Use of patterns in n-tuple combinatorial generation

Dominik Zalewski

Faculty of Mathematics and Computer Science
UAM
Otwarte Wykłady dla Doktorantów

25-26 May 2007

Agenda

1. Applications
   - Peer-to-peer networks
   - Calendaring
2. Combinatorial generation
   - Model
   - Data structures
   - Algorithms
3. Running time
   - Theoretical analysis
   - Experimental results

Bibliografia

- D. E. Knuth, *The art of computer programming: Volume 4*, Addison Wesley, 2005.
- D. L. Kreher and D. R. Stinson, *Combinatorial algorithms: Generation, enumeration and search*, CRC press LTC, Boca Raton, Florida, 1998.
- Dmitri Loguinov, Juan Casas, and Xiaoming Wang, *Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience*, IEEE/ACM Trans. Netw. **13** (2005), no. 5, 1107–1120.
- C. D. Savage, *A survey of combinatorial gray codes*, 605–629.
- Paul Vixie, *Cron manual*, 1993.

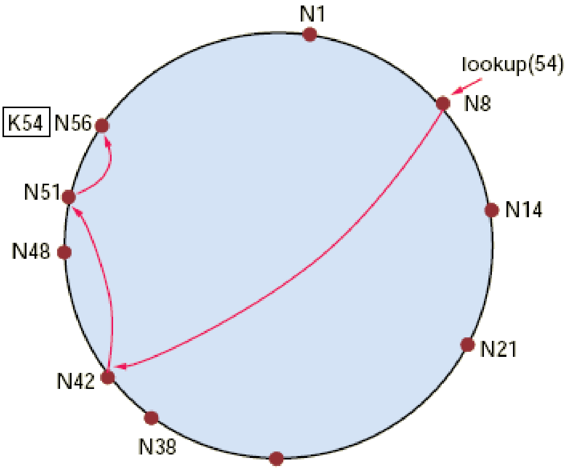
Agenda

1. Applications
   - Peer-to-peer networks
   - Calendaring
2. Combinatorial generation
   - Model
   - Data structures
   - Algorithms
3. Running time
   - Theoretical analysis
   - Experimental results

Chord – definition

- Choose **key space** $\mathcal{S}$, such that$|\mathcal{S}| = 2^m$.
- Assign a **random key** $k_i \in \mathcal{S}$ to every node in the network
- **Order** nodes such that they form **cyclic list** and $k_i < k_{i+1}$
- Every node v keeps its **routing table** R_v of size m, such that $R_i[j]$ stores address of node in distance 2^j from v

Chord – searching



Distributed Hashtables

Other structured peer-to-peer networks based on DHT use **minimal-change** topologies like

- *n*-cube
- Butterfly
- de Brujin graph

What if someone wanted to visit some subset of nodes?

Distributed Hashtables

Other structured peer-to-peer networks based on DHT use **minimal-change** topologies like

- *n*-cube
- Butterfly
- de Brujin graph

What if someone wanted to visit some subset of nodes?

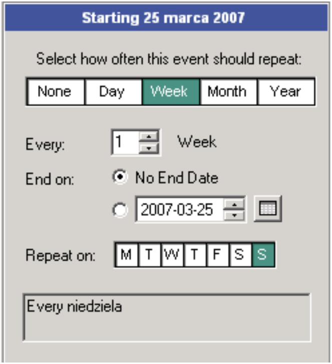
Agenda

Software

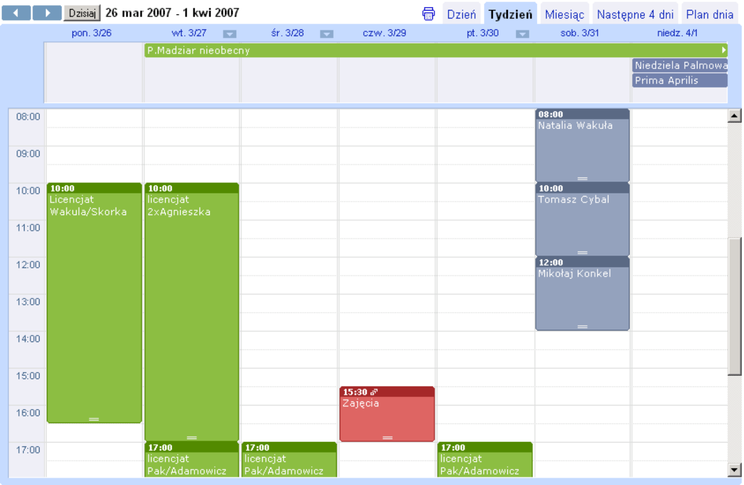
- Personal Information Mangement (*PIM*): Palm Desktop, Mozilla
- Google Calendar
- CRON

