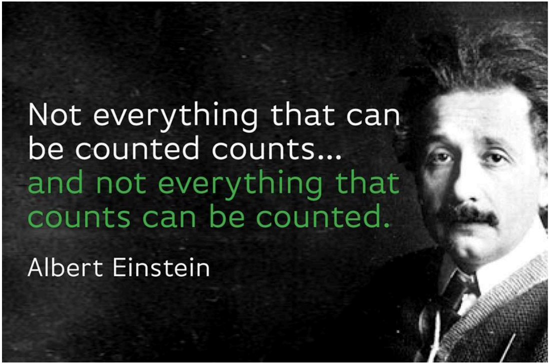
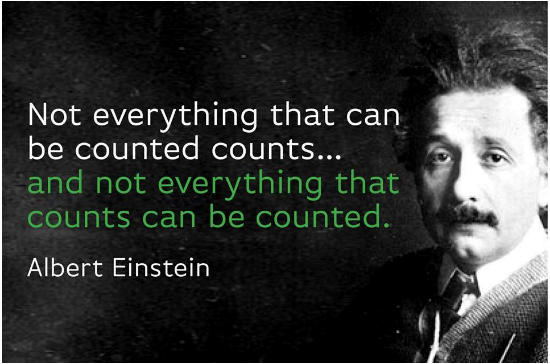
2-3 Counting

魏恒峰

hfwei@nju.edu.cn

2018 年 04 月 11 日


Not everything that can be counted counts...
and not everything that counts can be counted.
Albert Einstein



所以，学好“2-3 组合与计数”是多么重要!

Paring up (CS : $1.2 - 15$)

A tennis club has $2n$ members. We want to pair up the members by twos for singles matches.

(a) In how many ways can we pair up all the members of the club?

(b) Suppose that we also determine who serves first for each pairing. In how many ways can we specify our pairs?

Paring up (CS : $1.2 - 15$)

A tennis club has $2n$ members. We want to pair up the members by twos for singles matches.

(a) In how many ways can we pair up all the members of the club?

(b) Suppose that we also determine who serves first for each pairing. In how many ways can we specify our pairs?

$$\frac{1}{n!}\binom{2n}{\underbrace{2, 2, \cdots, 2}_{n}}$$

Paring up (CS : $1.2 - 15$)

A tennis club has $2n$ members. We want to pair up the members by twos for singles matches.

(a) In how many ways can we pair up all the members of the club?

(b) Suppose that we also determine who serves first for each pairing. In how many ways can we specify our pairs?

$$\frac{1}{n!}\binom{2n}{\underbrace{2, 2, \cdots, 2}_{n}} = \frac{(2n)!}{\underbrace{2^n}_{\text{intra-pair}} \cdot \underbrace{n!}_{\text{inter-pair}}}$$

Paring up (CS : $1.2 - 15$)

A tennis club has $2n$ members. We want to pair up the members by twos for singles matches.

(a) In how many ways can we pair up all the members of the club?

(b) Suppose that we also determine who serves first for each pairing. In how many ways can we specify our pairs?

$$\frac{1}{n!}\binom{2n}{\underbrace{2, 2, \cdots, 2}_{n}} = \frac{(2n)!}{\underbrace{2^n}_{\text{intra-pair}} \cdot \underbrace{n!}_{\text{inter-pair}}}$$

$$\frac{(2n)!}{2^n \cdot n!} \cdot 2^n = \frac{(2n)!}{n!}$$

Passing out Apples to Children

k-Permutation $(\text{CS} : 1.2 - 5)$

We need to pass out k distinct apples (pieces of fruit) to n children such that *each child may get at most one apple*.

(a) $k \leq n$?

(b) What if $k > n$?

k-Permutation $(\text{CS} : 1.2 - 5)$

We need to pass out k distinct apples (pieces of fruit) to n children such that *each child may get at most one apple*.

(a) $k \le n$?

(b) What if $k > n$?

$$n^{\underline{k}} \triangleq n(n-1)\cdots(n-k+1)$$

k-Permutation $(\mathrm{CS}: 1.2-5)$

We need to pass out k distinct apples (pieces of fruit) to n children such that *each child may get at most one apple*.

(a) $k \le n$?

(b) What if $k > n$?

$$n^{\underline{k}} \triangleq n(n-1)\cdots(n-k+1)$$

$$0$$

Multisets (CS : $1.5 - 4$)

Use multisets to determine the number of ways to pass out k **identical** apples to n children. Assume that a child may get more than one apple.

