Lecture 5 Combinatorics

Algorithm Analysis

- Permutations
  - Recursive
  - Johnson-Trotter algorithm
  - Lexicographical order
- Subsets
  - Recursive
  - Squashed ordering
  - Lexicographical order
- Gray code
- Snail sort

permutations

Recursive

1

21 12

321 231 213 312 132 123

recursive algorithm 1:
from each permutation of length n-1, generate n permutations of length n by inserting n at positions 0,…,n-1

recursive algorithm 2:
from each permutation of length n-1, generate n permutations of length n by inserting n at positions n-1, …, 0

1

12 21

123 132 312 213 231 321

Johnson-Trotter alg. - mobile

Step 1 ini: $\overleftarrow{1}\ \overleftarrow{2}\ \overleftarrow{3}\ \ldots\ \overleftarrow{n}$

Mobile: find an a

Either $(\vec{a}b \text{ and } \mathrm{b<a})$

Or $(b\overleftarrow{a} \text{ and } \mathrm{b<a})$

- Mobile example
- $\overleftarrow{3}\ \overrightarrow{2}\ \overrightarrow{1}\ \overleftarrow{4}$
- Mobile: 2 and 4

Johnson-Trotter alg.

while the permutation has a "mobile" element

find largest mobile element k

swap k and the element that k is "pointing to"

reverse the direction of all elements that are larger than k

$$\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3} \rightarrow \overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3} \rightarrow \overleftarrow{1}\,\overleftarrow{3}\,\overleftarrow{2} \rightarrow \overleftarrow{1}\,\overleftarrow{3}\,\overleftarrow{2} \rightarrow \overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2} \rightarrow \overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2} \rightarrow \overrightarrow{3}\,\overleftarrow{2}\,\overleftarrow{1}$$

$$\downarrow$$

$$\overleftarrow{2}\,\overleftarrow{1}\,\overrightarrow{3} \leftarrow \overleftarrow{2}\,\overleftarrow{1}\,\overrightarrow{3} \leftarrow \overleftarrow{2}\,\overrightarrow{3}\,\overleftarrow{1} \leftarrow \overleftarrow{2}\,\overrightarrow{3}\,\overleftarrow{1} \leftarrow \overrightarrow{3}\,\overleftarrow{2}\,\overleftarrow{1}$$

Either $(\vec{a}b \text{ and } b<a)$

Or $(b\overleftarrow{a} \text{ and } b<a)$

doesn't require to store all permutations of size n-1 and doesn't require going through all shorter permutations

lexicographic order

- Find the largest index k such that $a[k] < a[k+1]$. If no such index exists, the permutation is the last permutation.
- Find the largest index l greater than k such that $a[k] < a[l]$.
- Swap the value of $a[k]$ with that of $a[l]$.
- Reverse the sequence from $a[k+1]$ up to and including the final element $a[n]$.

lexicographic order - example

- **[1 2 3 4]**
  - $k = 2$, $l = 3$ (start at 0)
  - [1,2,4,3]
- **[1 2 4 3]**
  - $k = 1$, $l = 3$
  - [1,3,4,2]
- **[1 3 2 4]**
- …
- **[4 3 2 1]**
  - at which point $a[k] < a[k+1]$ does not exist, indicating that this is the last permutation

subsets

Recursive

- {1 2 3}
- {}
- {} ∪(1 + {}) → {} {1}
- {} {1} ∪ (2 + {} {1}) → {} {1} {2} {1 2}
- {} {1} {2} {1 2} ∪ (3 + {} {1} {2} {1 2}) → {} {1} {2} {1 2} {3} {1 3} {2 3} {1 2 3}

- 2^n

Squashed ordering

- 0 0000 {}
- 1 0001 {1}
- 2 0010 {2}
- 3 0011 {2,1}
- 4 0100 {3}
- 5 0101 {3,1}
- 6 0110 {3,2}
- 7 0111 {3,2,1}
- 8 1000 {4}
- 9 1001 {4,1}
- 10 1010 {4,2}
- 11 1011 {4,2,1}
- 12 1100 {4,3}
- 13 1101 {4,3,1}
- 14 1110 {4,3,2}
- 15 1111 {4,3,2,1}