PIM pattern specification

- choose **exclusively** one of the **patterns** (granularity)
- choose **start** and **end** date



PIM views


Dzisiaj
26 mar 2007 - 1 kwi 2007
Dzień
Tydzień
Miesiąc
Następne 4 dni
Plan dnia
pon. 3/26
wt. 3/27
śr. 3/28
czw. 3/29
pt. 3/30
sob. 3/31
niedz. 4/1
P.Madziar nieobecny
Niedziela Palmowa
Prima Aprilis
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
10:00 Licencjat Wakula/Skorka
10:00 licencjat 2xAgnieszka
08:00 Natalia Wakuła
10:00 Tomasz Cybal
12:00 Mikołaj Konkel
15:30 Zajęcia
17:00 licencjat Pak/Adamowicz
17:00 licencjat Pak/Adamowicz
17:00 licencjat Pak/Adamowicz

CRON pattern specification

- 6 fields: year, month, dom, dow, hour, minute
- example: $\star$, $\star$, 1–15, $\star$, 0–23/2, 0
- runs in background checking **every minute** if there is an event matching a pattern

How to generate monthly view for CRON?

CRON pattern specification

- 6 fields: year, month, dom, dow, hour, minute
- example: $\star$, $\star$, 1–15, $\star$, 0–23/2, 0
- runs in background checking **every minute** if there is an event matching a pattern

How to generate monthly view for CRON?

Agenda

Sequence spaces

Definition (Consecutive sequence space)

Space Ω_n is called **consecutive sequence space** iff for some $n \in \mathbb{N}$, $s_j \in \mathbb{N}$, $e_j \in \mathbb{N}$ we have

$$\Omega_n = \{(a_n) : s_j \leq a_j \leq e_j, 1 \leq j \leq n, a_j \in \mathbb{N}\}.$$

Definition (Pattern sequence space)

Space Ψ_n is called **pattern sequence space** iff for some $n \in \mathbb{N}$ and characteristic functions $\chi = (\chi_n)$, where $\chi_j : \mathbb{N} \mapsto \{0, 1\}$ for all $1 \leq j \leq n$, we have

$$\Psi_n = \{(a_n) : \chi_j(a_j) = 1, 1 \leq j \leq n, a_j \in \mathbb{N}\}.$$

Sequence spaces

Definition (Consecutive sequence space)

Space Ω_n is called **consecutive sequence space** iff for some $n \in \mathbb{N}$, $s_j \in \mathbb{N}$, $e_j \in \mathbb{N}$ we have

$$\Omega_n = \{(a_n) : s_j \leq a_j \leq e_j, 1 \leq j \leq n, a_j \in \mathbb{N}\}.$$

Definition (Pattern sequence space)

Space Ψ_n is called **pattern sequence space** iff for some $n \in \mathbb{N}$ and characteristic functions $\chi = (\chi_n)$, where $\chi_j : \mathbb{N} \mapsto \{0, 1\}$ for all $1 \leq j \leq n$, we have

$$\Psi_n = \{(a_n) : \chi_j(a_j) = 1, 1 \leq j \leq n, a_j \in \mathbb{N}\}.$$

Sequence ranges

Definition (Sequence range)

A pair $((f_n),(t_n))$ is called **sequence range** iff $(f_n) \in \Omega_n$, $(t_n) \in \Omega_n$ and $(f_n) \preceq_{\Omega_n} (t_n)$.

Definition (Sequence space subrange)

Sequence range $((f_n),(t_n))$ generates **sequence space subrange** $\Phi_n^{f,t}$ for some sequence space Φ_n where

$$\Phi_n^{f,t} = \{(a_n) \in \Phi_n : (f_n) \preceq (a_n) \preceq (t_n)\}$$

and by Φ_n we may take pattern sequence space Ψ_n or consecutive sequence space Ω_n.

Sequence ranges

Definition (Sequence range)

A pair $((f_n),(t_n))$ is called **sequence range** iff $(f_n) \in \Omega_n$, $(t_n) \in \Omega_n$ and $(f_n) \preceq_{\Omega_n} (t_n)$.