Multisets (CS : $1.5-4$)

Use multisets to determine the number of ways to pass out k **identical** apples to n children. Assume that a child may get more than one apple.

x_i : the # of apples the i-th child gets

$$x_1 + x_2 + \cdots + x_n = k, \qquad x_i \geq 0$$

Multisets (CS : $1.5 - 4$)

Use multisets to determine the number of ways to pass out k **identical** apples to n children. Assume that a child may get more than one apple.

x_i : the # of apples the i-th child gets

$$x_1 + x_2 + \cdots + x_n = k, \qquad x_i \geq 0$$

$$y_i \triangleq x_i + 1$$

Multisets (CS : $1.5 - 4$)

Use multisets to determine the number of ways to pass out k **identical** apples to n children. Assume that a child may get more than one apple.

$$x_i : \text{the \# of apples the } i\text{-th child gets}$$

$$x_1 + x_2 + \cdots + x_n = k, \qquad x_i \geq 0$$

$$y_i \triangleq x_i + 1$$

$$y_1 + y_2 + \cdots + y_n = n + k, \qquad y_i \geq 1$$

Multisets (CS : $1.5-4$)

Use multisets to determine the number of ways to pass out k **identical** apples to n children. Assume that a child may get more than one apple.

x_i : the # of apples the i-th child gets

$$x_1 + x_2 + \cdots + x_n = k, \qquad x_i \geq 0$$

$$y_i \triangleq x_i + 1$$

$$y_1 + y_2 + \cdots + y_n = n + k, \qquad y_i \geq 1$$

$$\binom{n+k-1}{n-1} = \binom{n+k-1}{k}$$

Multisets $(\text{CS} : 1.5 - 4)$

Use **multisets** to determine the number of ways to pass out k identical apples to n children. Assume that a child may get more than one apple.

Multisets $(\text{CS}: 1.5 - 4)$

Use **multisets** to determine the number of ways to pass out k identical apples to n children. Assume that a child may get more than one apple.

Q : k-multiset of $[1 \cdots n]$ *vs.* n-multiset of $[1 \cdots k]$

Multisets $(\text{CS} : 1.5-4)$

Use **multisets** to determine the number of ways to pass out k identical apples to n children. Assume that a child may get more than one apple.

Q : k-multiset of $[1 \cdots n]$ *vs.* n-multiset of $[1 \cdots k]$

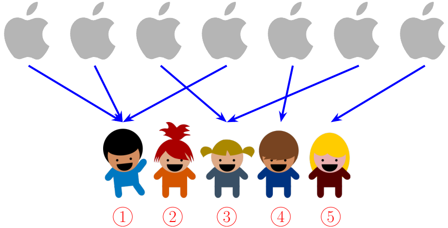
$$k = 7 \qquad n = 5$$

Multisets (CS : $1.5 - 4$)

Use **multisets** to determine the number of ways to pass out k identical apples to n children. Assume that a child may get more than one apple.

Q : k-multiset of $[1 \cdots n]$ *vs.* n-multiset of $[1 \cdots k]$

$$k = 7 \qquad n = 5$$

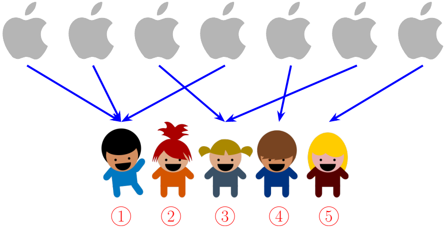


Multisets (CS : $1.5-4$)

Use **multisets** to determine the number of ways to pass out k identical apples to n children. Assume that a child may get more than one apple.

Q : k-multiset of $[1 \cdots n]$ *vs.* n-multiset of $[1 \cdots k]$

$$k = 7 \qquad n = 5$$



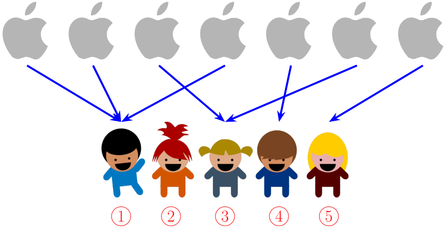
$$\{1, 1, 1, 3, 3, 4, 5\}$$

Multisets (CS : $1.5-4$)

Use **multisets** to determine the number of ways to pass out k identical apples to n children. Assume that a child may get more than one apple.

