Domande di Teoria - Analisi e Progetto di Algoritmi

Fabio Ferrario

2022

Indice

Capitolo 1

Domande di Teoria

1.1 Domande date da Dennunzio

Queste domande vengono da un file che ha distribuito il Professor Dennunzio.

1 Siano $X =< x_1, ..., x_m >$ e $Y =< y_1, ..., y_n >$ due sequenze e sia $Z =< z_1, ..., z_k >$ una LCS di X e Y. Scrivere la proprieta' di sottostruttura ottima di Z

Siano $X = \{x_1, x_2, ..., x_m\}$ e $Y = \{y_1, y_2, ..., y_n\}$ le sequenze; sia $Z = \{z_1, z_2, ..., z_m\}$ una qualsiasi LCS di X e Y.

1. Se $x_m = y_n$, allora $z_k = x_m = y_n$ e Z_{k-1} è una LCS di X_{m-1} e Y_{n-1}
2. Se $x_m \neq y_n$, allora $z_k \neq x_m$ implica che Z è una LCS di X_{m-1} e Y
3. Se $x_m \neq y_n$, allora $z_k \neq y_n$ implica che Z è una LCS di X e Y_{n-1}

2 Scrivere le equazioni di ricorrenza per risolvere il problema della LCS di due sequenze, specificando bene il significato delle variabili coinvolte

$$c[i,j] = \begin{cases} 0 & \text{se } i = 0 \text{ or } j = 0 \\ c[i-1, j-1] + 1 & \text{se } i, j > 0 \text{ and } x_i = y_j \\ max(c[i, j-1], c[i-1, j]) & \text{se } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

$c[i, j]$ è una matrice contenente la lunghezza di una qualsiasi LCS delle sottosequenze di lunghezza rispettivamente i e j di X e Y

3 Scrivere l'algoritmo che determina la lunghezza della LCS di due sequenze specificando il suo tempo di calcolo

Tempo di esecuzione di LCS-LENGHT: $\theta(mn)$ perchè il calcolo di ogni posizione della tabella richiede 1.

```
1  LCS-LENGHT(X,Y)
2  m = X.lenght
3  n = Y.lenght
4  Siano b[1..m,1..n] e c[0..m,0..m] due nuove tabelle
5  for i = 1 to m
6      c[i,0] = 0
7  for j= 0 to n
8      c[0,j] = 0
9  for i = 1 to m
10     for j = 1 to n
11         if x_i == y_i
12             c[i,j] = c[i-1,j-1] + 1
13             b[i,j] = ↖
14         elseif c[i - 1,j] ≥ c[i,j-1]
15             c[i,j] = c[i-1,j]
16             b[i,j] = ↑
17         else
18             c[i,j] = c[i,j-1]
19             b[i,j] = ←
20 return c e b
```

4 Definire qual è il sottografo dei predecessori (o albero BFS) prodotto dalla visita BFS di un grafo $G =< V, E >$, specificando bene da quali vertici e quali archi e' composto.

Il sottografo dei predecessori $G_\pi < V_\pi, E_\pi >$ è un albero BF (o albero di visita in ampiezza) se: V_π è formato da vertici raggiungibili da s e, per ogni $v \in V_\pi$, c'è un solo cammino semplice da s a v in G_π che è anche un cammino minimo da s a v in G. GLi archi in E_π sono detti *archi dell'albero*.

5 Scrivere qual e' il tempo di calcolo dell'algoritmo BFS motivando BENE la risposta (fare riferimento allo pseudocodice).

Il tempo totale di esecuzione di BFS è $O(V + E)$.
Di cui $O(V)$ è il tempo delle operazioni con la coda, e $O(E)$ è il tempo per l'ispezione di ADJ.

6 Spiegare il significato dei colori assegnati ai vertici dall'algoritmo BFS. Alla fine dell'esecuzione dell'algoritmo BFS su un grafo $G =< V, E >$, quali colori assumono i vertici?

BFS assegna ad ogni vertice del grafo un colore:
Bianco per i vertici non ancora scoperti, *Grigio* per i vertici scoperti ma di cui ancora non è stata scandita del tutto la lista di adiacenza e *Nero* ad ogni vertice scoperto di cui è stata scandita l'intera lista di adiacenza.
Alla fine dell'esecuzione di BFS, ogni nodo appartenente alla componente connessa del nodo sorgente assumera il colore *nero*

7 Definire qual e' il sottografo dei predecessori (o foresta DF) prodotto dalla visita DFS di un grafo $G =< V, E >$, specificando bene da quali vertici e quali archi e' composto.

Il sottografo dei predecessori o foresta DF è così definita: $G\pi = (V, E\pi)$, dove: $E\pi = (\pi[v], v) : v \in V \wedge \pi[v] \neq NIL$
In pratica, G_π è formato da tutti i vertici di G e tutti gli archi che vanno dal "padre" di un nodo v a v.

8 Descrivere la classificazione degli archi che la visita in profondità produce a partire da un grafo $G =< V, E >$. Come si classifica un generico arco (u, v) del grafo?