Definition (Sequence space subrange)

Sequence range $((f_n),(t_n))$ generates **sequence space subrange** $\Phi_n^{f,t}$ for some sequence space Φ_n where

$$\Phi_n^{f,t} = \{(a_n) \in \Phi_n : (f_n) \preceq (a_n) \preceq (t_n)\}$$

and by Φ_n we may take pattern sequence space Ψ_n or consecutive sequence space Ω_n.

Not so easy task

Definition (Pattern sequence rookie)

Sequence $(r_n) \in \Psi_n^{f,t}$ is called **pattern sequence rookie** iff

$$\forall (a_n) \in \Psi_n^{f,t} : (r_n) \preceq (a_n).$$

Solution: find pattern sequence rookie!

Not so easy task

Definition (Pattern sequence rookie)

Sequence $(r_n) \in \Psi_n^{f,t}$ is called **pattern sequence rookie** iff

$$\forall (a_n) \in \Psi_n^{f,t} : (r_n) \preceq (a_n).$$

Solution: find pattern sequence rookie!

Agenda

1. Applications
   - Peer-to-peer networks
   - Calendaring
2. Combinatorial generation
   - Model
   - **Data structures**
   - Algorithms
3. Running time
   - Theoretical analysis
   - Experimental results

Pattern

Definition (Pattern)

Let $\Gamma = (\Gamma_n)$ be a **pattern** defined as $(\Gamma_j) = supp\{\chi_j\}$ for all $1 \leq j \leq n$.

Definition (Filtered pattern)

For sequence range $((f_n), (t_n))$ we define $\Gamma^f = (\Gamma^f_n)$ to be a **filtered pattern**, where $\Gamma^f_j = \Gamma_j \cap \{f_j, \ldots, \infty\}$ for all $1 \leq j \leq n$.

Pattern

Definition (Pattern)

Let $\Gamma = (\Gamma_n)$ be a **pattern** defined as $(\Gamma_j) = supp\{\chi_j\}$ for all $1 \leq j \leq n$.

Definition (Filtered pattern)

For sequence range $((f_n), (t_n))$ we define $\Gamma^f = (\Gamma^f_n)$ to be a **filtered pattern**, where $\Gamma^f_j = \Gamma_j \cap \{f_j, \ldots, \infty\}$ for all $1 \leq j \leq n$.

Agenda

1. Applications
   - Peer-to-peer networks
   - Calendaring
2. Combinatorial generation
   - Model
   - Data structures
   - Algorithms
3. Running time
   - Theoretical analysis
   - Experimental results

Naive generate

Algorithm 2.1: $\textsc{NaiveGenerate}((f_n),(t_n),(\chi_n))$

global ε

$(c_n) \leftarrow (f_n)$ (1)

while $(c_n) \preceq (t_n)$ (2)

$$\textbf{do} \begin{cases} \textbf{if } \chi((c_n)) \equiv 1 & (3)\\ \quad \textbf{then output } ((c_n)) & (4)\\ (c_n) \leftarrow (c_n) \oplus \varepsilon & (5) \end{cases}$$

JSUCCESSOR()

Algorithm 2.2: JSUCCESSOR($j, (c_n), (\Gamma_n)$)

global (λ_n)
$i \leftarrow j$
while $i > 0$ **and** $\lambda_i = |\Gamma_i|$ (1)
  do $i \leftarrow i - 1$
if $i = 0$
  then return ("*undefined*")
$(s_n) \leftarrow (c_n)$ (2)
$\lambda_i \leftarrow \lambda_i + 1$
$s_i \leftarrow \Gamma_{i,\lambda_i}$
for $k \leftarrow i + 1$ **to** j (3)
  do $\begin{cases} \lambda_k = 1 \\ s_k \leftarrow \Gamma_{k,1} \end{cases}$
return $((s_n))$ (4)

More definitions

Definition (Favorite subsequence)

Subsequence $(u_j) = (u_1, u_2, \ldots, u_j)$ for $1 \leq j \leq n$ is called **favorite subsequence** of sequence space subrange $\Psi_n^{f,t}$ iff (u_j) is a pattern sequence rookie of sequence space subrange $\Psi_j^{f,t}$.

