Commento all'es. 1

Si tratta di identificare tra tutti i possibili sottoinsiemi di frecce da cambiare di verso il primo che soddisfa entrambe le seguenti condizioni:

- tutte le frecce partecipano ad uno scontro e tutti gli scontri sono equilibrati
- la cardinalità del sottoinsieme è minima.

Si tratta di un problema di powerset, che potrebbe essere realizzato:

- con le disposizioni ripetute. Vantaggio: viene automaticamente generato l'insieme vuoto (è infatti possibile che non sia necessario cambiare di verso nessuna freccia). Svantaggio: per determinare il sottoinsieme di cardinalità minima si devono generare tutti i sottoinsiemi
- con le combinazioni semplici. Svantaggio: bisogna includere esplicitamente il caso del sottoinsieme vuoto. Vantaggio: facendo crescere `k` ci si ferma automaticamente all'insieme di cardinalità minima che soddisfa la prima condizione.

La soluzione proposta si basa su powerset calcolato come disposizioni ripetute. Si osservi che è possibile anticipare il controllo di ottimalità, subordinando ad esso la discesa ricorsiva nel caso si decida di cambiare il verso di una freccia (istruzioni commentate alle righe 71 e 74). La funzione di verifica determina il numero di frecce consecutive nello stesso verso e, quando il verso cambia, nel verso opposto. Nel momento in cui avviene un altro cambio di verso, verifica se lo scontro precedente era equilibrato e, nel caso non lo fosse, interrompe anticipatamente la verifica.

Commento all'es. 2

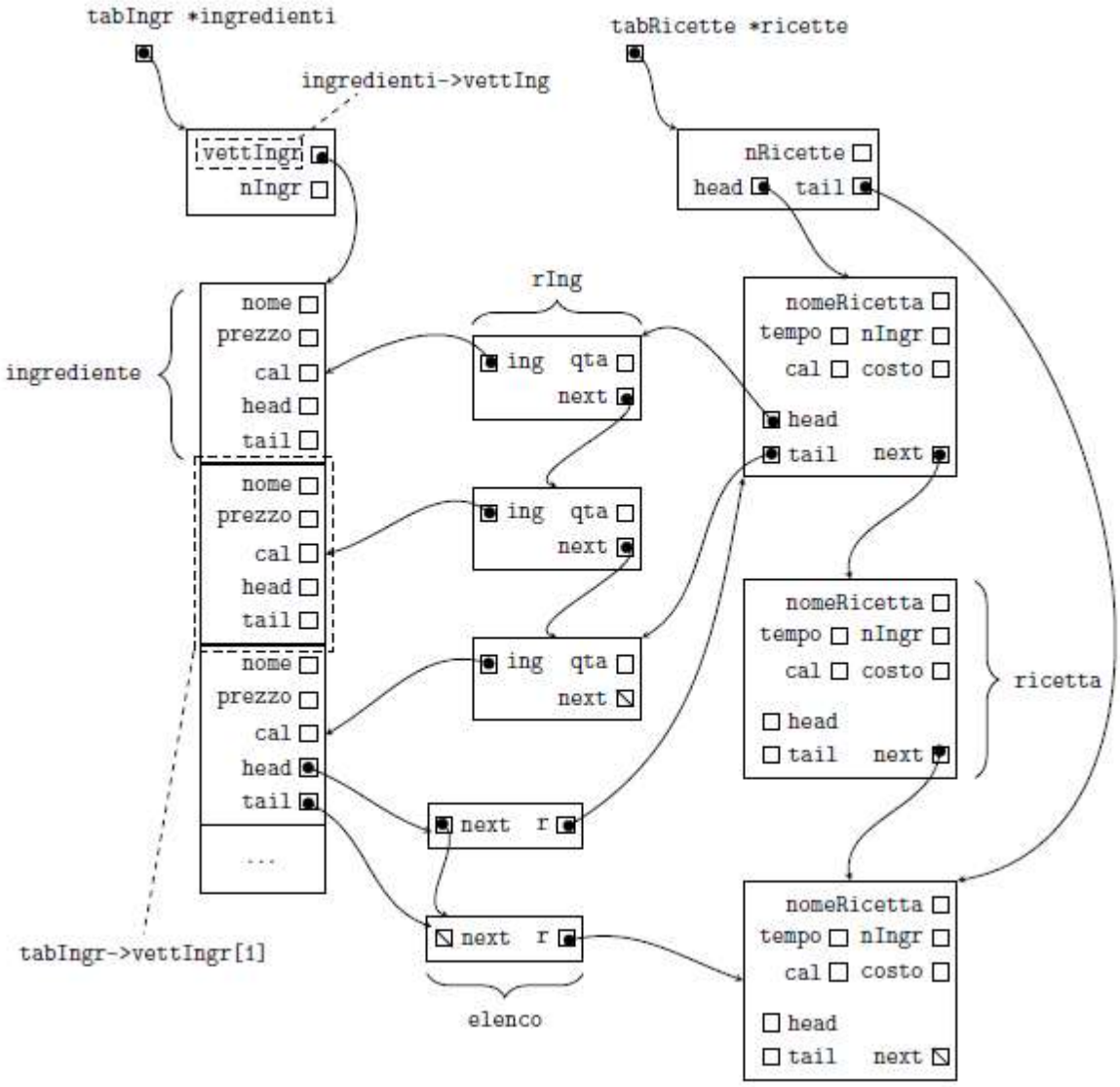
Ingredienti:

- la struttura dati wrapper `tabIngr` contiene il numero di ingredienti e un vettore `vettIng`, le cui celle sono di tipo `ingrediente`
- il tipo `ingrediente` è una struttura con campi nome, calorie, prezzo, nonché 2 puntatori a testa e coda di una lista di nodi di tipo `elenco`. Questa lista serve a memorizzare le ricette in cui compare quell'ingrediente
- ogni nodo di tipo `elenco` contiene, oltre al puntatore al prossimo nodo, anche un puntatore ad una ricetta, cioè ad un nodo che fa parte della lista delle ricette

Ricette:

- la struttura dati wrapper `tabRicette` contiene il numero di ricette e 2 puntatori a testa e coda di una lista di nodi di tipo `ricetta`
- ogni nodo di tipo `ricetta` contiene, oltre al puntatore al prossimo nodo, i campi per il nome, il numero di ingrediente, le calorie, il tempo e il costo, nonché due puntatori a testa e coda di una lista di nodi di tipo `rIng`. Questa lista serve a memorizzare gli ingredienti che compaiono in quella ricetta
- ogni nodo di tipo `rIng` contiene, oltre al puntatore al prossimo nodo, anche un puntatore ad un ingrediente, cioè ad una cella che fa parte del vettore degli ingredienti.

Si osservi che nella struttura dati `tabIngr` non vi sono duplicazioni delle ricette e che nella struttura dati `tabRicette` non vi sono duplicazioni degli ingredienti, bensì solo puntatori. Questo permette di mantenere facilmente la congruenza dei dati nel caso, non richiesto dall'esercizio, di modifica del nome di una ricetta o di un ingrediente.
La figura seguente illustra la struttura dati.



Funzioni: il main legge prima gli ingredienti e poi le ricette. La lettura degli ingredienti non presenta particolarità da commentare. Quando invece il main legge una ricetta (`leggiRicetta`), si osservi che, per ogni ingrediente che vi compare, si identifica tramite una ricerca con `cercaIngrediente` il puntatore alla cella del vettore `vettIng` in `tabIngr` che lo contiene e si aggiorna la lista degli ingredienti per tale ricetta. In seguito si crea un nuovo nodo di tipo `elenco` e lo si inserisce opportunamente nella lista delle ricette di quell'ingrediente.
Le funzioni di `cercaIngrediente`, `stampaIngredienti`, `stampaRicetteIngrediente`, `cercaRicetta`, `stampaRicette`, `stampaRicetta` sono semplici scansioni lineari di liste o di vettori. La funzione `nuovaRicetta` legge una ricetta e la inserisce opportunamente in lista in coda.