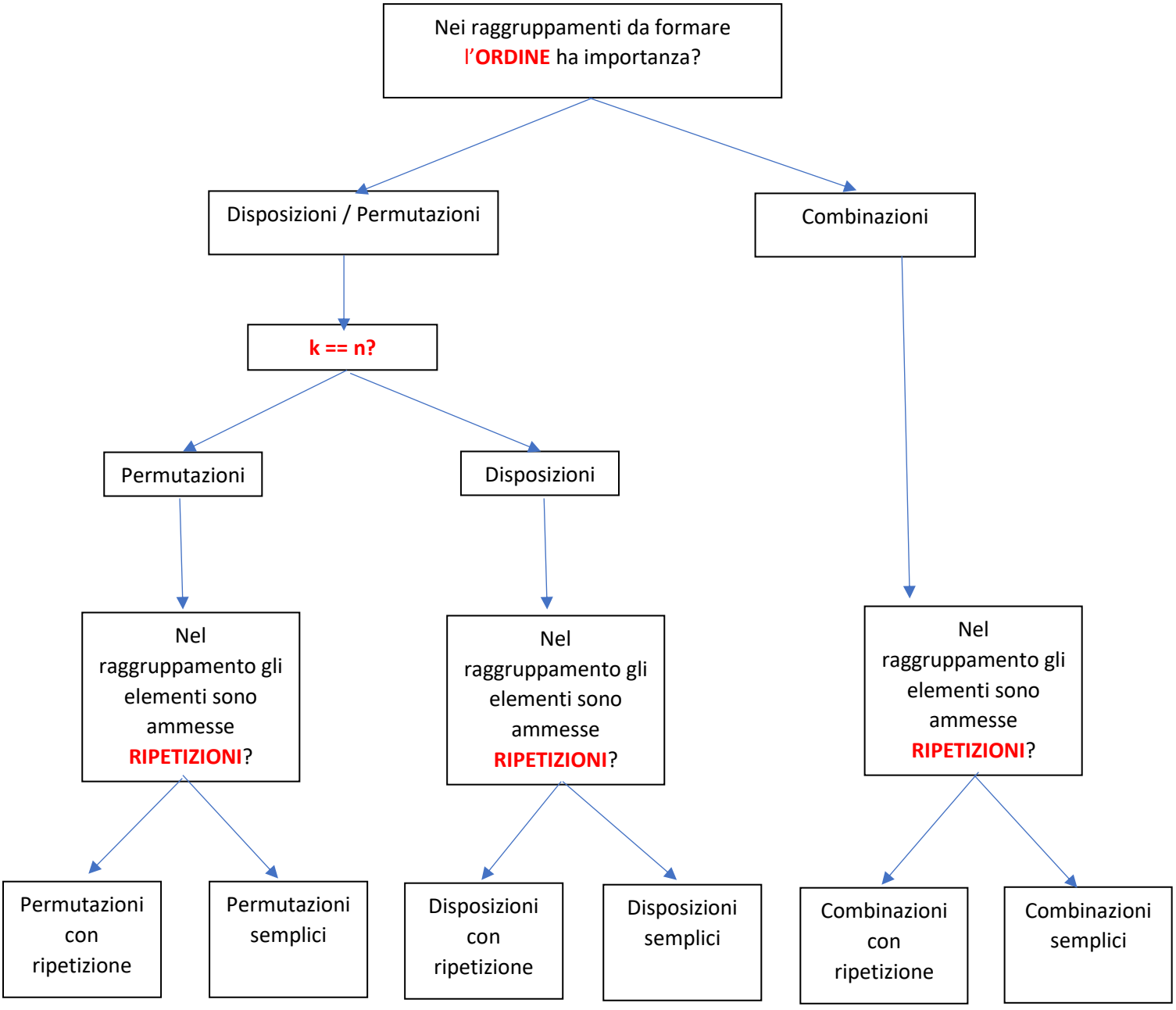
Prospetto Riassuntivo sul Calcolo combinatorio

n → cardinalità dell'insieme di partenza (source)

k → cardinalità del raggruppamento desiderato (target)

frecce sx → si

frecce dx→no



Definizioni e Formule

Il calcolo combinatorio affronta quanti raggruppamenti si possono ottenere da un dato numero n di oggetti collocati in un dato numero k di posti. Tali raggruppamenti possono essere formati senza ripetizione o con ripetizione degli n oggetti.

Raggruppamento detto **Permutazione**: si forma quando il numero degli oggetti è **uguale** al numero dei posti.

Raggruppamento detto **Disposizione**: si forma quando il numero degli oggetti è **diverso** dal numero dei posti e **l'ordine è importante**.

