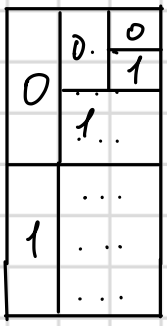
Комбинаторика

двоичные вектора фикс. длины

$\mathbb{B}^n$

000
001
...
...
111



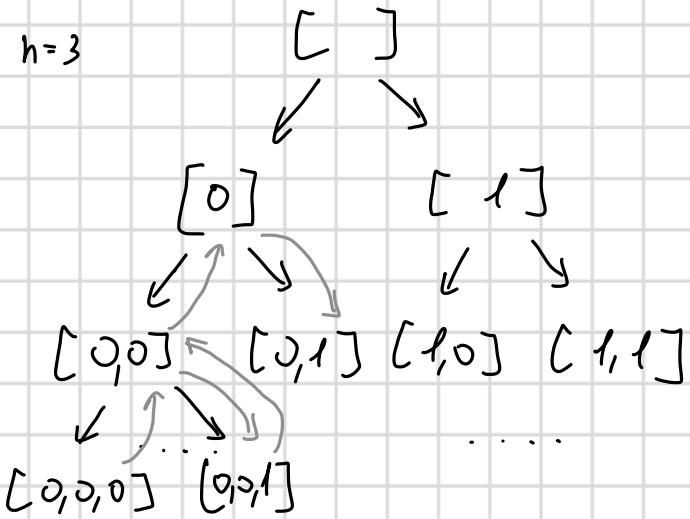
n левых n левых кодов

gen

gen all, starting with 0

gen all, starting with 1

n=3



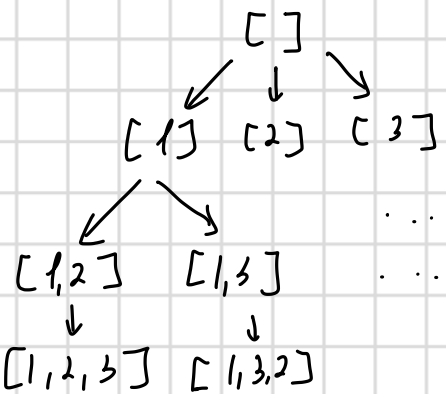

```
gen(p):
    if len(p) == n:
        print(p)
        return
    gen(p + [0])
    gen(p + [1])
```

```
main:
    gen([])
```

