $n - 1$ stökk og viljum fá summuna $m - x$ úr þeim.
- ▶ Með öðrum orðum höfum við smækkað dæmið.
- ▶ Við skilgreinum því

$$f(x, y) = \begin{cases} 1, & \text{ef } n = 0 \text{ og } y = 0 \\ 0, & \text{ef } n = 0 \text{ og } y \neq 0 \\ \sum_{j=1}^k f(x-1, y-j), & \text{annars.} \end{cases}$$

- ▶ Nú gildir að $f(x, y)$ er fjöldi leiða til að fá summuna y með x stökkum.

- ▶ Festum fyrsta spilið sem x .
- ▶ Við eigum þá eftir $n - 1$ stökk og viljum fá summuna $m - x$ úr þeim.
- ▶ Með öðrum orðum höfum við smækkað dæmið.
- ▶ Við skilgreinum því

$$f(x, y) = \begin{cases} 1, & \text{ef } n = 0 \text{ og } y = 0 \\ 0, & \text{ef } n = 0 \text{ og } y \neq 0 \\ \sum_{j=1}^k f(x-1, y-j), & \text{annars.} \end{cases}$$

- ▶ Nú gildir að $f(x, y)$ er fjöldi leiða til að fá summuna y með x stökkum.
- ▶ Við getum svo útfært þetta eins of við höfum verið að útfæra kvikva bestun.

```

7  ll dp_lookup(ll x, ll y)
8  {
9      if (y < 0) return 0;
10     if (d[x][y] != -1) return d[x][y];
11     if (x == 0) return y == 0 ? 1 : 0;
12     ll i;
13     d[x][y] = 0;
14     rep(i, k) d[x][y] = (d[x][y] + dp_lookup(x - 1, y - i - 1))%MOD;
15     return d[x][y];
16 }

```

- ▶ Við höfum $n \cdot m$ stöður og hverju stöðu má reikna í $\mathcal{O}(\quad)$ tíma.

- ▶ Við höfum $n \cdot m$ stöður og hverju stöðu má reikna í $\mathcal{O}(k)$ tíma.

- ▶ Við höfum $n \cdot m$ stöður og hverju stöðu má reikna í $\mathcal{O}(k)$ tíma.
- ▶ Svo tímaflækjan er $\mathcal{O}(\quad)$.

- ▶ Við höfum $n \cdot m$ stöður og hverju stöðu má reikna í $\mathcal{O}(k)$ tíma.
- ▶ Svo tímaflækjan er $\mathcal{O}(n \cdot m \cdot k)$.

- ▶ Við höfum $n \cdot m$ stöður og hverju stöðu má reikna í $\mathcal{O}(k)$ tíma.
- ▶ Svo tímaflækjan er $\mathcal{O}(n \cdot m \cdot k)$.
- ▶ Nú getur m ekki verið stærra en $n \cdot k$ svo við fáum $\mathcal{O}(\quad)$.

- ▶ Við höfum $n \cdot m$ stöður og hverju stöðu má reikna í $\mathcal{O}(k)$ tíma.
- ▶ Svo tímaflækjan er $\mathcal{O}(n \cdot m \cdot k)$.
- ▶ Nú getur m ekki verið stærra en $n \cdot k$ svo við fáum $\mathcal{O}(n^2 \cdot k^2)$.

Flykjaaðgerðir

- ▶ Í stærðfræði þýðir „fylki” annað en í tölvunarfræði.

Flykjaaðgerðir

- ▶ Í stærðfræði þýðir „fylki” annað en í tölvunarfræði.
- ▶ Í stærðfræði er *fylki* (e. *matrix*) tvívíð uppröðun á tölum.

Flykjaaðgerðir

- ▶ Í stærðfræði þýðir „fylki” annað en í tölvunarfræði.
- ▶ Í stærðfræði er *fylki* (e. *matrix*) tvívíð uppröðun á tölum.
- ▶ Fylkið er sagt vera $n \times m$ ef það hefur n línur og m dálka.