Raggruppamento detto **Combinazione**: si forma quando il numero degli oggetti è **diverso** dal numero dei posti e **l'ordine non è importante**.

Calcolo Combinatorio	
Permutazioni senza ripetizioni	$P_n = n!$
Permutazioni con ripetizioni	$P_n^r = \frac{n!}{r_1!\, r_2! \ldots r_k!}$
Disposizioni senza ripetizioni	$D_{n,k} = \frac{n!}{(n-k)!} \bigwedge n > k$
Disposizioni con ripetizioni	$D_{n,k}^r = n^k$
Combinazioni senza ripetizioni	$C_{n,k} = \frac{n!}{k!\,(n-k)!} \bigwedge n > k$
Combinazioni con ripetizioni	$C_{n,k}^r = \frac{(n+k-1)!}{k!\,(n-1)!}$

Template dei codici in C

Disposizioni semplici

Strutture dati / variabili di rilievo	Dimensione	Descrizione
val[]	n	**Insieme** degli elementi (source)
mark[]	n	Vettore per marcare gli elementi presi (1) / non presi (0)
sol[]	k (k<=n)	Vettore delle soluzioni, limitato superiormente da n
pos		Indice della discesa ricorsiva (fino a k, cioè la dimensione della soluzione target)

Per le disposizioni semplici la presenza del vettore mark permette di gestire il meccanismo del backtrack: un elemento può essere incluso nella soluzione oppure escluso. L'esclusione significa annullare la scelta dell'elemento. Il vettore mark traduce due vincoli:

1. In fase di costruzione del raggruppamento, una volta preso un elemento non è possibile prenderlo ancora, **le disposizioni semplici non ammettono ripetizioni**
2. Per le disposizioni semplici l'ordinamento è un ulteriore fattore di distinzione dei raggruppamenti: smarcare gli elementi permette di collocarli in altre posizioni della soluzione.

```
int disp_semplici(int *val, int *sol, int *mark, int pos, int n, int k, int count){
    int i;
    if(pos>=k){
        //stampa soluzione
        for(i=0; i<k; i++){
            printf("%d ", val[sol[i]]);
        }
        return count+1;
    }
    for(i=0; i<n ; i++){

        if(mark[i] == 0){

            mark[i] = 1; //marco la scelta fatta
            sol[pos] = i;
            count = disp_semplici(val, sol, mark, pos+1, n, k, count);
            mark[i] = 0; // annullo la scelta fatta
        }
    }
    return count;
}
```

Disposizioni con ripetizione

Strutture dati / variabili di rilievo	Dimensione	Descrizione
val[]	n	**Insieme** degli elementi (source)
~~mark[]~~	~~n~~	~~Vettore per marcare gli elementi presi (1) / non presi (0)~~
sol[]	k (k>0)	Vettore delle soluzioni, non limitato superiormente
pos		Indice della discesa ricorsiva (fino a k, cioè la dimensione della soluzione target)

Nelle disposizioni con ripetizione la dimensione k dei raggruppamenti **non è limitata** dalla cardinalità dell'insieme di partenza. Sono ammesse le ripetizioni e lo stesso elemento può essere preso fino a k volte.

Per questo motivo **non ci serve il vettore mark**

```
int disp_ripetute(int *val, int *sol, int pos, int n, int k, int count){
    int i;
    if(pos>=k){
        //stampa soluzione
        for(i=0; i<k; i++){
            printf("%d ", val[sol[i]]);
        }
        return count+1;
    }
    for(i=0; i<n ; i++){
            sol[pos] = i;
            count = disp_ripetute(val, sol, pos+1, n, k, count);
    }
    return count;
}
```

Permutazioni semplici

Strutture dati / variabili di rilievo	Dimensione	Descrizione
val[]	n	**Insieme** degli elementi (source)
mark[]	n	Vettore per marcare gli elementi presi (1) / non presi (0)
sol[]	n	Vettore delle soluzioni, limitato superiormente da n
pos		Indice della discesa ricorsiva (fino a n, cioè la dimensione della soluzione target)

Come le disposizioni semplici ma la dimensione della soluzione è pari alla dimensione dell'insieme di partenza. Dal punto di vista del codice cambia solo la condizione di terminazione.

```
int perm_semplici(int *val, int *sol, int *mark, int pos, int n, int count){
    int i;
    if(pos>=n){
        //stampa soluzione
        for(i=0; i<n; i++){
            printf("%d ", val[sol[i]]);
        }
        return count+1;
    }
    for(i=0; i<n ; i++){

        if(mark[i] == 0){

            mark[i] = 1; //marco la scelta fatta
            sol[pos] = i;
            count = perm_semplici(val, sol, mark, pos+1, n, count);
            mark[i] = 0; // annullo la scelta fatta
        }
    }
    return count;
}
```