lexicographic order

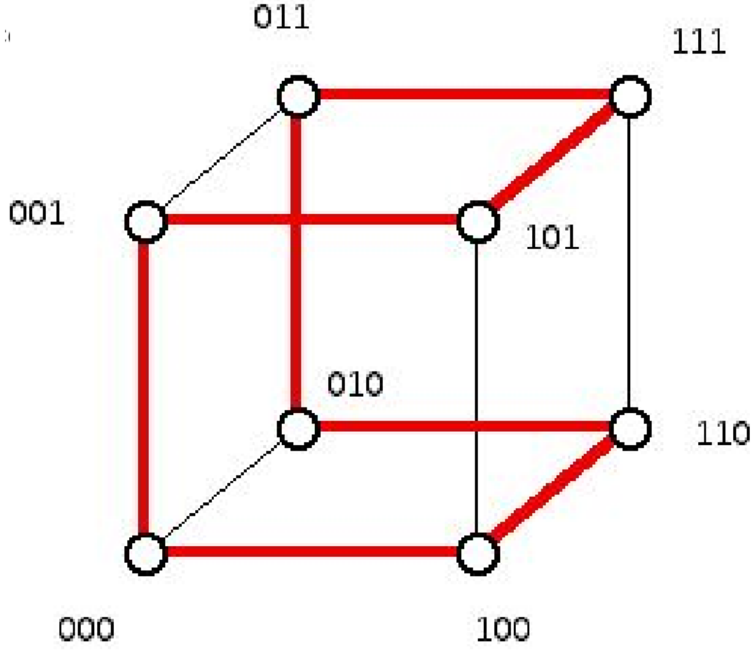
- 1
- 2
- 21
- 3
- 31
- 32
- 321
- 4
- 41
- 42
- 421
- 43
- 431
- 432
- 4321

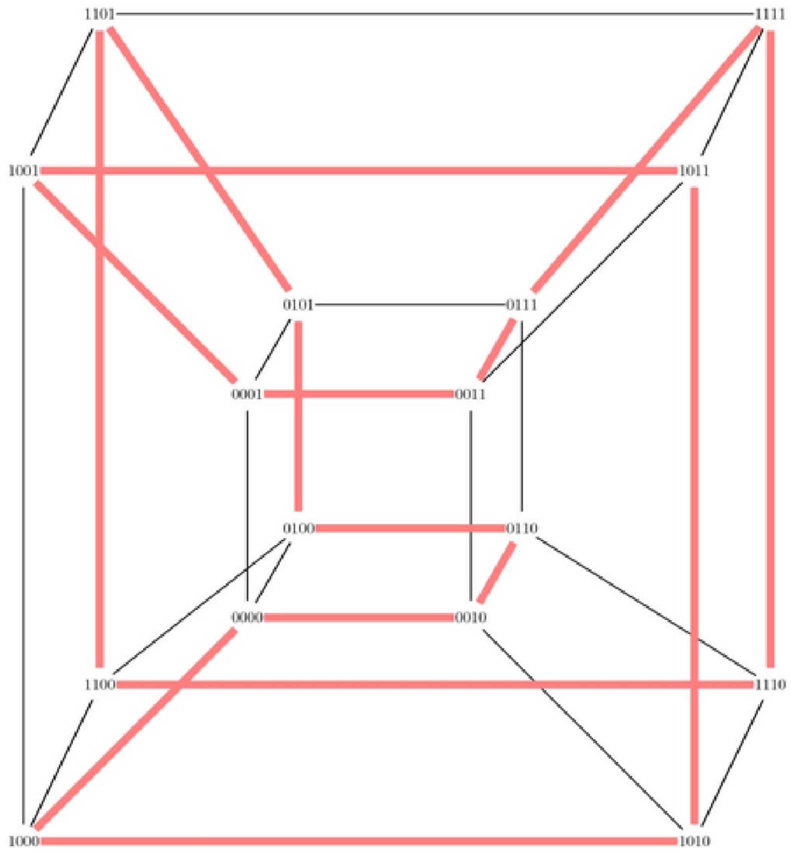
- 1
- 12
- 123
- 1234
- 124
- 13
- 134
- 14
- 2
- 23
- 234
- 24
- 3
- 34
- 4

Gray code

A list of the 2^n binary sequences of length n such that each sequence differs in exactly one place from the previous sequence is called Gray code of order n.

Gray code of order-3, 4





The reflected gray code

- The reflected gray code of order n
- n=1
  - 0 1
- n > 1
  - write up the reflected Gray code of order n−1,
  - add a 0 at the left of each sequence,
  - write up the reflected Gray code of order n−1 in its reversed order, from last to first, and
  - add a 1 at the left of each sequence.

- n = 1
  - 0 1
- n = 2
  - 0 1 → 00 01
  - 1 0 → 11 10
- n = 3
  - 00 01 11 10 → 000 001 011 010
  - 10 11 01 00 → 110 111 101 100

Algorithm

- Start with $a_n a_{n-1} a_{n-2} \cdots a_2 a_1 = 000\cdots00$
- 1. calculate $S = a_n + a_{n-1} + a_{n-2} \cdots a_2 + a_1$
- 2. if S is even, change a_1 from 1 to 0 or 0 to 1
- 3. if S is odd, find the smallest j such that $a_j = 1$
- 4. if j=n then terminate
- 5. if j<n then change a_{j+1}, and go to step1

Σ	j	a_4	a_3	a_2	a_1
		0	0	0	0
0		0	0	0	1
1	1	0	0	1	1
2		0	0	1	0
1	2	0	1	1	0
2		0	1	1	1
3	1	0	1	0	1
2		0	1	0	0
1	3	1	1	0	0
2		1	1	0	1
3	1	1	1	1	1
4		1	1	1	0
3	2	1	0	1	0
2		1	0	1	1
3	1	1	0	0	1
2		1	0	0	0
1	4	S	T	O	P

Snail sort

```
while not InOrder(list) {
    Next_Permutation(list) }
```