

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

TD DO  
 $E \rightarrow \{e_1, t\}$

# All permutations

1 2 3 4 ?

1 2 3  $\xrightarrow{n+1 \text{ place 1}}$

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

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

3 2 1 4

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

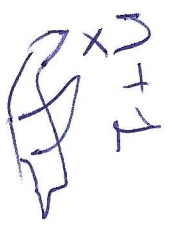
all of 3  
 $n!$

all of 4

$(n+1)!$   
 of  $n+1$  person  
 unique

Pigeon Hole Principle (MAT 258?)

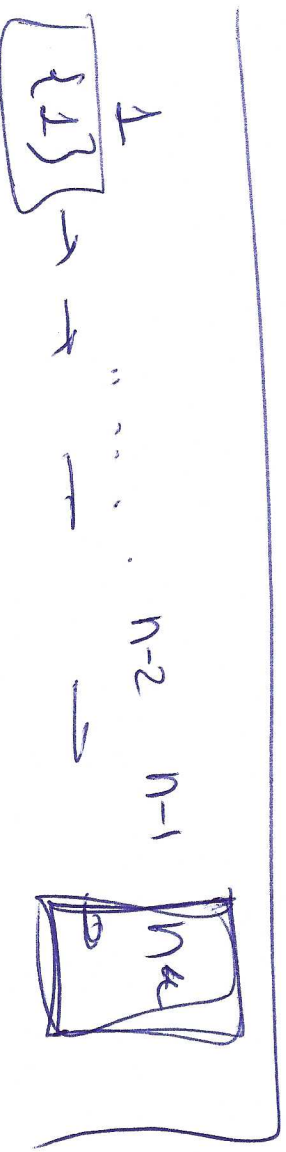
Dischlet



$n$   
 0 0 0 0 0

$\geq 2$  holes

2 pigeon



On demand

// object  
Perm Generator

pg(n) // ~~logic~~

while ( p = pg.next() )

single

{

:

}

// function (just) if no in to be

p = {1, 2, 3, ..., n}

~~while ( pg.next-perm(p) )~~

do {

:

} while ( next-perm(p) ) /

std::next-permutation

pres. algo / all at once - } -

ContType C = GetPerm();

→ for each el in C {

check el ;

}

preserved between calls.

Game

1 2 5

1 5 2

2 1 5

2 5 1

— shuffle digits

~~larger~~ bigger

Smallest of the bigger

4 2 5 3 1 → 4 3 1 2 5  
smallest larger than 2

4 3 1 2 5 → 4 3 1 5 2

4 3 1 5 2 → 4 3 2 1 5

Lexicographical order of permutations

files  
abc  
ab

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

# Subsets

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

0  $\emptyset$

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

1

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

2

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

$2^n$  subsets

a b c

in in in

or

out out

$2 \times 2$

$2 \times 2 = 2^2$

2 pairs of shoes

2 pairs of socks

2 pairs

$\{a, b, c, d\} - 5 -$

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

or

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

$\{a, b\}$

$\rightarrow \{a, b, d\}$

all at once

index



Subset on demand

a	b	c	
0	0	0	0
0	0	0	1
1	0	0	2
0	1	0	3
0	0	1	4
0	1	1	5
1	0	1	6
1	1	1	7

all possible  
values of n-bit  
000

Socks  
 RL RL  
 RL RL  
 none

~~1A = 64~~  
 n elements  
 n bit int.  
 for (i = 0; i <= 2<sup>n</sup> + 1)  
 {  
 convert i to bit,  
 create subset that  
 corresp. to it.  
 }  
 next subset (int i)  
 return i + 1;

for ( i=0; i < 10<sup>6</sup>; ++i ) {  
 C \* P = new C(i); // malloc  
 do work  
 }  
 slow

delete P; // throws away // free

char \* buffers = new char[ sizeof(C) ];  
 for ( i=0; i < 10<sup>6</sup>; ++i ) {  
C \* P = new ( buffers ) C(i); // char

P ~ C(); // after // salvage memory  
 delete [] buffers;

`c(i); // default`

-8-

for ( $i=0$ ;  $i < 10^6$ ;  $++i$ ) {

`c.seinit(i);` // `c(0) → c(1)` is very cheap

// only changes few byte;

$n \rightarrow \underbrace{n+1}_{\text{subset}}$

3

subset for each subset

order subset so just 2

Get change;

Exon code

0 0 0 0 0 0

000	000	000	000	000	000
000	001	001	001	001	001
001	002	002	002	002	002
002	003	003	003	003	003
003	004	004	004	004	004
004	005	005	005	005	005
005	006	006	006	006	006
006	007	007	007	007	007
007	008	008	008	008	008
008	009	009	009	009	009
009	010	010	010	010	010
010	011	011	011	011	011
011	012	012	012	012	012
012	013	013	013	013	013
013	014	014	014	014	014
014	015	015	015	015	015
015	016	016	016	016	016
016	017	017	017	017	017
017	018	018	018	018	018
018	019	019	019	019	019
019	020	020	020	020	020
020	021	021	021	021	021
021	022	022	022	022	022
022	023	023	023	023	023
023	024	024	024	024	024
024	025	025	025	025	025
025	026	026	026	026	026
026	027	027	027	027	027
027	028	028	028	028	028
028	029	029	029	029	029
029	030	030	030	030	030
030	031	031	031	031	031
031	032	032	032	032	032
032	033	033	033	033	033
033	034	034	034	034	034
034	035	035	035	035	035
035	036	036	036	036	036
036	037	037	037	037	037
037	038	038	038	038	038
038	039	039	039	039	039
039	040	040	040	040	040
040	041	041	041	041	041
041	042	042	042	042	042
042	043	043	043	043	043
043	044	044	044	044	044
044	045	045	045	045	045
045	046	046	046	046	046
046	047	047	047	047	047
047	048	048	048	048	048
048	049	049	049	049	049
049	050	050	050	050	050
050	051	051	051	051	051
051	052	052	052	052	052
052	053	053	053	053	053
053	054	054	054	054	054
054	055	055	055	055	055
055	056	056	056	056	056
056	057	057	057	057	057
057	058	058	058	058	058
058	059	059	059	059	059
059	060	060	060	060	060
060	061	061	061	061	061
061	062	062	062	062	062
062	063	063	063	063	063
063	064	064	064	064	064
064	065	065	065	065	065
065	066	066	066	066	066
066	067	067	067	067	067
067	068	068	068	068	068
068	069	069	069	069	069
069	070	070	070	070	070
070	071	071	071	071	071
071	072	072	072	072	072
072	073	073	073	073	073
073	074	074	074	074	074
074	075	075	075	075	075
075	076	076	076	076	076
076	077	077	077	077	077
077	078	078	078	078	078
078	079	079	079	079	079
079	080	080	080	080	080
080	081	081	081	081	081
081	082	082	082	082	082
082	083	083	083	083	083
083	084	084	084	084	084
084	085	085	085	085	085
085	086	086	086	086	086
086	087	087	087	087	087
087	088	088	088	088	088
088	089	089	089	089	089
089	090	090	090	090	090
090	091	091	091	091	091
091	092	092	092	092	092
092	093	093	093	093	093
093	094	094	094	094	094
094	095	095	095	095	095
095	096	096	096	096	096
096	097	097	097	097	097
097	098	098	098	098	098
098	099	099	099	099	099
099	100	100	100	100	100

minim

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

MIN. CHANGE

ORDER



for  $(i=0; i < (1 \leq n); ++i)$   $\{$   
 $\quad + \text{bits}(i \wedge (i < 2));$

~