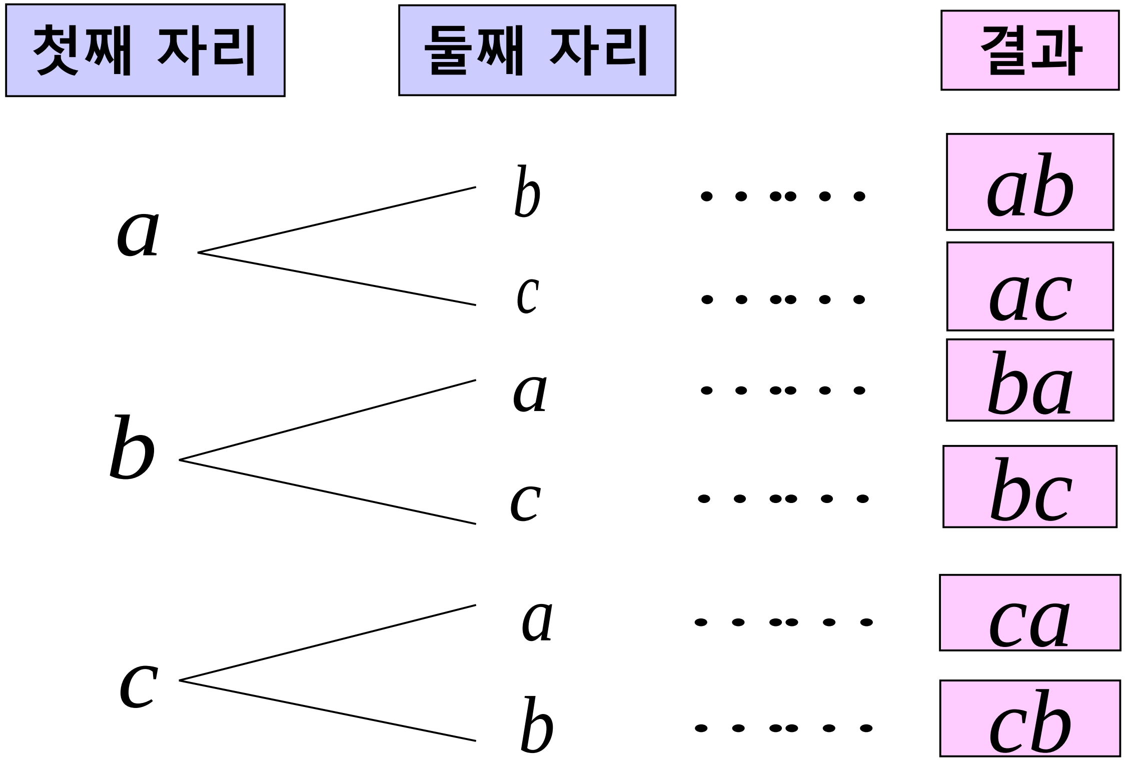
순열, 조합

세 개의 문자 a, b, c 중 두 개를 택하여 일렬로 배열하는 방법의 수는?



- 따라서, 앞의 결과는 첫째 자리에 들어갈 수 있는 경우의 수에다 둘째 자리에 들어갈 수 있는 경우의 수를 곱의 법칙에 의하여 곱한 가지 수가 된다.

$$\therefore 3 \times 2 = 6 \text{(가지)}$$

- 이것을 서로 다른 3개에서 2개를 택하는 순열 이라 한다.

순열

- 서로 다른 n 개에서 r 개를 택하여 일렬로 나열하는 방법을 n개에서 r개를 택하는 순열이라 하고, 이 순열의 수를 ${}_nP_r$ 로 나타낸다.

$${}_nP_r = n(n-1)(n-2)\cdots(n-r+1) \quad (\text{단, } r \le n)$$

${}_nP_r$ 에서 $r = n$ 이면,

$${}_nP_n = n\cdot(n-1)\cdot(n-2)\cdots2\cdot1 = n!$$

$${}_nP_r = \frac{n!}{(n-r)!} \qquad 0! = 1 \qquad {}_nP_0 = 1$$

- 1, 2, 3, 4, 5 의 다섯 개의 숫자 중에서 서로 다른 세 숫자를 이용하여 만들 수 있는 세 자리의 자연수는 모두 몇 개인가?