Перестановки

n = 3

123
132
213
231
312
321



n эл-ов $\{1 \dots n\} = M$

$P_n = \{ v \in M^n : v_i \neq v_j \text{ при } i \neq j \}$

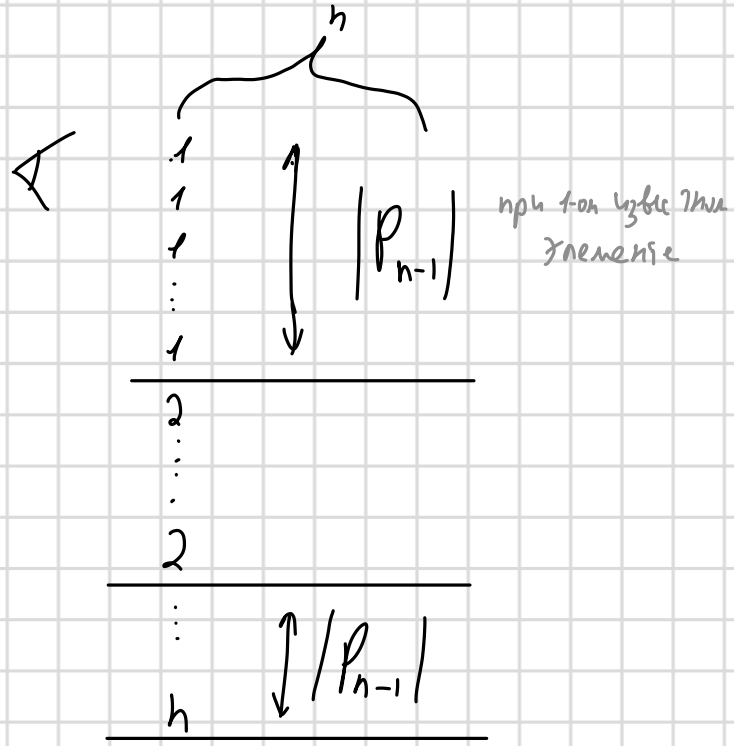
$n = 3$

123	1
132	2
213	3
231	4
312	5
321	6

$|P_n| = ?$

$|P_1| = 1$

$|P_2| = 2$



Правило суммирования:

$A = B \sqcup C$

$|A| = |B| + |C|$

$$|P_n| = \sum_{k=1}^{n} |P_{n-1}| = n \cdot |P_{n-1}|$$

$$|P_n| = n!$$

```
gen(p):
    if len(p) == n:
        print(p)
        return
    for i = 1...n:
        if i not in p:
            gen(p+[i])
```

```
main:
    gen([])
```

∀ строки из 0 и 1 длины $\leq n$

$\bigcup_{K=0}^{n} \mathbb{B}^K$ ε - пуст. стр.

0
00
000
001
01
010
011

```
gen(p):
    print(p)
    if len(p) == n: return
    gen(p+[0])
    gen[p+[1])
```

```
gen(p)   p - префикс К.О.
    if p - К.О.: print(p)
    for c in Σ (sorted):
        if p+[c] - префикс К.О.:
            gen(p+[c])
```

К.О. - комбинаторные объекты

$\exists C \subseteq \Sigma^*$

(пример: 1) строки посл длины
↳ $\Sigma = \mathbb{B}$
2) $\leq n$
3) перестановка ↳ $\Sigma = \{1...n\}$)

