Subsequence:

order

e_1 e_2 e_3 e_4 e_5 . . . e_{100}

↑ ↑ ↑ ↑ subseq of 4

e_1 e_3 e_4 e_{100} ← in this order

Subarray e_1 e_2 [e_3 e_4 e_5 e_6] e_7 e_8

↑ ↑

Subset = subsequence without order

---

tuple → ordered

→ unordered

TODO

E → best

All permutations

1 2 3 4 ?

1 2 3
3 2 1
3 1 2
2 1 3
2 3 1
1 3 2

all of 3
n!

n+1 place

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

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

(n+1)! of n+1 person..

unique

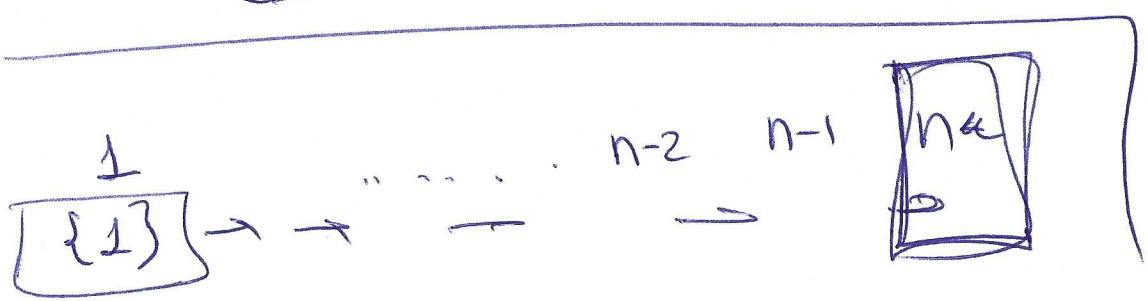
all of 4

Pigeon Hole Principle (MAT 258?)

Dirichlet

n+1

n



On demand

```
// object
PermGenerator pg(n); // logic

while( p = pg.next() )
{
   :
}
```

single

prev. alg/all at once

```
ContType C = GetPerm();
foreach el in C {
    check el ;
    ...
}
```

preserved Between calls.

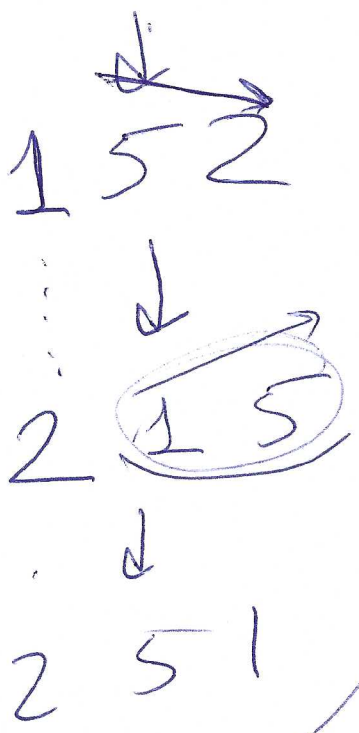
```
// function (just) if no info needs to be
   P = (1, 2, 3, ..., n))
while ( next_perm(p) )
do {
   :
} while ( next_perm(p) );
```

