Esercitazione 6: More on Greedy Algorithms and Dynamic Programming

Giacomo Paesani

April 24, 2024

Esercizio 1 (23.1-11, [1])**.** Sia $G = (V, E)$ un grafo non diretto e connesso con gli archi pesati da una funzione w e T un albero di copertura di G di peso minimo. Supponiamo che il peso di un'arco (u, v) che non appartiene a T diminuisce. Fornire un algoritmo in pseudo-codice che dato G, T e come viene modificato il peso dei un arco (u, v), trova un albero di copertura di peso minimo nel grafo modificato (senza calcolarlo da capo).

Esercizio 2 (23.2-7,[1])**.** Sia $G = (V, E)$ un grafo non diretto e connesso con gli archi pesati da una funzione w e T un albero di copertura di G di peso minimo. Supponiamo che viene aggiunto un nuovo vertice u e gli archi a se incidenti a G. Fornire un algoritmo in pseudo-codice che dato G, T e la lista di adiacenza di u nei vertici di G, trova un albero di copertura di peso minimo nel grafo $G \cup \{u\}$ (senza calcolarlo da capo).

Esercizio 3 (23.2-8, [1])**.** Sia $G = (V, E)$ un grafo non diretto, connesso e con pesi sugli archi dati da una funzione w. Consideriamo la seguente strategia ricorsiva per calcolare un albero di copertura di peso minimo di G. Partizioniamo l'insieme V in due sottoinsiemi V_1 e V_2 tali che $|V_1|$ e $|V_2|$ differiscono di al più uno. Sia E_1 l'insieme di archi che sono incidenti solo a vertici di V_1 e E_2 l'insieme di archi che sono incidenti solo a vertici di V_2. Ricorsivamente troviamo un albero di copertura di peso minimo per i due sottografi $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$. In fine selezionare l'arco di peso minimo che attraversa il taglio (V_1, V_2) e aggiungerlo ai due alberi di copertura di peso minimo per i sottografi G_1 e G_2.

Dimostrare che la strategia proposta permette di implementare un algoritmo che trova correttamente una soluzione o esibire un contro-esempio che mostra come la strategia non sempre produce soluzioni ottime.

Esercizio 4 (33.4-3:4, [1])**.** Possiamo definire la distanza tra due punti in modi diversi, non solo usando quella euclidea. Sul piano, la distanza L_m tra due punti $p_1 = (x_1, y_1)$ e $p_2 = (x_2, y_2)$ è data dalla seguente espressione $d_m(p_1, p_2) = (|x_1 - x_2|^m + |y_1 - y_2|^m)^{1/m}$. Quindi, la distanza euclidea è esattamente la distanza L_2. Inoltre la distanza L_∞ è definita nella seguente maniera: $d_\infty(p_1, p_2) = \max(|x_1 - x_2|, |y_1 - y_2|)$

Modificare l'algoritmo per trovare la coppia di punti più vicini nel piano usando le distanze L_1 e L_∞.

References

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