Flykjaaðgerðir

- ▶ Í stærðfræði þýðir „fylki” annað en í tölvunarfræði.
- ▶ Í stærðfræði er *fylki* (e. *matrix*) tvívíð uppröðun á tölum.
- ▶ Fylkið er sagt vera $n \times m$ ef það hefur n línur og m dálka.
- ▶ Dæmi um 2×3 fylki er

$$\begin{pmatrix} 2 & -1 & 0 \\ 0 & 14 & 0 \end{pmatrix}.$$

Flykjaaðgerðir

- ▶ Í stærðfræði þýðir „fylki” annað en í tölvunarfræði.
- ▶ Í stærðfræði er *fylki* (e. *matrix*) tvívíð uppröðun á tölum.
- ▶ Fylkið er sagt vera $n \times m$ ef það hefur n línur og m dálka.
- ▶ Dæmi um 2×3 fylki er

$$\begin{pmatrix} 2 & -1 & 0 \\ 0 & 14 & 0 \end{pmatrix}.$$

- ▶ Við táknum yfirleitt stakið í línu j og dálki k í fylki A með A_{jk} .

Flykjaaðgerðir

- ▶ Í stærðfræði þýðir „fylki” annað en í tölvunarfræði.
- ▶ Í stærðfræði er *fylki* (e. *matrix*) tvívíð uppröðun á tölum.
- ▶ Fylkið er sagt vera $n \times m$ ef það hefur n línur og m dálka.
- ▶ Dæmi um 2×3 fylki er

$$\begin{pmatrix} 2 & -1 & 0 \\ 0 & 14 & 0 \end{pmatrix}.$$

- ▶ Við táknum yfirleitt stakið í línu j og dálki k í fylki A með A_{jk} .
- ▶ Flyki í stærfræði er því eins og tvívítt fylki í tölvunarfærði.

Flykjaaðgerðir

- ▶ Þegar við viljum geyma stærðfræði fylki í tölvu notum við oftast tvívítt tölvunarfræði fylki.

Flykjaaðgerðir

- ▶ Þegar við viljum geyma stærðfræði fylki í tölvu notum við oftast tvívítt tölvunarfræði fylki.
- ▶ Við höfum þá að A_{jk} svarar til $a[j][k]$.

Flykjaaðgerðir

- ▶ Þegar við viljum geyma stærðfræði fylki í tölvu notum við oftast tvívítt tölvunarfræði fylki.
- ▶ Við höfum þá að A_{jk} svarar til $a[j][k]$.
- ▶ Ef A er $n \times m$ fylki getum við líka geymt það með einvíðu tölvunarfræði fylki með samsvöruninni A_{jk} við $a[j*m + k]$.

Flykjaaðgerðir

- ▶ Við getum lagt saman fylki af sömu stærð stakvíst, það er að segja $(A + B)_{jk} = A_{jk} + B_{jk}$.

Flykjaaðgerðir

- ▶ Við getum lagt saman fylki af sömu stærð stakvíst, það er að segja $(A + B)_{jk} = A_{jk} + B_{jk}$.
- ▶ Frádráttur virkar eins.

Flykjaaðgerðir

- ▶ Við getum lagt saman fylki af sömu stærð stakvíst, það er að segja $(A + B)_{jk} = A_{jk} + B_{jk}$.
- ▶ Frádráttur virkar eins.
- ▶ Við margföldum saman fylki af stærð $n \times m$ og $m \times r$ með

$$(A \cdot B)_{jk} = \sum_{l=1}^m A_{jl} \cdot B_{lk}.$$

Flykjaaðgerðir

- ▶ Við getum lagt saman fylki af sömu stærð stakvíst, það er að segja $(A + B)_{jk} = A_{jk} + B_{jk}$.
- ▶ Frádráttur virkar eins.
- ▶ Við margföldum saman fylki af stærð $n \times m$ og $m \times r$ með

$$(A \cdot B)_{jk} = \sum_{l=1}^m A_{jl} \cdot B_{lk}.$$

- ▶ Oftast erum við að vinna með fylki sem eru af stærð $n \times n$.