Q : k-multiset of $[1 \cdots n]$ *vs.* n-multiset of $[1 \cdots k]$

$$k = 7 \qquad n = 5$$



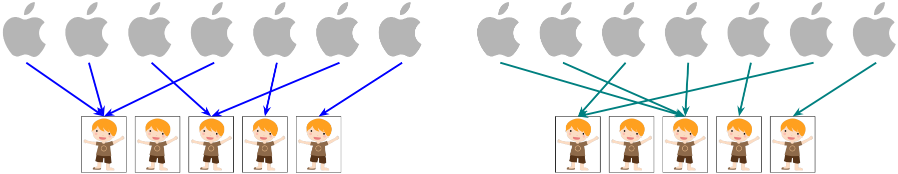
$$\{1, 1, 1, 3, 3, 4, 5\}$$

Integer Partition (CS : $1.5 - 4$ Extended)

What is the number of ways to pass out k identical apples to n-胞胎.
Assume that a child may get more than one apple.

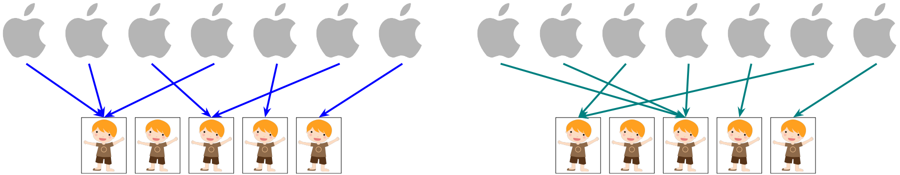
Integer Partition (CS : $1.5-4$ Extended)

What is the number of ways to pass out k identical apples to n-胞胎. Assume that a child may get more than one apple.



Integer Partition (CS : $1.5-4$ Extended)

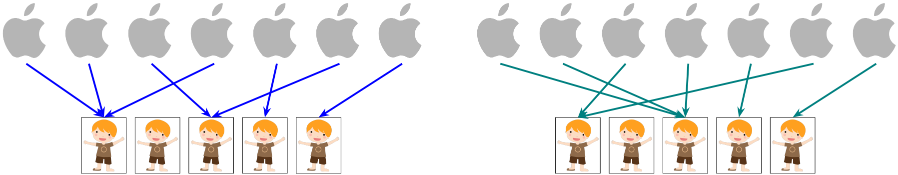
What is the number of ways to pass out k identical apples to n-胞胎.
Assume that a child may get more than one apple.



Integer partition of k into $\leq n$ parts

Integer Partition (CS : $1.5 - 4$ Extended)

What is the number of ways to pass out k identical apples to n-胞胎.
Assume that a child may get more than one apple.



Integer partition of k into $\leq n$ parts

Theorem (G. H. Hardy, Ramanujan (1918))

$$p(k) \triangleq \sum_{x=1}^{x=n} p_x(k) \sim \frac{1}{4\sqrt{3}n} \exp\left(\pi\sqrt{\frac{2n}{3}}\right)$$

$p_n(k)$: # of partitions of k into n parts

Theorem (Recurrence for $p_n(k)$)

$$p_n(k) = p_{n-1}(k-1) + p_n(k-n)$$

$p_n(k)$: # of partitions of k into n parts

Theorem (Recurrence for $p_n(k)$)

$$p_n(k) = p_{n-1}(k-1) + p_n(k-n)$$

Proof.

$$1 \le x_1 \le x_2 \le \cdots \le x_n$$

Case $x_1 > 1$

$$1 < x_1 \le x_2 \le \cdots \le x_n$$

Case $x_1 = 1$

$$1 = x_1 \le x_2 \le \cdots \le x_n$$

$p_n(k)$: # of partitions of k into n parts

Theorem (Recurrence for $p_n(k)$)

$$p_n(k) = p_{n-1}(k-1) + p_n(k-n)$$

Proof.

$$1 \le x_1 \le x_2 \le \cdots \le x_n$$

CASE $x_1 > 1$

$$1 < x_1 \le x_2 \le \cdots \le x_n$$

CASE $x_1 = 1$

$$1 = x_1 \le x_2 \le \cdots \le x_n$$

$$\boxed{p_{n-1}(k-1)}$$

$p_n(k)$: # of partitions of k into n parts

Theorem (Recurrence for $p_n(k)$)

$$p_n(k) = p_{n-1}(k-1) + p_n(k-n)$$

Proof.

$$1 \le x_1 \le x_2 \le \cdots \le x_n$$

CASE $x_1 = 1$

$$1 = x_1 \le x_2 \le \cdots \le x_n$$

$$\boxed{p_{n-1}(k-1)}$$

CASE $x_1 > 1$

$$1 < x_1 \le x_2 \le \cdots \le x_n$$

$$1 \le x_1 - 1 \le x_2 - 1 \le \cdots \le x_n - 1$$

$p_n(k)$: # of partitions of k into n parts

Theorem (Recurrence for $p_n(k)$)

$$p_n(k) = p_{n-1}(k-1) + p_n(k-n)$$

Proof.