DFS classifica gli alberi in questo modo: Arco dell'albero: ogni arco che va a scoprire un nuovo vertice, Arco all'indietro: se va in un nodo grigio, Arco in avanti: se va in un nodo nero scoperto dopo u, Arco di attraversamento: se va in un nodo nero scoperto prima di u.

9 Scrivere qual e' il tempo di calcolo dell'algoritmo DFS motivando bene la risposta (fare riferimento allo pseudocodice)

DFS ha un tempo di esecuzione di $\theta(V + E)$. DFS-VISIT è chiamata una volta per ogni vertice (quando è bianco, quindi $\theta(V)$) e il ciclo in DFS-VISIT è chiamato una volta per ogni arco (ogni volta che c'è una adiacenza, quindi $\theta(E)$)

11 Dare la definizione di ordinamento topologico, specificando bene a che tipo di grafo si applica. Descrivere come si ottiene l'ordinamento topologico sfruttando l'algoritmo DFS

Un ordinamento topologico di un DAG G è un ordinamento lineare di tutti i suoi vertici tale che, se G contiene un arco (u,v), allora u appare prima di v nell'ordinamento. l'algoritmo TOPOLOGICAL-SORT(G) per ottenere un ordinamento topologico chiama DFS per calcolare i tempi di completamento $v.f$ e poi completata l'ispezione inserisce il vertice in una lista concatenata che poi ritorna.

12 Descrivere la caratterizzazione della struttura di un cammino minimo $p =< v1, \ldots, vl >$ utilizzata dall'algoritmo di Floyd-Warshall.

L'algoritmo considera i vertici "intermedi" di un cammino minimo, dove un vertice intermedio di un cammino semplice $p =< v_1, .., v_l >$ è un vertice qualsiasi di p diverso da v_1 e v_l ovvero un vertice qualsiasi dell'insieme $\{v_2, ..., v_{l-1}\}$.

13 Illustrare e motivare le equazioni di ricorrenza su cui si basa l'algoritmo di Floyd-Warshall, specificando bene il significato delle variabili coinvolte.

Equazione di ricorrenza:

$$d_{ij}^k = \begin{cases} w_{ij} & k = 0 \\ min\{d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\} & k > 0 \end{cases}$$

in cui: w_ij è il peso dell'arco da i a j, d_{ij}^k è il peso del cammino minimo da i a j usando al più i vertici di indice k. d_{ij}^k assume il valore del peso dell'arco diretto tra i e j se non ci sono archi intermedi, altrimenti il minore tra il cammino minimo usando un vertice intermedio in meno e la somma di due cammini minimi usando un vertice intermedio in meno.

14 illustrare un metodo per costruire i cammini minimi nell'algoritmo di Floyd-Warshall.

Per costruire i cammini minimi in FW calcolo la matrice D dei pesi dei cammini minimi e poi costruisco la matrice dei predecessori della matrice D.

15 Che cos'è la **chiusura transitiva** di un grafo orientato? Descrivere un modo per calcolarla.

La *chiusura transitiva* di un grafo orientato è un grafo che ha un arco da un nodo i a j se nel grafo originale esiste un cammino tra i due.
Più formalmente: dato un grafo orientato si vuole determinare se per ciascuna coppia di vertici i, j esiste un cammino da i a j. La chiusura transitiva è definita come il grafo $G* = (V, E*)$ dove $E* = \{(i, j)$: esiste un cammino dal vertice i al vertice j in $G\}$

Un modo per calcolare la CT di un grafo consiste nell'assegnare un peso 1 a ogni arco di E e nell'eseguire FW. se esiste un cammino da i a j si ha d_{ij} ¡ n, altrimenti infinito.

16 Descrivere come modificare le equazioni di ricorrenza dell'algoritmo di Floyd-Warshall per calcolare la chiusura transitiva di un grafo orientato.

La variabile t_{ij}^k assume valore 1 se esiste un cammino da i a j utilizzando al più vertici intermedi di indice k, 0 altrimenti (equivalgono a *True* e *False*)
Per $k = 0$

$$t_{ij}^0 = \begin{cases} 0 & i \neq j \wedge (i,j) \notin E \\ 1 & i = j \vee (i,j) \in E \end{cases}$$

$k > 0$

$$t_{ij}^k = t_{ij}^{k-1} \vee (t_{ik}^{k-1} \wedge t_{kj}^{k-1})$$

17 Dare la definizione di:

1. Sistema di Indipendenza.
2. Problema associato ad un coppia costituita da un sistema di indipendenza e da funzione peso definita sul sistema di indipendenza.

Scrivere inoltre qual è l'algoritmo Greedy associato a tale coppia.