Flykjaaðgerðir

- ▶ Við getum lagt saman fylki af sömu stærð stakvíst, það er að segja $(A + B)_{jk} = A_{jk} + B_{jk}$.
- ▶ Frádráttur virkar eins.
- ▶ Við margföldum saman fylki af stærð $n \times m$ og $m \times r$ með

$$(A \cdot B)_{jk} = \sum_{l=1}^m A_{jl} \cdot B_{lk}.$$

- ▶ Oftast erum við að vinna með fylki sem eru af stærð $n \times n$.
- ▶ Slík fylki kallast *ferningsfylki*.

Flykjaaðgerðir

- ▶ Við getum lagt saman fylki af sömu stærð stakvíst, það er að segja $(A + B)_{jk} = A_{jk} + B_{jk}$.
- ▶ Frádráttur virkar eins.
- ▶ Við margföldum saman fylki af stærð $n \times m$ og $m \times r$ með

$$(A \cdot B)_{jk} = \sum_{l=1}^m A_{jl} \cdot B_{lk}.$$

- ▶ Oftast erum við að vinna með fylki sem eru af stærð $n \times n$.
- ▶ Slík fylki kallast *ferningsfylki*.
- ▶ Takið eftir að fylkið I , gefið með $I_{jk} = 0$ ef $j \neq k$ og $I_{jk} = 1$ annars, er margföldunarhlutleysa.

```

3 // Utfaersla a eindfoldum ferningsfylkjaadgerdum.
4
5 void addto(int* a, int* b, int n)
6 { // Baetir fylkinu |b| vid fylkid |a|. Eins og 'a += b'.
7     int i, j;
8     rep(i, n) rep(j, n) a[i*n + j] += b[i*n + j];
9 }
10
11 void subfrom(int* a, int* b, int n)
12 { // Dregur fylkid |b| fra fylkinu |a|. Eins og 'a -= b'.
13     int i, j;
14     rep(i, n) rep(j, n) a[i*n + j] -= b[i*n + j];
15 }
16
17 void multo(int* a, int* b, int n)
18 { // Eins og 'a *= b'.
19     int i, j, k, c[n][n];
20     rep(i, n) rep(j, n) c[i][j] = 0;
21     rep(i, n) rep(j, n) rep(k, n) c[i][j] += a[i*n + k]*b[k*n + j];
22     rep(i, n) rep(j, n) a[i*n + j] = c[i][j];
23 }

```

- ▶ Takið eftir að `multo(...)` hefur tímaflækju $\mathcal{O}(\quad)$.

- ▶ Takið eftir að `multo(...)` hefur tímaflækju $\mathcal{O}(n^3)$.

- ▶ Takið eftir að `multo(...)` hefur tímaflækju $\mathcal{O}(n^3)$.
- ▶ Ef A er $n \times n$ ferningsfylki getum við reinkað A^p .

- ▶ Takið eftir að `multo(...)` hefur tímaflækju $\mathcal{O}(n^3)$.
- ▶ Ef A er $n \times n$ ferningsfylki getum við reiknað A^p .
- ▶ Með því að nota deila og drottna aðferð, líkt og við gerðum í síðustu viku, getum við reiknað A^p í $\mathcal{O}(\quad)$ tíma.

- ▶ Takið eftir að `multo(...)` hefur tímaflækju $\mathcal{O}(n^3)$.
- ▶ Ef A er $n \times n$ ferningsfylki getum við reinkað A^p .
- ▶ Með því að nota deila og drottna aðferð, líkt og við gerðum í síðustu viku, getum við reiknað A^p í $\mathcal{O}(n^3 \log p)$ tíma.
- ▶ Þetta má nýta mikið í talningarfræði.