순열의 점화식

$$_{n}P_{r} = n \cdot {}_{n-1}P_{r-1}$$

- 1, 2, 3, 4 네 개의 숫자에서 세 개를 뽑는 순열을 생각해 보자.
- 먼저 4가 마지막에 있는 경우 (X, Y, 4)는 나머지 1, 2, 3에서 두 개의 위치를 채우면 된다.
- 그런데 4 이외에 1, 2, 3가 맨 마지막에 있는 경우 (X, Y, 1), (X, Y, 2), (X, Y, 3)도 생각해야 하므로 결국 마지막에 있는 수를 제외한 나머지 세 개의 숫자에서 두 개의 순열을 뽑으면 된다.
- 이 경우의 수가 위의 점화식이다.

1, 2, 3, 4 네 개의 숫자에서 세 개를 뽑는 순열

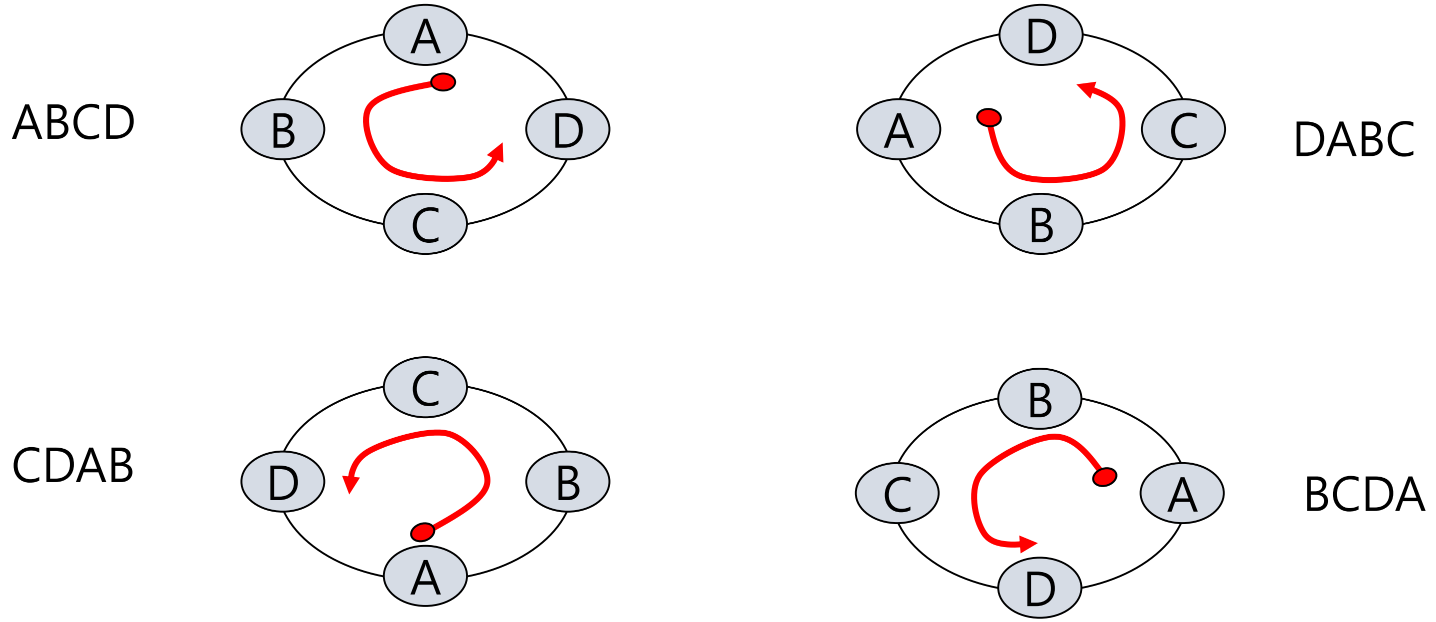
```
int t[10];
int a[10]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

void Perm(int n, int r, int q)
{
        if(r == 0) print(q);
        else {
                for(int i = n-1; i >= 0; i--) {
                        swap(&a[i], &a[n-1]);
                        t[r-1] = a[n-1];
                        Perm(n-1, r-1, q);
                        swap(&a[i], &a[n-1]);
                }
        }
}
void main(void)
{       Perm(4, 3, 3);     }
```

```
void print(int q)
{
        while(q)printf(" %d", t[--q]);
        printf("\n")
}
```

원순열

- 서로 다른 n개의 원소를 원형으로 배열하는 것을 원순열이라 한다.
- 또 이를 계산하는 방법은 $(n-1)!$



중복 순열

- 서로 다른 n개의 **중복을 허용하여** r개를 택하여 일렬로 나열하는 방법을 n개에서 r개를 택하는 **중복순열** 이라 한다.

$$_{n}\Pi_{r} = n^{r}$$

- 서로 다른 3 개의 과일 사과, 배, 수박이 있다. 2개를 택하여 일렬로 배열할 때 중복을 허용하여 나열한 순열의 개수는?
  - (사과, 사과), (배,배), (수박,수박) 의 3가지를 더 생각할 수 있다.

