Preparazione all'esame di Analisi e Progetto di Algoritmi

Fabio Ferrario

2022

Indice

Capitolo 1

Come si risolvono gli esercizi

1.1 Introduzione

All'esame di Analisi e Progetto di Algoritmi vengono (generalmente) proposti degli esercizi che ti richiedono di scrivere delle varianti di alcuni algoritmi. Per risolvere questi esercizi bisogna innanzitutto *sapere bene gli algoritmi di cui bisogna fare le varianti.* Questi generalmente sono:

- Algoritmo di Floyd-Warshall
- Algoritmi BFS e DFS
- Algoritmo LCS
- Raramente Algorimto di Dijkstra

Se conoscete bene questi algoritmi vi basterà conoscere i procedimenti che riporto qua sotto per poter risolvere (bene o male) ogni tipo di esercizio.

1.2 Varianti di FW

Un possibile tipo di esercizio d'esame è problema in cui is chiede di creare un algoritmo di programmazione dinamica che, dato un grafo pesato e una funzione che associa un numero o un colore a ogni arco o vertice, stabilisce il peso di/se seiste un cammino minimo da i a j che rispetti alcuni vincoli.

Il procedimento è sostanzialmente sempre lo stesso, bisogna trovare un sottoproblema k-esimo e capire se le informazioni date da i vertici (i, k) e (k, j) sono sufficienti a risolvere il problema. in caso contrario bisogna definire un problema ausiliario.

Variabili associate del sottoproblema (quindi quelle introdotto) sono sempre: il matricione D^k in cui d_{ij}^k è il peso/l'esistenza del cammino da i a j i cui vertici intermedi appartengono a $\{1, ..., k\}$

soluzione del sottoproblema Se le informazioni sono sufficienti, si formano le due equazioni di ricorrenza in cui si calcola d_{ij}^k, per il caso base e per il Passo Ricorsivo.
Il passo ricorsivo (k¿0) è sempre diviso in due casi ipotetici (sibilla cumana), il caso in cui k NON fa parte del cammino minimo, in cui il valore di d_{ij}^k diventerà il valore del cammino ij escludendo k come cammino intermedio, quindi k-1.
Oppure il caso in cui k fa parte del cammino minimo, in cui il valore di d_{ij}^k diventerà la somma (oppure l'AND) dei valori dei cammini ik e kj (tutti e due escludendo k come cammino intermedio)
L'equazione di ricorrenza diventerà quindi il minimo dei due casi (oppure un OR nel caso in cui si voglia verificare l'esistenza).
Il caso base invece è dove i calcoli veri vengono fatti. in genere il valore di d_{ij}^k ha tre possibilità: se i=j assumerà 0 o TRUE, perchè se i e j coincidono vuol dire che non ci sono archi per passare da uno all'altro. Se i e j non coincidono, il loro cammino è compreso in E e sono rispettate le condizioni del problema allora d_{ij}^k diventerà il peso del cammino (oppure TRUE). Diventa sempre ∞ o FALSE altrimenti.

Problema ausiliario Nel caso in cui non sia possibile calcolare il risultato con le informazioni date è necessario introdurre un Problema ausiliario. Questo problema generalmente è lo stesso iniziale ma aggiunge un controllo aggiuntivo, per esempio sul primo arco uscente e l'ultimo entrante del cammino. Una volta introdotto il problema ausiliario si procede come se fosse un problema normale, quindi si introduce il sottoproblema, le variabili e le equazioni di ricorrenza. Con le equazioni di ricorrenza però vengono aggiunte delle condizioni particolari che dipendono caso per caso (guarda esercizi).

Solzuone del problema La soluzone del problema nel caso in cui non ci sia un problema ausiliario è semplicemente formata da i valori nel matricione introdotto dal sottoproblema (D), se invece è presente un problema ausiliario, allora la soluzione diventa il minimo (o OR) di tutte le soluzioni del Problema ausiliario

1.3 Varianti DFS

Questo è un generico esercizio del tipo "conta i vertici ∀ Componente Connessa di un grafo"

Spiegazione

è una modifica di DFS in cui DFS-VISIT ritorna il numero di vertici che ha trovato nella componente connessa del nodo in esame. Quindi dopo ongi chiamata di DFS-VISIT, DFS si ritrova con il numero dei vertici e può farci quello che vuole. DFS-VISIT funziona come di norma (senza usare i padri che non servono) ma inizializza un contatore (che poi ritornerà) e ogni volta che fa una chiamata ricorsiva aggiorna il valore.

Gli elementi da contare

- Conta i **Vertici in una CC**: Ogni volta che DFS_VISIT trova un nodo WHITE nella lista di adiacenza, incrementa di 1 il contatore.