- ▶ Takið eftir að `multo(...)` hefur tímaflækju $\mathcal{O}(n^3)$.
- ▶ Ef A er $n \times n$ ferningsfylki getum við reinkað A^p .
- ▶ Með því að nota deila og drottna aðferð, líkt og við gerðum í síðustu viku, getum við reiknað A^p í $\mathcal{O}(n^3 \log p)$ tíma.
- ▶ Þetta má nýta mikið í talningarfræði.
- ▶ Tökum dæmi.

- ▶ Látum $G = (V, E)$ vera net og A vera nágrannflyki G .

- ▶ Látum $G = (V, E)$ vera net og A vera nágrannflyki G .
- ▶ Við munum nú leifa netinu að hafa fleiri en einn veg á milli tveggja hnúta.

- ▶ Látum $G = (V, E)$ vera net og A vera nágrannflyki G .
- ▶ Við munum nú leifa netinu að hafa fleiri en einn veg á milli tveggja hnúta.
- ▶ Þá segir A_{uv} hversu margir vegir liggja frá hnúta u til hnúts v .

- ▶ Látum $G = (V, E)$ vera net og A vera nágrannflyki G .
- ▶ Við munum nú leifa netinu að hafa fleiri en einn veg á milli tveggja hnúta.
- ▶ Þá segir A_{uv} hversu margir vegir liggja frá hnúta u til hnúts v .
- ▶ Sýna má með þrepuna að $(A^p)_{uv}$ segir okkur þá hversu margir vegir liggja á milli hnúts u og hnúts v , sem eru af lengd nákvæmlega p .

- ▶ Látum $G = (V, E)$ vera net og A vera nágrannflyki G .
- ▶ Við munum nú leifa netinu að hafa fleiri en einn veg á milli tveggja hnúta.
- ▶ Þá segir A_{uv} hversu margir vegir liggja frá hnúta u til hnúts v .
- ▶ Sýna má með þrepuna að $(A^p)_{uv}$ segir okkur þá hversu margir vegir liggja á milli hnúts u og hnúts v , sem eru af lengd nákvæmlega p .
- ▶ Takið eftir að við getum leyst þetta dæmi fyrir mjög stór p .

- ▶ Látum $G = (V, E)$ vera net og A vera nágrannflyki G .
- ▶ Við munum nú leifa netinu að hafa fleiri en einn veg á milli tveggja hnúta.
- ▶ Þá segir A_{uv} hversu margir vegir liggja frá hnúta u til hnúts v .
- ▶ Sýna má með þrepuna að $(A^p)_{uv}$ segir okkur þá hversu margir vegir liggja á milli hnúts u og hnúts v , sem eru af lengd nákvæmlega p .
- ▶ Takið eftir að við getum leyst þetta dæmi fyrir mjög stór p .
- ▶ Til dæmis væri lausnin okkar leifturhröð fyrir $n = 50$ og $p = 10^{18}$.

- Runa $(a_n)_{n \in \mathbb{N}}$ kallast *k-ta stigs línulega rakningarvensl* ef til eru c_1, \dots, c_k þannig að

$$a_n = \sum_{j=1}^k c_j \cdot a_{n-j},$$

fyrir öll n þannig að $n - k \in \mathbb{N}$.

- Runa $(a_n)_{n \in \mathbb{N}}$ kallast *k-ta stigs línulega rakningarvensl* ef til eru c_1, \dots, c_k þannig að

$$a_n = \sum_{j=1}^k c_j \cdot a_{n-j},$$

fyrir öll n þannig að $n - k \in \mathbb{N}$.

- Munið, til dæmis, að Fibonccí tölurnar eru línulega rakningarvensl með $a_1 = a_2 = c_1 = c_2 = 1$.

- Runa $(a_n)_{n \in \mathbb{N}}$ kallast *k-ta stigs línulega rakningarvensl* ef til eru c_1, \dots, c_k þannig að

$$a_n = \sum_{j=1}^k c_j \cdot a_{n-j},$$

fyrir öll n þannig að $n - k \in \mathbb{N}$.