- 중복 순열의 점화식

$$_{n}\Pi_{r} = n \cdot {}_{n}\Pi_{r-1}$$

중복 순열 구하기 코드 예

```
def PI(n, r, q):
    if r == 0:
        myprint(q)
    else:
        for i in range(n-1, -1, -1):
            A[i], A[n-1] = A[n-1], A[i]
            T[r-1] = A[n-1]
            PI(n, r-1, q)
            A[i], A[n-1] = A[n-1], A[i]

PI(4, 3, 3)
```

```
def myprint(q):
    while q != 0:
        q = q - 1
        print(" %d" % (T[q]), end='')
    print("")
```

조합

- 서로 다른 n 개에서 순서를 생각하지 않고 r 개 (0 <= r <= n)를 택하는 것을 조합이라 한다

$$ {}_nP_r = {}_nC_r \times r! $$

$$ {}_nC_r = \frac{{}_nP_r}{r!} = \frac{n!}{r!(n-r)!}, \quad {}_nC_0 = 1 $$

$$ {}_nC_r = {}_nC_{n-r} $$

서로 다른 *n* 개의 물건을
*p*개, *q*개, *r*개 (*p*+*q*+*r* = *n*)
의 3개조로 나누는 방법의 수

$$ {}_nC_p \times {}_{n-p}C_q \times {}_rC_r $$

- 서로 다른 종류의 꽃 15송이를 다섯 송이씩 세 묶음으로 나누는 방법의 수는?

다음 식을 생각해 보자

$$(a+b)^4 = (a+b)(a+b)(a+b)(a+b)$$
$$= a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4$$

- 4개의 인수 (a+b) 로 부터 각각 a 또는 b 를 하나씩 택하여 곱한 것이다.
- 예를 들어, ab^3 항은 4개의 (a+b) 중에서 3개로부터 b를 택하고, 나머지 하나는 a를 택하여 곱한 것. 즉, ${}_4C_3 \times {}_1C_1 = 4$

이항정리

$$(a+b)^n = \boxed{{}_nC_0}\,a^n + \boxed{{}_nC_1}\,a^{n-1}b + \boxed{{}_nC_2}\,a^{n-2}b^2 +$$

$$\cdots + \boxed{{}_nC_r}\,a^{n-r}b^r + \cdots + \boxed{{}_nC_n}\,a^0b^n$$

$$= \sum_{r=0}^{n} {}_nC_r\,a^{n-r}b^r \rightarrow \text{일반항}$$

이항계수

파스칼의 삼각형

$(a+b)$　　1　1

$(a+b)^2$　　1　2　1

$(a+b)^3$　　1　3　3　1

$(a+b)^4$　　1　4　6　4　1

$(a+b)^5$　　1　5　10　10　5　1

$\cdots$　　$\cdots\cdots\cdots$

조합의 점화식

$$ {}_nC_r = {}_{n-1}C_{r-1} + {}_{n-1}C_r \quad {}_nC_0 = 1 $$

```
def Combination(n, r, q):
    if r == 0:
        myprint(q)
    else:
        if n < r:
            return
        else:
            T[r-1] = A[n-1]
            Combination(n-1, r-1, q)
            Combination(n-1, r, q)
```

```
def myprint(q):
    while q != 0:
        q = q - 1
        print(" %d" % (T[q]), end='')
    print("")
```

중복조합

- 서로 다른 n 개에서 **중복을 허락**하여 r 개를 택하는 조합을 **중복조합** 이라 하고, 이 중복조합의 수를 ${}_{n}H_{r}$ 로 나타낸다

$$ {}_{n}H_{r} = {}_{n+r-1}C_{r} $$

- 중복조합의 점화식

$$ {}_{n}H_{r} = {}_{n}H_{r-1} + {}_{n-1}H_{r} $$

- 예 : 숫자 1, 2에서 중복을 허락하여 3개를 택하는 조합은?
  - (1,1,1), (1,1,2), (1,2,20, (2,2,2)

순열, 중복순열, 조합, 중복조합의 차이점

- ${}_{n}P_{r}$: 중복을 허락하지는 않지만, 순서는 생각한다.
- ${}_{n}\Pi_{r}$: 중복을 허락하고 순서도 생각한다.
- ${}_{n}C_{r}$: 중복을 허락하지 않고 순서도 생각하지 않는다.
- ${}_{n}H_{r}$: 중복을 허락하고 순서도 생각하지 않는다.