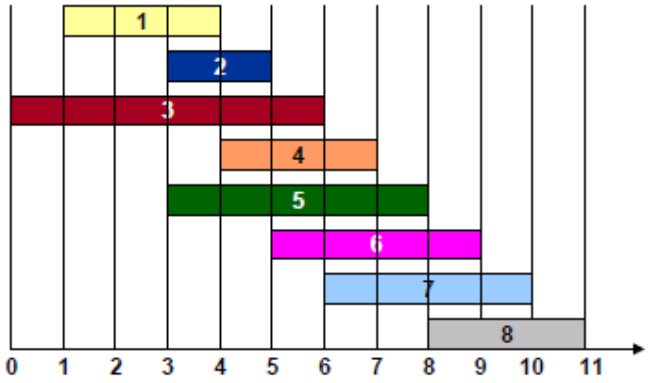
Commento al Laboratorio n. 9

Esercizio n. 1: **Sequenza di attività (versione 2)**

Le `n` attività sono identificate da un indice `i` che inizia da 1. L'indice 0 indica l'attività 0 che è fittizia. Le attività sono memorizzate in un vettore di attività `v` (vedi es. 1 del Lab. 8) di `n`+1 celle.

Il criterio di ordinamento è il tempo di fine crescente do ogni attività, conformemente a quanto fatto per la versione vista a lezione risolta con il paradigma greedy per il problema che mira a massimizzare il numero di attività (non la durata complessiva). Si usa il `MergeSort` per ordinare il vettore. Con riferimento al file di esempio `att1.txt`, il risultato di questo passo è riportato nella figura seguente:



Come secondo passo si identifica per ogni attività `i` quale è l'indice dell'attività che termina più tardi e che è compatibile con `i` e lo si memorizza in un vettore `q` di `n`+1 celle, dove `q[0]`=0 si riferisce all'attività fittizia 0. Per l'esempio di cui sopra il contenuto di `q` è:

0	0	0	0	1	0	2	3	5
0	1	2	3	4	5	6	7	8

Passo 1: applicabilità

Si ipotizzi che `opt[i]` sia la soluzione ottima al problema in cui si considerano le attività da 1 a `i`:

- si ipotizzi che l'attività `i` faccia parte della soluzione. Il sottoproblema da risolvere è quello `q(i)`-esimo che prende in considerazione le attività da 1 a `q(i)`, quindi certamente compatibili con l'attività `i`-esima. Se la soluzione al problema `q(i)`-esimo non fosse ottima, se ne troverebbe una con valore `opt'[q(i)]` maggiore, che, sommato alla durata dell'attività `i`, porterebbe ad un valore `opt'[i]` > `opt[i]` contraddicendo l'ipotesi di `opt[i]` ottimo
- se l'attività `i` non fa parte della soluzione, `opt[i]=opt[i-1]` e se `opt[i-1]` non fosse ottimo non lo potrebbe neanche essere `opt[i]`.

Passo 2: soluzione ricorsiva divide et impera

L'analisi precedente può essere riassunta con la seguente formulazione ricorsiva:

$$opt(i) = \begin{cases} 0 & i = 0 \\ max(opt(i-1), f_i - s_i + opt(q(i))) & 1 \leq i \leq n \end{cases}$$

Passo 3: soluzione con programmazione dinamica bottom-up (calcolo del valore della soluzione ottima)
Ispirandosi alla formulazione ricorsiva della soluzione, la si trasforma in forma iterativa:

- `opt[0]` è noto a priori,
- per $1 \leq i \leq n$ `opt[i] = max(opt[i-1], attDurata(v[i] + opt[q[i]]).`

Il valore della soluzione ottima è memorizzato in `opt[n]`.

Passo 4: costruzione della soluzione ottima
La funzione `displaySol` costruisce ricorsivamente e visualizza la soluzione ritracciando all'indietro (partendo da `pos=n`) quanto fatto per determinare il valore ottimo. La condizione di terminazione si raggiunge per `pos=0` e, trattandosi dell'attività fittizia, non si fa nulla, ma si ritorna.
Se è vera la condizione `attDurata(v[pos])+opt[q[pos]] >= opt[pos-1]` si ricorre per `pos=q[pos]` e al termine si stampa l'attività `v[pos]`, altrimenti si ricorre su `pos`-1.

Esercizio n. 2: Gioco di ruolo (multi-file, con ADT)
La soluzione all'esercizio segue direttamente le strategie proposte per la realizzazione degli ADT, tenendo presente le linee-guida date nelle specifiche.