- Munið, til dæmis, að Fibonccí tölurnar eru línulega rakningarvensl með $a_1 = a_2 = c_1 = c_2 = 1$.
- Látum *k-ta stigs línulega rakningarvensl* vera gefnin, líkt og að ofan.

► Skilgreinum nú flykið

$$M = \begin{pmatrix} c_1 & c_2 & \dots & c_k \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}.$$

- Skilgreinum nú flykið

$$M = \begin{pmatrix} c_1 & c_2 & \dots & c_k \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}.$$

- Takið eftir að

$$M \cdot \begin{pmatrix} a_{n-1} \\ a_{n-2} \\ \vdots \\ a_{n-k} \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^k c_j \cdot a_{n-j} \\ a_{n-1} \\ a_{n-2} \\ \vdots \\ a_{n-k+1} \end{pmatrix} = \begin{pmatrix} a_n \\ a_{n-1} \\ a_{n-2} \\ \vdots \\ a_{n-k+1} \end{pmatrix}.$$

- Skilgreinum nú flykið

$$M = \begin{pmatrix} c_1 & c_2 & \dots & c_k \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}.$$

- Takið eftir að

$$M \cdot \begin{pmatrix} a_{n-1} \\ a_{n-2} \\ \vdots \\ a_{n-k} \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^k c_j \cdot a_{n-j} \\ a_{n-1} \\ a_{n-2} \\ \vdots \\ a_{n-k+1} \end{pmatrix} = \begin{pmatrix} a_n \\ a_{n-1} \\ a_{n-2} \\ \vdots \\ a_{n-k+1} \end{pmatrix}.$$

- Svo við getum notað fylkið M til að fá næstu tölu í rakningarvenslunum.

- ▶ Við þurfum þó ekki að hætta þar.

- ▶ Við þurfum þó ekki að hætta þar.
- ▶ Við getum notað M^p til að fá a_{k+p} .

- ▶ Við þurfum þó ekki að hætta þar.
- ▶ Við getum notað M^p til að fá a_{k+p} .
- ▶ Þetta getum við reinkað í $\mathcal{O}(\quad)$ tíma.

- ▶ Við þurufm þó ekki að hætta þar.
- ▶ Við getum notað M^p til að fá a_{k+p} .
- ▶ Þetta getum við reinkað í $\mathcal{O}(k^3 \log p)$ tíma.

- ▶ Við þurfum þó ekki að hætta þar.
- ▶ Við getum notað M^p til að fá a_{k+p} .
- ▶ Þetta getum við reinkað í $\mathcal{O}(k^3 \log p)$ tíma.
- ▶ Við getum notað þetta til að reikna n -tu Fibonacci töluna í $\mathcal{O}(\quad)$ tíma.

- ▶ Við þurufm þó ekki að hætta þar.
- ▶ Við getum notað M^p til að fá a_{k+p} .
- ▶ Þetta getum við reinkað í $\mathcal{O}(k^3 \log p)$ tíma.
- ▶ Við getum notað þetta til að reikna n -tu Fibonacci töluna í $\mathcal{O}(\log n)$ tíma.