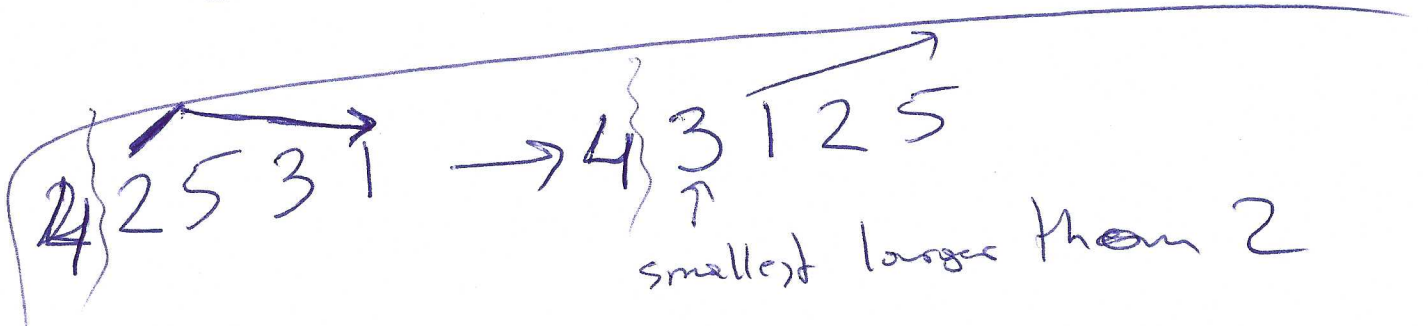
By ref

std::next_permutation

Game

1 2 5 — shuffle digits
~~bigger~~ bigger
smallest of the bigger





4 3 1 2 5 → 4 3 1 5 2

4 3 1 5 2 → 4 3 2 1 5

Lexicographical order of permutation

1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1 → done

files
abc
ab

Subsets

Set $\{a, b, c\}$

0 $\emptyset$

1 $\{a\}$ $\{b\}$ $\{c\}$

2 $\{a, b\}$ $\{a, c\}$ $\{b, c\}$

3 $\{a, b, c\}$

2^n subsets

a b c

in in in

out out or

$2 * 2 * 2 = 2^3$

---

2 pair of shoes

2 pair of socks,

$\{a, b, c, d\}$ — 5 —

/

subset of $\{a, b, c\}$

or

s/s of $\{a, b, c\}$ + $\{d\}$

$\{a, b\}$ → $\{a, b\}$

→ $\{a, b, d\}$

---

all at once idea

Subsets on demand

	A	B	C	
1	0	0	0	0
	1	0	0	4
	0	1	0	2
	0	0	1	1
3	0	1	1	6
	1	1	0	5
	1	0	1	3
	0	1	1	7
	1	1	1	

all possible values of 3-bit 0..7

Socks

RL RR

BL BR

none

} left foot

~~A=bi~~

n elements

n bit int.

```
for( i=0 ; i<<2^n ; ++i)
{ convert i to bits
  create subset that
  corresp. to 1's.
}
```

```
next subset( int i)
  return i+1 ;
```

```
for( i=0; i<10^6; ++i){
    C * p = new C(i); // malloc
    :
    do work
    :
    delete p; // throws away // free
}
```

slow

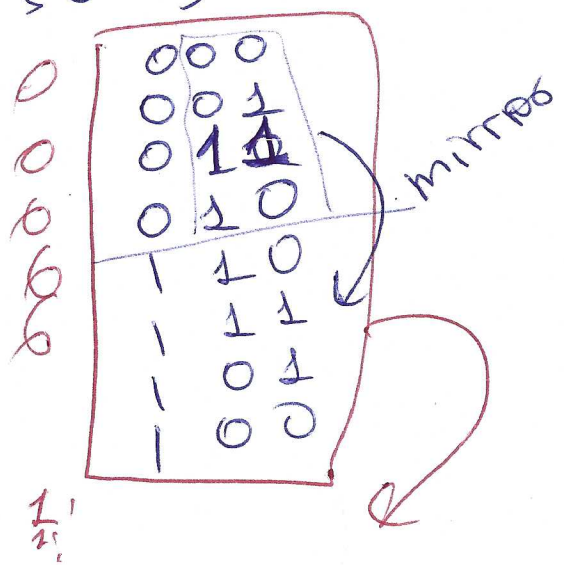
```
char * buffer = new char[sizeof(C)];
for (i=0; i<10^6; ++i){
    C * p = new (buffer) C(i); // ctor

    p->~C(); // dtor // salvage memory
}
delete [] buffer;
```

```
C c(); //default
for (i=0; i<10^6; ++i) {
    c.reinit(i); // c(0) → c(1) is very cheap
                 // only changes few bytes,
                 // n → n+1
                 // in subset  subs.
    :
    :
}
```

subset | for each subset

// each next subset
// is a small change
// from the previous.

order subsets so just 1 Bit changes, Gray code



MIN. CHANGE ORDER

```
for (i=0; i<(1<<n); ++i) {
    toBits( i ^ (i<<1) );
}
```