Pseudocodice

```
1 DFS_COUNT(G)
2     for ogni u ∈ V
3         col[u] = WHITE
4     for ogni u ∈ V
5         if col[u] == WHITE
6             num_in_cc = DFS_VISIT_COUNT(u)
7             //Qui fai quello che vuoi con num_in_cc
```

```
1 DFS_VISIT_COUNT(u)
2     col[u] = GRAY
3     count = 0
4     for ogni v ∈ adj[u]
5         if col[v] == WHITE
6             count = count + DFS_VISIT_COUNT(v)
7     col[u] = BLACK
8     RETURN count + 1
```

Capitolo 2

Esercizi FW

2.1 Esattamente 3 vertici Blu

Es 2 esame 2017
Grafo G dove ad ogni vertice è associato un colore R,V,B. Stabilire per ogni coppia di vertici se esiste un cammino con esattamente 3 vertici blu

Problema ausiliario Questo esercizio non necessita di un problema ausiliario, viene semplicemente aggiunta una specifica al sottoproblema di dimensione k in cui si "limita" il numero di vertici blu.

Sottoproblema Dato un grafo $G(V, E, f)$, $k \in \{0, ..., |V|\}$, $b \in \{0, ..., 3\}$ si vuole stabilire se per ogni (i, j) in $V \times V$ esiste un cammino p da i a j con vertici intermedi appartenenti a $\{1, ..., k\}$ con esattamente b vertici Blu.

Variabili Introdotte $D^{n,b}$ Matrice $|V| \cdot |V|$
$d_{ij}^{(k,b)}$ = True se esiste un cammino da i a j con esattamente b vertici BLU con vertici intermedi appartenenti a $\{1, .., k\}$.
con $b = \{0, ..., 3\}$ e $k = \{0, ..., |V|\}$

Caso Base

k=0
Tre casi, con $b \in \{0, 1, 2\}$ perchè $b < k + 2$ siccome con 0 vertici intermedi il cammino (i, j) ha al più due vertici

$$d_{ij}^{(0,0)} = \begin{cases} TRUE & i = j \wedge col(i) \neq B \wedge col(j) \neq B \\ TRUE & i \neq j \wedge (i,j) \in E \wedge col(i) \neq B \wedge col(j) \neq B \\ FALSE & \text{Altrimenti} \end{cases}$$

$$d_{ij}^{(0,1)} = \begin{cases} TRUE & i = j \wedge col(i) = col(j) = B \\ TRUE & i \neq j \wedge (i,j) \in E \wedge col(i) = BXORcol(j) = B \\ FALSE & \text{Altrimenti} \end{cases}$$

$$d_{ij}^{(0,2)} = \begin{cases} TRUE & i \neq j \wedge (i,j) \in E \wedge col(i) = col(j) = B \\ FALSE & \text{Altrimenti} \end{cases}$$

Passo ricorsivo

Se $k \notin cammino$ allora

$$d_{ij}^{(k,b)} = d_{ij}^{(k-1,b)}$$

che per comodità chiameremo $e1$

Se $k \in cammino$ allora dobbiamo distinguere due casi:

$col(k) \neq B$

$$d_{ij}^{(k,b)} = d_{ik}^{(k-1,b_1)} \wedge d_{kj}^{(k-1,b_2)}$$

in cui $b_1 + b_2 == b$, che per comodità chiameremo $e2_a$

$col(k) = B$

$$d_{ij}^{(k,b)} = d_{ik}^{(k-1,b_1)} \wedge d_{kj}^{(k-1,b_2)}$$

in cui $b_1 + b_2 == b + 1$, che per comodità chiameremo $e2_b$

Quindi l'equazione del passo ricorsivo è:

$$d_{ij}^{(k,b)} = \begin{cases} e1 \vee e2_b & k = B \\ e1 \vee e2_a & k \neq B \end{cases}$$

Soluzione del problema

La soluzione del Problema è contenuta in tutti i valori di $D^{|V|,3}$

2.2 Esattamente 3 Archi Blu

Questa è una variante dell'esercizio di settembre 2017 in cui si richiede: Grafo G dove ad ogni vertice 'e associato un colore R,V,B. Stabilire per ogni coppia di vertici se esiste un cammino con esattamente 3 archi blu.

Sottoproblema $\{k, b\}$ Dato un grafo $G(V, E, f)$, $k \in \{0, ..., n\}$, $b \in \{0, ..., 3\}$, si vuole stabilire se per ogni coppia $(i, j) \in V \times V$ esiste un cammino da i a j con vertici intermedi apaprtenenti a $\{0, ..., k\}$ in cui ci sono esattamente b archi blu.