Favorite subsequence lemma

Lemma

- *If* $(u_{j-1}) \in \Psi^{f,t}_{j-1}$ *is a* ***favorite subsequence*** *of* $\Psi^{f,t}_{n}$ *for* $1 < j \leq n$ *and pattern sequence rookie* $(r_n) \in \Psi^{f,t}_{n}$ *exists then:*
  - (T1) $(u_1, u_2, \ldots, u_{j-1}, \Gamma^{f}_{j,1}, \Gamma_{j+1,1}, \Gamma_{j+2,1}, \ldots, \Gamma_{n,1})$ *is a pattern sequence rookie of* $\Psi^{f,t}_{n}$ *when* $f_j < \Gamma^{f}_{j,1}$,
  - (T2) $(v_1, v_2, \ldots, v_{j-1}, \Gamma_{j,1}, \Gamma_{j+1,1}, \ldots, \Gamma_{n,1})$ *is a pattern sequence rookie of* $\Psi^{f,t}_{n}$ *when* $\Gamma^{f}_{j,1}$ *does not exist, where* $(v_{j-1}) \in \Psi^{f,t}_{j-1}$ *is a* ***successor*** *of* $(u_{j-1}) \in \Psi^{f,t}_{j-1}$,
  - (R) $(u_1, u_2, \ldots, u_{j-1}, \Gamma^{f}_{j,1}) \in \Psi^{f,t}_{j}$ *is a favorite subsequence of* $\Psi^{f,t}_{n}$ *when* $f_j = \Gamma^{f}_{j,1}$.
- *If* $j = 1$ *and pattern sequence rookie* $(r_n) \in \Psi^{f,t}_{n}$ *exists, then* $(\Gamma^{f}_{1,1})$ *is a favorite subsequence of* $\Psi^{f,t}_{n}$.

Auxiliary array (λ^f)

Definition

$$\lambda_j^f = \begin{cases} 0, & (\Gamma_j^f) = \emptyset \\ z : \Gamma_{j,z} = \Gamma_{j,1}^f, & (\Gamma_j^f) \neq \emptyset \end{cases}$$

GETFIRST()

Algorithm 2.3: GETFIRST$((f_n),(t_n),(\Gamma_n))$

for $j \leftarrow 1$ **to** n

$$\textbf{do}\ \begin{cases} \textbf{if}\ \lambda_j^f = 0 \\ \quad \textbf{then}\ \begin{cases} (r_n) \leftarrow \text{JSUCCESSOR}(j-1,(r_n),(\Gamma_n)) \\ \textbf{break} \end{cases} \\ r_j \leftarrow \Gamma_{j,\lambda_j^f} \\ \lambda_j \leftarrow \lambda_j^f \\ \textbf{if}\ f_j < \Gamma_{j,\lambda_j^f} \\ \quad \textbf{then}\ \begin{cases} j \leftarrow j+1 \\ \textbf{break} \end{cases} \end{cases}$$

for $i \leftarrow j$ **to** n

  do $r_i \leftarrow \Gamma_{i,1}$

return $((r_n))$

GETFIRST() validity

Theorem

Algorithm GETFIRST*() returns* ***pattern sequence rookie*** (r_n) *of sequence space subrange* $\Psi_n^{f,t}$*, if* (r_n) *exists.*

Pattern generate

Algorithm 2.4: $\text{PATTERNGENERATE}((f_n), (t_n), (\Gamma_n))$

$(c_n) \leftarrow \text{GETFIRST}((f_n), (t_n), (\Gamma_n))$
while $(c_n) \neq$ "*undefined*"
do $\begin{cases} \textbf{output } ((c_n)) \\ (c_n) \leftarrow \text{JSUCCESSOR}(n, (c_n), (\Gamma_n)) \end{cases}$

Agenda

Granularity

Definition (Pattern granularity)

Let us take sequence range $((f_n), (t_n))$ and pattern Γ. We define **pattern granularity** as

$$\mathrm{gran}(\Gamma) = \frac{|\Psi_n^{f,t}|}{|\Omega_n^{f,t}|}$$

Naive generate running time

Algorithm 3.1: NAIVEGENERATE$((f_n),(t_n),(\chi_n))$

global ε
$(c_n) \leftarrow (f_n)$ (1)
while $(c_n) \preceq (t_n)$ (2)
do
- **if** $\chi((c_n)) \equiv 1$ (3)
- **then output** $((c_n))$ (4)
- $(c_n) \leftarrow (c_n) \oplus \varepsilon$ (5)

$$\frac{1}{\varepsilon} O(n|\Omega_n^{f,t}|) = O(n\frac{1}{\varepsilon}|\Omega_n^{f,t}|)$$