$$1 \le x_1 \le x_2 \le \cdots \le x_n$$

Case $x_1 = 1$

$$1 = x_1 \le x_2 \le \cdots \le x_n$$

$$\boxed{p_{n-1}(k-1)}$$

Case $x_1 > 1$

$$1 < x_1 \le x_2 \le \cdots \le x_n$$

$$1 \le x_1 - 1 \le x_2 - 1 \le \cdots \le x_n - 1$$

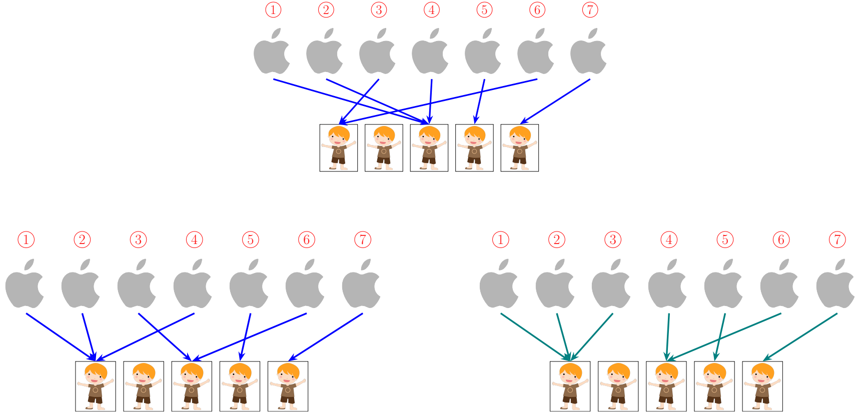
$$\boxed{p_n(k-n)}$$

Set Partition (CS : $1.5 - 4$ Extended)

What is the number of ways to pass out k distinct apples to n-胞胎.
Assume that a child may get more than one apple.

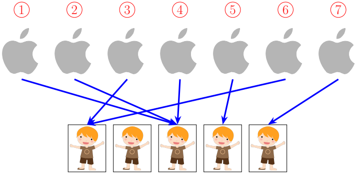
Set Partition (CS : $1.5 - 4$ Extended)

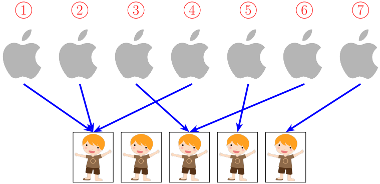
What is the number of ways to pass out k distinct apples to n-胞胎.
Assume that a child may get more than one apple.

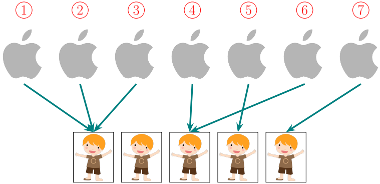


Set Partition (CS : $1.5 - 4$ Extended)

What is the number of ways to pass out k distinct apples to n-胞胎.
Assume that a child may get more than one apple.







Set partition of $[1 \cdots k]$ into $\leq n$ parts

Set Partition (CS : $1.5-12$)

$$S(n,k)\ \left(\left\{ n \atop k \right\}\right) : \#\text{ of set partitions of } [1\cdots n] \text{ into } k \text{ classes}$$

Set Partition (CS : $1.5 - 12$)

$$S(n,k) \; \left(\left\{ {n \atop k} \right\}\right) : \# \text{ of set partitions of } [1 \cdots n] \text{ into } k \text{ classes}$$

Stirling number of the second kind

Set Partition $(\mathsf{CS}: 1.5 - 12)$

$$S(n,k) \ (\left\{ {n \atop k} \right\}) : \# \text{ of set partitions of } [1 \cdots n] \text{ into } k \text{ classes}$$

Stirling number of the second kind

Theorem (Recurrence for $S(n,k)$)

$$S(0,0) = 1, \quad S(n,0) = S(0,n) = 0 \ (n > 0)$$

$$S(n,k) = S(n-1,k-1) + kS(n-1,k), \quad n > 0, k > 0$$

Set Partition (CS : $1.5 - 12$)

$$S(n,k)\ \left(\left\{ n \atop k \right\}\right) : \#\text{ of set partitions of } [1 \cdots n] \text{ into } k \text{ classes}$$

Stirling number of the second kind

Theorem (Recurrence for $S(n,k)$)

$$S(0,0) = 1, \quad S(n,0) = S(0,n) = 0 \ (n > 0)$$

$$S(n,k) = S(n-1,k-1) + kS(n-1,k), \quad n > 0, k > 0$$

Proof.