Permutazioni con ripetizione

Strutture dati / variabili di rilievo	Dimensione	Descrizione
val[]	n	**Multinsieme** degli elementi (source). Lo stesso elemento può comparire più volte
dist_val[]	n_dist	**Insieme** ottenuto a partire dal multinsieme considerando solo le istanze distinte
mark[]	n_dist	Vettore per marcare le occorrenze disponibili degli elementi in dist_val[]
sol[]	n	Vettore delle soluzioni, limitato superiormente da n
pos		Indice della discesa ricorsiva (fino a n, cioè la dimensione della soluzione target)

Per affrontare il problema, il multiset è disaccoppiato in 2 informazioni:

- dist_val[] → vettore degli elementi distinti del multiset
- mark[] → vettore delle occorrenze di ciascun elemento in dist_val[]. Registra quante volte un elemento compare nel multiset

Siamo nel dominio delle permutazioni: i raggruppamenti hanno la stessa dimensione del vettore originale, ovvero il multiset. Non utilizziamo la variabile k nella condizione di terminazione

Non useremo nel codice il vettore val[] avendo disaccoppiato le informazioni, useremo invece dist_val[].

```
int perm_ripetute(int *dist_val, int *sol, int *mark, int pos, int n_dist,int n, int count){
    int i;

    if(pos>=n){
        //stampa soluzione
        for(i=0; i<n; i++){
            printf("%d ", dist_val[sol[i]]);
        }
        return count+1;
    }

    for(i=0; i<n_dist ; i++){
        if(mark[i] > 0){
            mark[i]--; //prendo l'oggetto, riduco occorrenza
            sol[pos] = i;
            count = perm_ripetute(dist_val, sol, mark, pos+1, n_dist, n, count);
            mark[i]++; //annullo la scelta dell'oggetto e ripristino l'occorrenza
        }
    }

    return count;
}
```

Combinazioni semplici

Strutture dati / variabili di rilievo	Dimensione	Descrizione
val[]	n	**Insieme** degli elementi (source)
start		Indice che determina a partire da quale valore del vettore val si inizia a riempire il vettore sol
sol[]	k (k<=n)	Vettore delle soluzioni, **limitato superiormente da n**
pos		Indice della discesa ricorsiva (fino a n, cioè la dimensione della soluzione target)

Nel dominio delle combinazioni in cui l'ordine non conta usiamo l'indice **start** per forzare uno dei possibili ordinamenti, impedendo ulteriori permutazioni. Ricordiamo infatti che nell'ambito delle combinazioni le permutazioni degli stessi elementi sono tenute in conto una volta sola.

Il codice è stato ottenuto modificando opportunamente quello delle disposizioni ripetute.

Si noti che nella chiamata ricorsiva si ricorre sulla **scelta corrente incrementata di 1 (i+1)** → così forziamo un ordinamento

```
int comb_semplici(int *val, int *sol, int pos, int n, int k, int count, int start){
    int i;
    if(pos>=k){
        //stampa soluzione
        for(i=0; i<k; i++){
            printf("%d ", val[sol[i]]);
        }
        return count+1;
    }
    for(i=start; i<n ; i++){
        sol[pos] = i;
        count = comb_semplici(val, sol, pos+1, n, k, count, i+1);
    }
    return count;
}
```

Combinazioni con ripetizione

Strutture dati / variabili di rilievo	Dimensione	Descrizione
val[]	n	**Insieme** degli elementi (source)
start		Indice che determina a partire da quale valore del vettore val si inizia a riempire il vettore sol
sol[]	k (k>0)	Vettore delle soluzioni, **non limitato superiormente**
pos		Indice della discesa ricorsiva (fino a n, cioè la dimensione della soluzione target)

L'approccio è analogo a quello delle combinazioni semplici, le ripetizioni nel raggruppamento vengono gestite tramite **start**, che viene incrementato solo al ritorno dalla chiamata ricorsiva.

In fase di chiamata ricorsiva, anziché inviare (i+1) mandiamo **start** proprio per permettere le configurazioni in cui lo stesso elemento è ripetuto.

Il codice di seguito è un riadattamento di quello impostato per le combinazioni semplici.

```
int comb_ripetute(int *val, int *sol, int pos, int n, int k, int count, int start){
    int i;
    if(pos>=k){
        //stampa soluzione
        for(i=0; i<k; i++){
            printf("%d ", val[sol[i]]);
        }
        return count+1;
    }
    for(i=start; i<n ; i++){
        sol[pos] = i;
        count = comb_ripetute(val, sol, pos+1, n, k, count, start);
        start++;
    }
    return count;
}
```