Размещения

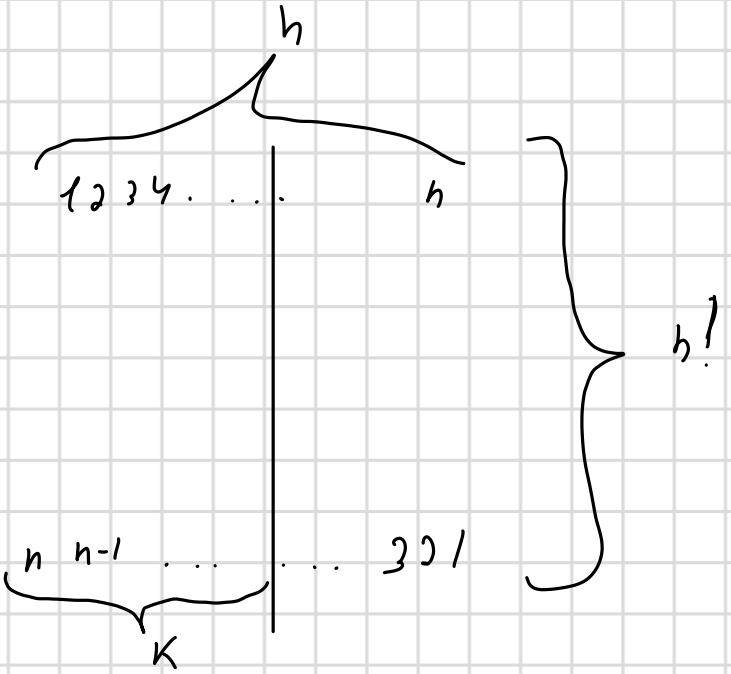
n эл. k мест $\quad k \le n$

$M = \{1 \dots n\}$

$\{ V \in M^k : V_i = V_j$ при $i \neq j \}$

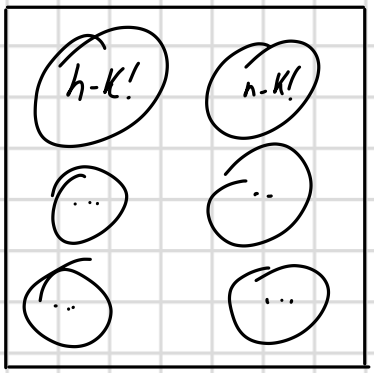
Пр $n = 4, \ k = 2$

1 | 1 2
2 | 1 3
3 | 1 4
4 | 2 1
5 | 2 3
6 | 2 4
⋮
12 | 4 3



Хвостов: $(n-k)!$

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



$$A_n^k = \frac{n!}{(n-k)!}$$

Сочетания

$$\binom{n}{k} \Leftrightarrow C_n^k = \frac{A_n^k}{k!} = \frac{n!}{k!(n-k)!}$$

$\sum = \{1, 2, \dots n\}$ $\quad \{ v \in M^k : v_i < v_j$ при $i < j \}$

$\{1,2\}$ ← корректная запись т.к. возрастает

$\{2,1\}$ ← нет

```
gen(p)   p - префикс к.о.
    if len(p) == k:
        print(p)
        return
    for c = 1....n:   можно c = p[-1]+1 ... n-k+len(p)+1  и 2 if не нужен
        if p ≠ [] and c ≤ p[-1]:
            continue
        if n-c < k-len(p)-1:
            break
        gen(p+[c])
```

Правильные скобочные посл-ти

(( )) ( )

баланс = #(− #) ≥ 0

в конце баланс = 0

n=1 ()

n=2 ()() (())

n=3 ((())) (()()) (())() ()(()) ()()()

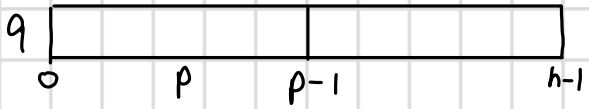
$\sum = \{ (,) \}$

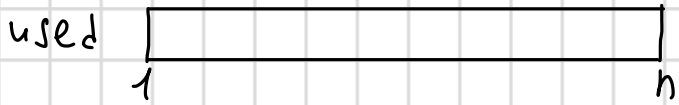
(<)

```
gen (p, bal)
    if len(p) == 2n
        print(p)
        return
    if bal+1 ≤ 2n-len(p)-1:
        gen(p+[(], bal+1)
    if bal ≠ 0:
        gen(p+[)], bal-1)
```

```
main():
    gen([], 0)
```

Перестановки






```
gen(p)   длина префикса
    if p == n:
        print(a); return
```

```
for i = 1...n
    if !used[i]
        used[i] = true
        a[p] = i
        gen(p+1)
        used[i] = false
```

K

0 1 2 3
1 1 3 2
2 2 1 3
3 2 3 1
4 3 1 2
5 3 2 1

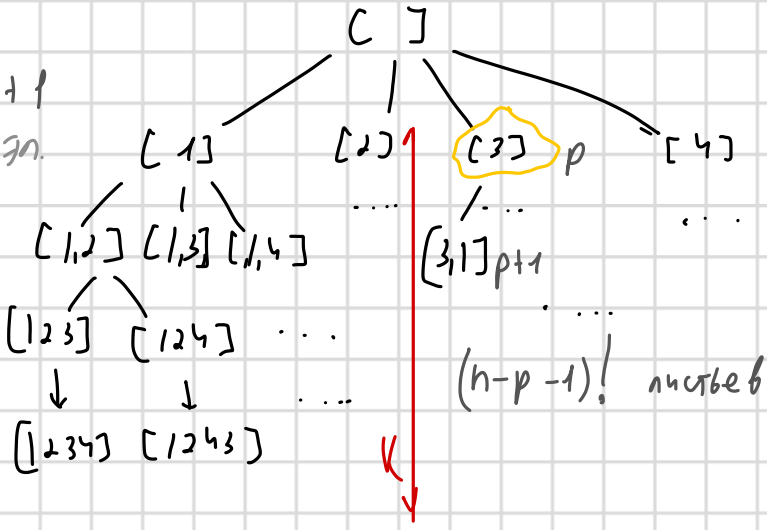
gen(p) $O(n^2) \longrightarrow O(n \log n)$

```
if p == n
    ~~if K == 0:~~ не нужно из-за след. проверок
        print(a)
    K--
    return
for i = 1...n
    if !used[i]
        if K >= (n-p-1)!: K -= (n-p-1)!
        else:
            used[i] = true
            a[p] = i
            gen(p+1)
            return
```

$C = \lfloor K / (n-p-1)! \rfloor + 1$

взять с-е не исп. эл.

