# *Commento all'es. 1*

**Dati:**

- la soluzione proposta non definisce un tipo **`Item`**, vista la semplicità dei dati e il numero ridotto di funzioni che vi opererebbero(lettura)
- **ADT di I classe `ST`:** tabella di simboli implementata come tabella di hash con open addressing e linera probing. Non si ci preoccupa del dimensionamento della tabella: la sua dimensione per semplicità coincide con il numero di vertici. La tabella serve per memorizzare il nome dell'elaboratore e l'indice intero che lo contraddistingue, permettendo di recuperare l'indice dato il nome. Un vettore `net` di nodi di tipo `node_t` (`struct` con campi stringa per nome dell'elaboratore e nome della sottorete) dato l'indice permette l'accesso al nome
- **ADT di I classe `Graph`:** è un grafo non orientato e non pesato, con doppia memorizzazione come lista delle adiacenze e come matrice delle adiacenze. Non si propone un tipo per l'arco, che viene gestito direttamente nel grafo. La funzione `GRAPHlist2mat` genera la matrice a partire dalla lista delle adiacenze. Le altre funzioni operano sia sulla rappresentazione a lista, sia su quella a matrice con selezione basata su flag. L'ADT grafo esporta una funzione di inserzione di arco (`GRAPHinsertE`), di calcolo del grado di un nodo di indice dato (`GRAPHdegree`) e di identificazione dei nodi e relativi pesi adiacenti a un nodo di indice dato (`GRAPHadjacent`. Essendo la tabella di simboli esterna al grafo, è compito del main fornire al grafo le informazioni sui vertici come interi ed interpretare gli interi ritornati dalle funzioni del grafo come nomi.

**Algoritmo:** il main inizializza le strutture dati (grafo, tabella di simboli e vettore di corrispondenza) e offre all'utente un menu di operazioni. Dalla tabella di simboli estrae le informazioni da passare alle funzioni del grafo e da queste ritorna le informazioni testuali.

# *Commento all'es. 2*

**Dati:**

- **ADT `Item`:** serve a memorizzare i dati relativi a ciascuna sala. Su di esso si definiscono le funzioni richieste dalla specifica, funzioni ausiliarie per il confronto tra item (`ITEMgreater, ITEMless,`), nonché il tipo chiave `Key` e le funzioni per estrarre la chiave (`KEYget`) e confrontarla (`KEYcompare`)
- **ADT di I classe `PQ`:** coda a priorità di item implementata come heap. Il wrapper contiene il vettore, una variabile intera per l'heapsize e una variabile intera per la dimensione massima allocata del vettore. La funzione di inserzione (`PQinsert`) provvede a una `realloc` con raddoppio della dimensione quando si raggiunge la condizione di vettore pieno. La funzione di `PQget` è equivalente alla funzione di `PQExtractMax`. Le funzioni sulla coda a priorità sono quelle viste a lezione

# *Commento all'es. 3*

Nella prima versione si determina innanzitutto il numero massimo di sottostringhe `nstr` in cui potrà essere suddivisa la stringa di partenza. Esso si ottiene come risultato della divisione intera della lunghezza della stringa di partenza per la lunghezza della più corta delle stringhe in cui si

suddivide, determinata in fase di lettura del vettore `lunghezze`. Visto che trattandosi di stringhe l'ordine conta, il modello adottato è quello delle disposizioni ripetute di `n` elementi a `k` a `k`, dove `k` è il numero di sottostringhe utilizzate, compreso tra 1 e `nstr`. In questo modo, se fosse richiesto, si potrebbe facilmente determinare la decomposizione con numero minimo di sottostringhe. Si introduce una semplice forma di pruning del tipo `pos + lunghezze[i] <= lungh` per evitare di considerare un caso certamente inaccettabile, in quanto si oltrepasserebbe la lunghezza della stringa. La soluzione viene memorizzata in un vettore di interi `sol` di dimensione `k` allocato/liberato dinamicamente dal main ad ogni iterazione su `k`. Esso contiene le lunghezze delle sottostringhe usate. La funzione di accettazione nel caso terminale verifica che la somma delle lunghezze delle sottostringhe usate sia pari a quella della stringa di partenza.
La seconda versione differisce dalla prima nell'input semplificato, nel numero massimo di sottostringhe `nstr` in cui potrà essere suddivisa la stringa di partenza sovradimensionato a `N` e nel fatto che si usa una variante delle disposizioni ripetute in cui ad ogni ricorsione si opera su quanto rimane della stringa dopo aver deciso con quale sottostringa decomporla.