

# Esercitazione 8: More on Dynamic Programming

Giacomo Paesani

May 16, 2024

**Esercizio 1** (24.2-3, [1]). Dato un grafo orientato  $G = (V, E)$  e con archi pesati da una funzione  $w$ , senza cicli orientati di peso negativo, e sia  $m$  il massimo del numero minimo di archi di un cammino minimo da un vertice  $s$  a  $v$ , per ogni vertice  $v \in V$ . Fornire uno pseudo-codice modificando l'algoritmo di *Bellman-Ford* in modo che vengono fatte al più  $m + 1$  iterazioni, anche se  $m$  non è noto a priori.

**Esercizio 2** (16.2-2, [1]). Il problema dello *zaino* (knapsack problem) è un classico problema di ottimizzazione definito nella seguente maniera. Data una collezione di oggetti  $A = \{1, \dots, n\}$ , ad ogni oggetto  $i$ -esimo vengono associati numeri interi  $v_i$  e  $w_i$  che rappresentano il valore e il peso, rispettivamente. Allora dato un intero  $C$ , il problema richiede un insieme  $U \subseteq A$  tale che  $\sum_{a_i \in U} w_i \leq C$  e  $\sum_{a_i \in U} v_i$  è massima, cioè l'obiettivo è di selezionare il sottoinsieme di oggetti che ha un peso totale che non supera  $C$  e che massimizza il valore complessivo.

Fornire lo pseudo-codice di un algoritmo che risolve il problema dello zaino in tempo  $\mathcal{O}(n \cdot C)$ . Usare un approccio di programmazione dinamica. Come devo modificare l'algoritmo per avere anche gli elementi che compongono una soluzione?

**Esercizio 3.** Sia  $X = (X[1], \dots, X[n])$  una stringa di  $n$  elementi e  $Y = (Y[1], \dots, Y[m])$  una stringa di  $m$  elementi. Definiamo  $d(X, Y)$  come il minimo numero di operazioni necessarie per convertire  $X$  in  $Y$ , dove le operazioni permesse sono le seguenti:

- *inserire*( $k, z$ ) il simbolo  $z$  alla posizione  $k$  (e far slittare tutti i successivi di una posizione a destra) in  $X$ ;

- *rimuovere*( $k$ ) il simbolo in posizione  $k$  (e far slittare tutti i successivi di una posizione a sinistra) in  $X$ ;
- *sostituire*( $k, z$ ) assegnare al simbolo in posizione  $k$  il valore  $z$  in  $X$ .

Ad esempio, se  $X = (C, O, L, T)$  e  $Y = (C, I, T)$ , allora  $d(X, Y) = 2$  e prodotta, ad esempio, da *sostituire*(2,  $i$ ) e *rimuovere*(3). Rispondere alle seguenti richieste:

1. dimostrare che la funzione  $d$  è una distanza, cioè che  $d(X, X) = 0$ ,  $d(X, Y) = d(Y, X)$  e  $d(X, Y) \leq d(X, Z) + d(Z, Y)$ ;
2. fornire in pseudo-codice un algoritmo che, date due stringhe  $X$  e  $Y$  di lunghezza  $n$  e  $m$  rispettivamente, calcoli  $d(X, Y)$  ed esibisca le operazioni eseguite, in tempo  $\mathcal{O}(n \cdot m)$ .

**Esercizio 4.** Sia  $X = (x_1, \dots, x_n)$  una stringa di  $n$  elementi. Definiamo  $d_p(X)$  come il minimo numero di cancellazioni di sotto-stringhe palindrome di  $X$  per ottenere la stringa vuota. Ad esempio, se  $X = (1, 3, 3, 4, 3, 5, 2, 7, 1, 7, 2, 3)$  allora  $d_p(X) = 4$  rimuovendo le seguenti stringhe palindrome  $(3, 4, 3)$ ,  $(2, 7, 1, 7, 2)$ ,  $(3, 5, 3)$  e infine  $(1)$ . Fornire in pseudo-codice un algoritmo che, data una stringa  $X$ , calcoli  $d_p(X)$  ed esibisca le operazioni eseguite, in tempo  $\mathcal{O}(n^3)$ .

## References

- [1] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. Introduction to algorithms. 2022.