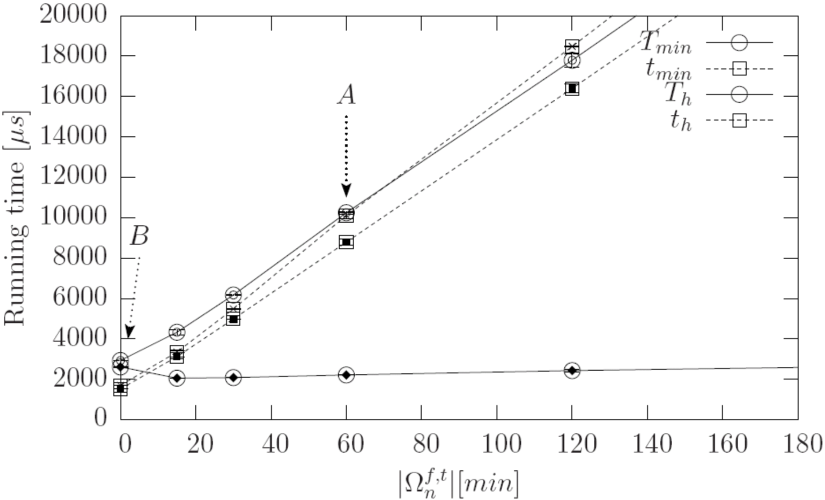
Pattern generate running time

Algorithm 3.2: PATTERNGENERATE$((f_n),(t_n),(\Gamma_n))$

$(c_n) \leftarrow$ GETFIRST$((f_n),(t_n),(\Gamma_n))$
while $(c_n) \neq$ "*undefined*"
do $\begin{cases} \textbf{output } ((c_n)) \\ (c_n) \leftarrow \text{JSUCCESSOR}(n,(c_n),(\Gamma_n)) \end{cases}$

$$O(n \log n + n\,\mathrm{gran}(\Gamma_n)|\Omega_n^{f,t}|) = O(n\,\mathrm{gran}(\Gamma_n)|\Omega_n^{f,t}|).$$

Agenda


Running time $[\mu s]$
$|\Omega_n^{f,t}|[min]$
T_{min}
t_{min}
T_h
t_h
A
B
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
0
20
40
60
80
100
120
140
160
180

$\lvert\Omega_n^{f,t}\rvert$	T_h	t_h	T_{min}	t_{min}
$0'$	$2.61ms$	$1.53ms$	$2.94ms$	$1.7ms$
$15'$	$2.06ms$	$3.12ms$	$4.33ms$	$3.35ms$
$30'$	$2.09ms$	$4.99ms$	$6.18ms$	$5.49ms$
$1h$	$2.22ms$	$8.79ms$	$10.25ms$	$10.11ms$
$2h$	$2.44ms$	$16.4ms$	$17.79ms$	$18.48ms$
$4h$	$2.75ms$	$31.79ms$	$33.3ms$	$35.6ms$
$8h$	$3.96ms$	$62.54ms$	$63.68ms$	$70.5ms$
$16h$	$4.9ms$	$125.82ms$	$125.36ms$	$140.74ms$
$1m$	$129.28ms$	$6.01''$	$5.71''$	$6.82''$
$2m$	$243.82ms$	$11.85''$	$10.83''$	$12.89''$
$4m$	$502.85ms$	$23.55''$	$22.14''$	$26.41''$
$8m$	$1''$	$48.11''$	$44.97''$	$54.19''$
$1y$	$1.5''$	$1.21'$	$1.12'$	$1.37'$
$2y$	$3''$	$2.44'$	$2.26'$	$2.76'$
$4y$	$6.03''$	$4.88'$	$4.51'$	$5.5'$

Summary

- Define RANK() and UNRANK() for **lexicographic** ordering
- Define **all** algorithms for **minimal-change** ordering
- **Apply** to **peer-to-peer** structured networks

Summary

- Define RANK() and UNRANK() for **lexicographic** ordering
- Define **all** algorithms for **minimal-change** ordering
- **Apply** to **peer-to-peer** structured networks

Summary

- Define RANK() and UNRANK() for **lexicographic** ordering
- Define **all** algorithms for **minimal-change** ordering
- **Apply** to **peer-to-peer** structured networks