1. Sistema di indipendenza: Data la coppia $< E, F >$ dove E è un insieme finito e F è una famiglia di sottoinsiemi di E, definiamo tale coppia sistema di indipendenza se vale la seguente proprietà:

$$\forall A \in F se B \subseteq A \implies B \in F$$

18 Definire cos'è un matroide, enunciare il Teorema di Rado.

Un *Matroide* è un sistema di independenza che rispetta la seguente proprietà:

$$\forall A, B \in F \text{ e } |A| = |B| + 1 \implies \exists a \in A \backslash B \text{ tale che } B \cup \{a\} \in F$$

il Teorema di Rado dice: Dato un sitema di indipendenza $< E, F >$ le seguenti proposizioni sono equivalenti:

- Per ogni funzione peso $w : E \to R+$, l'algoritmo greedy associato fornisce una soluzione ottima
- $< E, F >$ è un matroide

19 Cos'è una struttura dati per insiemi disgiunti? Definire formalmente quali sono le operazioni principali su una struttura dati per insiemi disgiunti.

Una struttura dati per insiemi disgiunti è un tipo di struttura dati che serve a mantenere una collezione di insiemi disgiunti in cui ogni insieme è identificato da un *rappresentante* (che è un elemento dell'insieme) e ogni elemento di un insieme è rappresentato da un oggetto x.
Le operazioni principali sono:

- *Make-set(x)*: Crea un nuovo insieme con x come unico elemento.
- *Find-set(x)*: Restituisce un puntatore al rappresentante del set in cui è contenuto x.
- *Union(x,y)*: Unisce gli insiemi di cui x e y fanno parte.

20 Illustrare i possibili modi con cui è possibile rappresentare una struttura dati per insiemi disgiunti e complessità relative

Una semplice rappresentazione di questi insiemi è data dalle *liste concatenate*. Il primo oggetto di ogni lista viene utilizzato come rappresentante. Ogni oggetto nella lista ha i seguenti campi:

- info: l'informazione contenuta nel nodo
- rappr: il puntatore al rappresentante
- succ: il puntatore al nodo successivo

21a Definire formalmente cos'è un albero di compertura minimo(MST).

Sia $G(V, E)$ un grafo *Pesato* tramite la funzione $w : E \to \mathbb{R}$, *Connesso* e *non orientato*.
Se $G_c(V, T)$ è un albero con $T \subseteq E$ ed è aciclico, allora G_c è un albero di copertura di G.
Un albero di copertura minimo (MST) di G è un Albero Aciclico $G_{MST}(V, T)$ con $T \subseteq E$ tale che:

$$w(T) = \min\{w(A) | A \subseteq E \wedge (V, A) \text{ albero di copertura di } G\}$$

21b Illustrare l'approccio greedy generico, fornendo la procedura GENERIC-MST, per determinare un MST illustrando il principio di funzionamento (cosa fa la proceduara ad ogni iterazione, proprietà invariante di ciclo, nozione di arco sicuro, ecc).

21c Come si trova un'arco sicuro?

21d Dare le definizioni di: taglio, arco che attraversa un taglio, taglio che rispetta un sottoinsieme di archi, arco leggero.

Sia $G(V, E)$ un grafo non orientato e connesso, valgono le seguenti definizioni:
Taglio:Si definisce *Taglio del Grafo G* una partizione $(S, V - S)$, dell'insieme dei vertici del grafo V.
Arco che attraversa un Taglio: Sia $(S, V - S)$ un taglio di G. Si dice che un arco $(u, v) \in E$ attrversa il taglio se: $u \in S \wedge v \in V - S \vee u \in V - S \wedge v \in S$
Taglio che rispetta un insieme di archi: Si dice che il taglio $(S, V - S)$ rispetta l'insieme A se nessun arco di A attraversa il taglio.

21e Enunciare il Teorema che da una condizione sufficiente affinchè un arco sia sicuro.

Sia $G(V, E)$ un grafo connesso, non orientato e pesato. Siano inolre:

1. $A \subseteq E$ un sottoinsieme di archi di un MST, che indichiamo con T, di G.
2. (S,V-S) un taglio definito su G che rispetta A;
3. $(u, v) \in E$ un arco del grafo di peso minimo che attraversa il taglio definito su G.

Allora (u, v) è un arco sicuro per A, cioè se ad A aggiungiamo anche l'arco (u,v) allora A continua a rappresentare un sottoinsieme di archi di un MST.

22 Illustrare gli algoritmi di Kruskal e Prim facendo riferimento alla procedura GENERICS-MST. Che differenze tra i due rispetto all'insieme A e a cosa è un arco sicuro. Complessità computazionale dei due algoritmi.

23 Complessità computazionale dell'algoritmo di Floyd-Warshall (giustificare)

L'algoritmo di Floyd-Warshall ha una complessità di $O(|V|^3)$. Questo algoritmo contiene tre cicli *for* di dimensione $|V|$, in cui confronta il cammino di ogni nodo con ogni altro nodo usando al più ogni "livello" di nodi intermedi.

1.2 Domande dagli Esami

Descrivere come usare la DFS per calcolare le componenti connesse di un grafo non orientato.

Per il funzionamento dell'algoritmo, ogni volta che *DFS* chiama *DFS-VISIT* su un nodo bianco, significa che ha trovato una nuova componente connessa. Se si volesse trovare il numero di CC di un grafo, basta mettere un contatore che viene incrementato ogni volta che *DFS* chiama *DFS-VISIT*

Descrivere come la visita BFS produce un'albero BFS dando descrizione dell'algoritmo e delle variabili che utilizza