$K = K \% (n-p-1)!$



gen(p)   p – префикс К.О.

```
gen(p)
    if p – К.О.:
        if k==0: print(p); return
        k--
    for c in Σ (sorted):
        if p+[c] – префикс К.О.:
            t = кол-во К.О. с преф p+[c]
            if k≥t: k-=t
            else:
                gen(p+[c])
                return
```

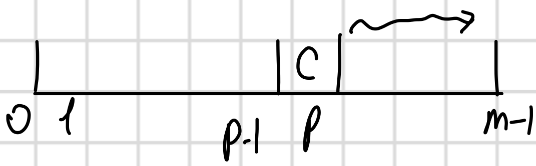
Сложность: $len \cdot |\Sigma|$

Сочетания

gen(p): $O(nm+nm)$ [обозначение $\binom{n}{m} = C_n^m$]

```
gen(p):
    if p==n: print(a); return
    for c = 1 или a[p-1]+1 ... n-m+p+1
        t = C(n-c, m-p-1)
        if k≥t: k-=t
        else: gen(p+1)
```

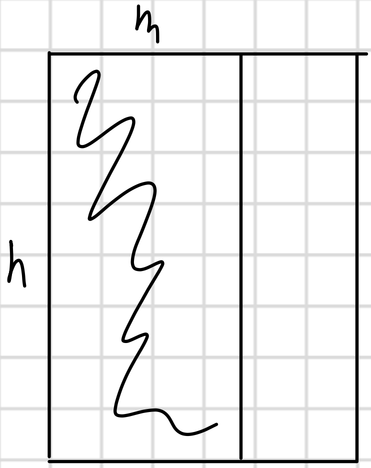
$t = \binom{n-c}{m-p-1}$



m-p-1

c+1...n

n-c



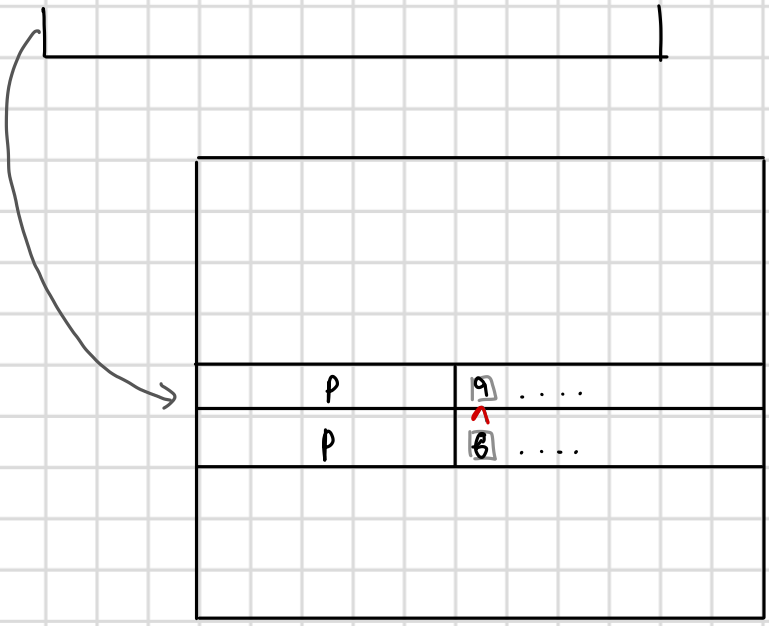
можно сгенерировать все сочетания с пом. треуг. Паск. и обращаться