- ▶ Við þurfum þó ekki að hætta þar.
- ▶ Við getum notað M^p til að fá a_{k+p} .
- ▶ Þetta getum við reinkað í $\mathcal{O}(k^3 \log p)$ tíma.
- ▶ Við getum notað þetta til að reikna n -tu Fibonacci töluna í $\mathcal{O}(\log n)$ tíma.
- ▶ Það er meiri en nógu hratt fyrir $n < 10^{18}$.

```

14 void matpow(ll* a, ll p, ll n)
15 { // Eins og a = a^p.
16   ll r[n][n], i, j;
17   rep(i, n) rep(j, n) r[i][j] = 0;
18   rep(i, n) r[i][i] = 1;
19   while (p > 0)
20   {
21     if (p%2 == 1) multo(*r, a, n);
22     p /= 2;
23     multo(a, a, n);
24   }
25   rep(i, n) rep(j, n) a[i*n + j] = r[i][j];
26 }
27
28 int main()
29 { // reiknar |n|-ta Fibonacci töluna.
30   ll n, a[2][2] = {{1, 1}, {1, 0}};
31   scanf("%lld", &n);
32   if (n == 1 || n == 2) printf("1\n");
33   else
34   {
35     matpow(a, n - 2, 2);
36     printf("%lld\n", (a[0][0] + a[0][1])%MOD);
37   }
38   return 0;
39 }

```

- ▶ Í dæmum sem beita má þessari aðferð er oft erfitt að finna rakningarvenslin (þegar þau eru ekki gefin beint).

- ▶ Í dæmum sem beita má þessari aðferð er oft erfitt að finna rakningarvenslin (þegar þau eru ekki gefin beint).
- ▶ Þá má oft nota netafræðina í staðin.

- ▶ Í dæmum sem beita má þessari aðferð er oft erfitt að finna rakningarvenslin (þegar þau eru ekki gefin beint).
- ▶ Þá má oft nota netafræðina í staðin.
- ▶ Tökum dæmi.

- ▶ Þú vilt leggja flísarnar á ganginn þinn þannig að hver flís er annað hvort svört eða hvít, og engar tvær aðliggjandi flísar mega vera hvítar.

- ▶ Þú vilt leggja flísarnar á ganginn þinn þannig að hver flís er annað hvort svört eða hvít, og engar tvær aðliggjandi flísar mega vera hvítar.
- ▶ Gerum ráð fyrir að gangurinn sé þrjár flísar á breidd og n flísar á lengd.

- ▶ Þú vilt leggja flísarnar á ganginn þinn þannig að hver flís er annað hvort svört eða hvít, og engar tvær aðliggjandi flísar mega vera hvítar.
- ▶ Gerum ráð fyrir að gangurinn sé þrjár flísar á breidd og n flísar á lengd.
- ▶ Á hversu marga vegu getur þú lagt flísarnar?

- ▶ Þú vilt leggja flísarnar á ganginn þinn þannig að hver flís er annað hvort svört eða hvít, og engar tvær aðliggjandi flísar mega vera hvítar.
- ▶ Gerum ráð fyrir að gangurinn sé þrjár flísar á breidd og n flísar á lengd.
- ▶ Á hversu marga vegu getur þú lagt flísarnar?
- ▶ Flísar er snertast horn í horn teljast ekki aðliggjandi.

- ▶ Hægt er að setja þessa talningu fram með fjórða stigs línuegum rakningarvenslum.

- ▶ Hægt er að setja þessa talningu fram með fjórða stigs línuegum rakningarvenslum.
- ▶ Það er þau ekki auðséð hverjir stuðlarnir í þeim venslum erum.

- ▶ Hægt er að setja þessa talningu fram með fjórða stigs línuegum rakningarvenslum.
- ▶ Það er þau ekki auðséð hverjir stuðlarnir í þeim venslum erum.
- ▶ En hvað ef við breytum þessu í net.

- ▶ Hægt er að setja þessa talningu fram með fjórða stigs línuegum rakningarvenslum.
- ▶ Það er þau ekki auðséð hverjir stuðlarnir í þeim venslum erum.
- ▶ En hvað ef við breytum þessu í net.
- ▶ Látum hnútana í netinu vera mögulegir dálkar á ganginum:

- ▶ Hægt er að setja þessa talningu fram með fjórða stigs línuegum rakningarvenslum.
- ▶ Það er þau ekki auðséð hverjir stuðlarnir í þeim venslum erum.
- ▶ En hvað ef við breytum þessu í net.
- ▶ Látum hnútana í netinu vera mögulegir dálkar á ganginum:

Hnutur:	0	1	2	3	4	5	6	7
	0	0	0	0	1	1	1	1
Gangur:	0	0	1	1	0	0	1	1
	0	1	0	1	0	1	0	1

- ▶ Hægt er að setja þessa talningu fram með fjórða stigs línuegum rakningarvenslum.
- ▶ Það er þau ekki auðséð hverjir stuðlarnir í þeim venslum erum.
- ▶ En hvað ef við breitung þessu í net.
- ▶ Látum hnútana í netinu vera mögulegir dálkar á ganginum:

Hnutur:	0	1	2	3	4	5	6	7
	0	0	0	0	1	1	1	1
Gangur:	0	0	1	1	0	0	1	1
	0	1	0	1	0	1	0	1

- ▶ Við bætum svo við legg á milli hnútanna u og v ef dálkarnir mega koma fyrir á ganginum.

- ▶ Hægt er að setja þessa talningu fram með fjórða stigs línuegum rakningarvenslum.
- ▶ Það er þau ekki auðséð hverjir stuðlarnir í þeim venslum erum.
- ▶ En hvað ef við breytum þessu í net.
- ▶ Látum hnútana í netinu vera mögulegir dálkar á ganginum:

Hnutur:	0	1	2	3	4	5	6	7
	0	0	0	0	1	1	1	1
Gangur:	0	0	1	1	0	0	1	1
	0	1	0	1	0	1	0	1

- ▶ Við bætum svo við legg á milli hnútanna u og v ef dálkarnir mega koma fyrir á ganginum.
- ▶ Látum 1 tákna hvítar flísar.

- ▶ Hægt er að setja þessa talningu fram með fjórða stigs línuegum rakningarvenslum.
- ▶ Það er þau ekki auðséð hverjir stuðlarnir í þeim venslum erum.
- ▶ En hvað ef við breitung þessu í net.
- ▶ Látum hnútana í netinu vera mögulegir dálkar á ganginum:

Hnutur:	0	1	2	3	4	5	6	7
	0	0	0	0	1	1	1	1
Gangur:	0	0	1	1	0	0	1	1
	0	1	0	1	0	1	0	1

- ▶ Við bætum svo við legg á milli hnútanna u og v ef dálkarnir mega koma fyrir á ganginum.
- ▶ Látum 1 tákna hvítar flísar.
- ▶ Takið eftir að stöður 3, 6 og 7 eru alfarið ólöglegar.

- ▶ Við fáum þá færslufylkið:

► Við fáum þá færslufylkið:

	0	1	2	3	4	5	6	7	
	+-----								
0		1	1	1	0	1	1	0	0
1		1	0	1	0	1	0	0	0
2		1	1	0	0	1	1	0	0
3		0	0	0	0	0	0	0	0
4		1	1	1	0	0	0	0	0
5		1	0	1	0	0	0	0	0
6		0	0	0	0	0	0	0	0
7		0	0	0	0	0	0	0	0

- Við fáum þá færslufylkið:

	0	1	2	3	4	5	6	7	
	+-----								
0		1	1	1	0	1	1	0	0
1		1	0	1	0	1	0	0	0
2		1	1	0	0	1	1	0	0
3		0	0	0	0	0	0	0	0
4		1	1	1	0	0	0	0	0
5		1	0	1	0	0	0	0	0
6		0	0	0	0	0	0	0	0
7		0	0	0	0	0	0	0	0

- Köllum þetta fylki A.

- ▶ Við fáum þá færslufylkið:

	0	1	2	3	4	5	6	7
	+-----							
0	1	1	1	0	1	1	0	0
1	1	0	1	0	1	0	0	0
2	1	1	0	0	1	1	0	0
3	0	0	0	0	0	0	0	0
4	1	1	1	0	0	0	0	0
5	1	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

- ▶ Köllum þetta fylki A .
- ▶ Við þurfum síðan að leggja saman $(A^{p-1})_{uv}$ fyrir öll u og v þar sem u er lögleg.