$$S(n,k) = \underbrace{S(n-1,k-1)}_{n \text{ is alone}} + \underbrace{kS(n-1,k)}_{n \text{ is not alone}}$$

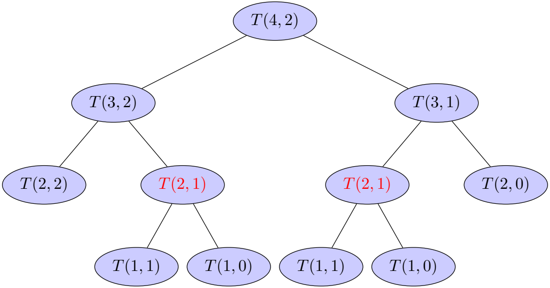
□

Computing $\binom{n}{k}$ (CS $1.5 : 14$)

```
procedure BINOM(n, k)                          ▷ Required: n ≥ k ≥ 0
    if k = 0 ∨ n = k then
        return 1
    return BINOM(n − 1, k) + BINOM(n − 1, k − 1)
```

Computing $\binom{n}{k}$ (CS $1.5 : 14$)

```
procedure Binom(n, k)                          ▷ Required: n ≥ k ≥ 0
    if k = 0 ∨ n = k then
        return 1
    return Binom(n − 1, k) + Binom(n − 1, k − 1)
```



```
procedure Binom(n, k)                          ▷ Required: n ≥ k ≥ 0
    if k = 0 ∨ n = k then
        return 1
    return Binom(n − 1, k) + Binom(n − 1, k − 1)
```

```
procedure BINOM(n, k)                          ▷ Required: n ≥ k ≥ 0
    if k = 0 ∨ n = k then
        return 1
    return BINOM(n − 1, k) + BINOM(n − 1, k − 1)
```

(i) # of "+":

```
procedure BINOM(n, k)                          ▷ Required: n ≥ k ≥ 0
    if k = 0 ∨ n = k then
        return 1
    return BINOM(n − 1, k) + BINOM(n − 1, k − 1)
```

(i) # of "+":

$$A(n,k) = 1 + A(n-1,k) + A(n-1,k-1)$$

procedure $\textsc{Binom}(n, k)$ ▷ Required: $n \geq k \geq 0$
    if $k = 0 \vee n = k$ **then**
        return 1
    return $\textsc{Binom}(n-1, k) + \textsc{Binom}(n-1, k-1)$

(i) # of "+":

$$A(n, k) = 1 + A(n-1, k) + A(n-1, k-1)$$

(ii) # of recursive calls of $\textsc{Binom}$:

procedure $\textsc{Binom}(n, k)$ ▷ Required: $n \geq k \geq 0$

  if $k = 0 \vee n = k$ **then**

    return 1

  return $\textsc{Binom}(n-1, k) + \textsc{Binom}(n-1, k-1)$

(i) # of "+":

$$A(n, k) = 1 + A(n-1, k) + A(n-1, k-1)$$

(ii) # of recursive calls of $\textsc{Binom}$:

$$R(n, k) = 2 + R(n-1, k) + R(n-1, k-1)$$

```
procedure BINOM(n, k)                         ▷ Required: n ≥ k ≥ 0
    if k = 0 ∨ n = k then
        return 1
    return BINOM(n − 1, k) + BINOM(n − 1, k − 1)
```

(i) # of "+":

$$A(n,k) = 1 + A(n-1,k) + A(n-1,k-1)$$

(ii) # of recursive calls of BINOM:

$$R(n,k) = 2 + R(n-1,k) + R(n-1,k-1)$$

$$\boxed{T(n,k) = T(n-1,k) + T(n-1,k-1) + c}$$

$$\binom{0}{0}$$

$$\binom{1}{0} \quad \binom{1}{1}$$

$$\binom{2}{0} \quad \binom{2}{1} \quad \binom{2}{2}$$

$$\binom{3}{0} \quad \binom{3}{1} \quad \binom{3}{2} \quad \binom{3}{3}$$

$$\binom{4}{0} \quad \binom{4}{1} \quad \binom{4}{2} \quad \binom{4}{3} \quad \binom{4}{4}$$

$$\binom{5}{0} \quad \binom{5}{1} \quad \binom{5}{2} \quad \binom{5}{3} \quad \binom{5}{4} \quad \binom{5}{5}$$

```
procedure BINOM(n, k)                          ▷ Required: n ≥ k ≥ 0
    for i ← 0 to n do
        B[i][0] ← 1
        B[i][i] ← 1
    for i ← 2 to n do
        for j ← 1 to k do
            B[n][k] ← B[n − 1][k] + B[n − 1][k − 1]
    return B[n][k]
```


Thank
You!