z — к.о. хотим найти номер z

```
gen(p)
    if p - к.о.:
        if p == z          плохой код
            res = k
        else: k++
    for c in ∑
        if (p+[c] - преф к.о.):
            gen(p+[c])
```

```
gen(p)   p - префикс к.о.
    if p - к.о:
        if k == z: print(p); return
        k++                          хороший код
    for c in ∑ (sorted):
        if p+[c] - префикс к.о.:
            t = кол-во к.о. с преф p+[c]
            if p+[c] - префикс z:
                gen(p+[c])
                return
            else:
                k += t
```

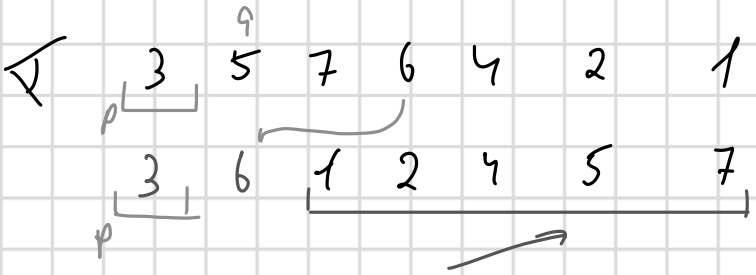
Следующий К.О.



1) макс префикс, который можно сохр увеличив след.
2) увел-ть след. эл. мин-м образом превратив а в в
3) приписать p+в минимальный „хвост"

Пр.: 01011011011

01011011100



Код Грея

• любые 2 соседние строки отл в 1 месте

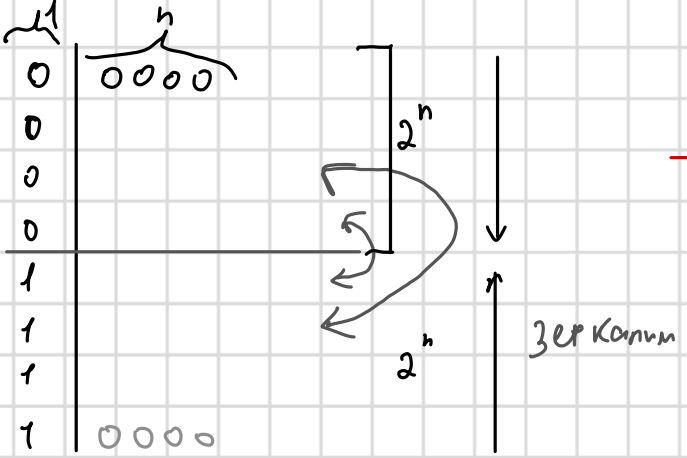
Факт: ∀n ∃ код Грея для ∀в

▷ n=1 n=2

0
1

00
01
11
10

если 1 и посл. К.О. тоже отл в 1 месте — циклический код Грея



Зеркальный код Грея

$n = 3$

0	0 0
0	0 1
0	1 1
0	1 0
1	1 0
1	1 1
1	0 1
1	0 0

0
1
1
0

Если не получается разбиение

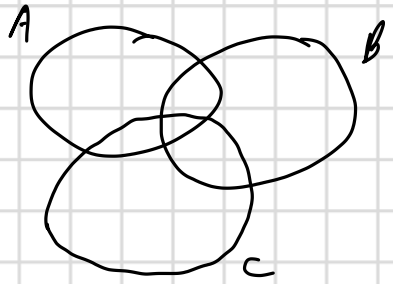
$A = B \sqcup C$ если все хорошо

Иначе

$\exists A = B \vee B$

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

$|A+B+C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$



Формула включения исключений

$$|A_1 \cup A_2 \cup \dots \cup A_n| = \sum_i |A_i| - \sum_{i<j} A_i A_j + \sum_{i<j<k} |A_i A_j A_k| +$$

$$+ \dots + |A_1 A_2 \dots A_n|$$