- ▶ Við fáum þá færslufylkið:

	0	1	2	3	4	5	6	7
	+-----							
0	1	1	1	0	1	1	0	0
1	1	0	1	0	1	0	0	0
2	1	1	0	0	1	1	0	0
3	0	0	0	0	0	0	0	0
4	1	1	1	0	0	0	0	0
5	1	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

- ▶ Köllum þetta fylki A .
- ▶ Við þurfum síðan að leggja saman $(A^{p-1})_{uv}$ fyrir öll u og v þar sem u er lögleg.
- ▶ Þá eru við að telja saman alla vegi í netinu af lengd p sem byrja í löglegri stöðu.

- ▶ Við fáum þá færslufylkið:

	0	1	2	3	4	5	6	7
	+-----							
0	1	1	1	0	1	1	0	0
1	1	0	1	0	1	0	0	0
2	1	1	0	0	1	1	0	0
3	0	0	0	0	0	0	0	0
4	1	1	1	0	0	0	0	0
5	1	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

- ▶ Köllum þetta fylki A .
- ▶ Við þurfum síðan að leggja saman $(A^{p-1})_{uv}$ fyrir öll u og v þar sem u er lögleg.
- ▶ Þá eru við að telja saman alla vegi í netinu af lengd p sem byrja í löglegri stöðu.
- ▶ Við þurfum ekki að passa að síðasti hnúturinn sé löglegur því það liggur enginn leggur í ólöglegann hnút.

```

28 int main()
29 { // reiknar |n|-ta Fibonacci toluna.
30   ll i, j, n, r = 0, a[8][8] =
31   {
32       {1, 1, 1, 0, 1, 1, 0, 0},
33       {1, 0, 1, 0, 1, 0, 0, 0},
34       {1, 1, 0, 0, 1, 1, 0, 0},
35       {0, 0, 0, 0, 0, 0, 0, 0},
36       {1, 1, 1, 0, 0, 0, 0, 0},
37       {1, 0, 1, 0, 0, 0, 0, 0},
38       {0, 0, 0, 0, 0, 0, 0, 0},
39       {0, 0, 0, 0, 0, 0, 0, 0}
40   };
41   scanf("%lld", &n);
42   matpow(*a, n - 1, 8);
43   rep(i, 8) rep(j, 8) if (i != 3 && i != 6 && i != 7) r = (r + a[i][j])%MOD;
44   printf("%lld\n", r);
45   return 0;
46 }

```

- ▶ Umræða um fylkjaaðgerðir er ekki fullkláruð fyrr en minnst er á Gauss eyðingu.

- ▶ Umræða um fylkjaaðgerðir er ekki fullkláruð fyrr en minnst er á Gauss eyðingu.
- ▶ Þið munið eflaust eftir henni úr Línulegri Algebru.

- ▶ Umræða um fylkjaaðgerðir er ekki fullkláruð fyrr en minnst er á Gauss eyðingu.
- ▶ Þið munið eflaust eftir henni úr Línulegri Algebru.
- ▶ Hún er nytsamlega til að, til dæmis, leysa jöfnuhneppi eða finna andhverfur fylkja.

```

6 int gauss(double* a, int s, int n, int m)
7 { // Her er |a| |n|x|m| fylki. Gauss eyding er framkvaemd upp ad dalki |s|.
8   int i, j, k, t, r = 0;
9   rep(i, n)
10    {
11      t = -1;
12      while (++t < s && fabs(a[i*m + t]) < 1e-9);
13      if (t == s) continue;
14      r++;
15      for (j = m - 1; j >= t; j--) a[i*m + j] = a[i*m + j]/a[i*m + t];
16      rep(j, n) if (i != j) for (k = m - 1; k >= t; k--)
17        a[j*m + k] = a[j*m + k] - a[i*m + k]*a[j*m + t];
18    }
19   return r;
20 }

```