Variabili Introdotte Siano $k \in \{0, ..., n\}$, $b \in \{0, ..., 3\}$ si consideri il sottoproblema di dimensione (k, b). Per ogni $(i, j) \in V \times V$ introduciamo la variabile $d_{i,j}^{k,b}$ così definita:
$d_{(i,j)}^{(k,b)}$: True sse esiste un cammino dal vertice i al vertice j con esattamente b archi blu che utilizza vertici intermedi appartenenti a $\{1, ..., k\}$.
Al sottoproblema (k, b) associamo la variabile $D^{k,b}$ il cui generico elemento $d_{(i,j)}^{(k,b)}$ corrisponde alla coppia (i, j).

Caso Base Con $k = 0$.
Con $k = 0$ Le possibilità sono due: o abbiamo un vertice intermedio (se $(i, j) \in E$ e $i \neq j$) oppure non abbiamo vertici intermedi ($i = j$). Quindi sappiamo che quando $b \in \{2, 3\}$ la variabile è automaticamente false. Di conseguenza nel caso base $k = 0$ abbiamo $d_{(i,j)}^{0,b}$ con $b \in \{0, ..., 3\}$:

$$d_{(i,j)}^{0,0} = \begin{cases} TRUE & \text{sse } i \neq j \wedge (i, j) \in E \wedge (i, j) \neq B \\ TRUE & \text{sse } i = j \\ FALSE & \text{Altrimenti} \end{cases}$$

$$d_{(i,j)}^{0,1} = \begin{cases} TRUE & \text{sse } i \neq j \wedge (i, j) \in E \wedge (i, j) = B \\ FALSE & \text{Altrimenti} \end{cases}$$

$$d_{(i,j)}^{0,2} = \Big\{ FALSE$$

$$d_{(i,j)}^{0,3} = \Big\{ FALSE$$

Passo Ricorsivo con $k > 0$. Consideriamo la generica coppia (i, j). Supponiamo che $b \leq k + 1$ (altrimenti b supererebbe il numero di archi nel

cammino). Considerando un generico cammino p da i a j che usa vertici intermedi appartenenti a $\{0, ..., k\}$ bisogna considerare due casi: $k \in p$ e $k \notin p$.
Se $k \notin p$: $d^{k,b}_{(i,j)} = \left\{ d^{k-1,b}_{(i,j)} \right.$
Per comodità chiameremo questo caso e_1.
Se invece $k \in p$: $d^{k,b}_{(i,j)} = \left\{ d^{k-1,b_1}_{(i,k)} \wedge d^{k-1,b_2}_{(k,j)} \right.$ tale che $b_1 + b_2 = b$.
Per comodità chiameremo questo caso e_2.
L'equazione del passo ricorsivo sarà quindi:

$$d^{k,b}_{(i,j)} = e_1 \vee e_2$$

Soluzione Considerando tutti i $d^{n,3}_{(i,j)}$ è possibile trovare la soluzione del problema.

2.3 Cammino minimo vertici con segno alternato

Esercizio 1 esame 12/09/2016
Sia A l'insieme dei numeri interi escluso 0.
Dato un grafo non orientato (V,E,f) in cui ad ogni vertice è associato un numero intero diverso da zero (mediante la funzione $f : V \to A$), mediante la tecnica della programmazione dinamica si vuole stabilire per ogni coppia di vertici (i, j), se esiste un cammino da i a j avente vertici che danno luogo ad una alternanza del segno dei numeri ad esso associati. RISPONDERE PER PUNTI alle seguenti richieste:

1. Esplicitare e definire le variabili che servono per risolvere il problema
2. Scrivere l'equazione di ricorrenza per il CASO BASE, giustificando perché è fatta in quel modo
3. scrivere la/le equazione/i di ricorrenza per il PASSO RICORSIVO, giustificando perché è/sono fatta/e in quel modo
4. Scrivere qualè la soluzione del problema, espressa rispetto alle variabili introdotte

Variabili introdotte

D^k matrice $|V| \cdot |V|$
$d^k_{i,j}$ True sse esiste un cammino da i a j che da luogo ad una alternanza di segno usando $1, ..., k$ vertici intermedi

Caso Base

Non ci sono vertici intermedi $k = 0$

$$d^0_{i,j} = \begin{cases} TRUE & \text{se } i = j \\ TRUE & \text{se } i \neq j \wedge (i,j) \in E \wedge f(i) \cdot f(j) < 0 \\ FALSE & \text{altrimenti} \end{cases}$$

Passo Ricorsivo

$k > 0$
due casi:

- $k \notin$ cammino $\rightarrow d^k_{ij} = d^{k-1}_{ij}$
- $k \in$ cammino $\rightarrow d^k_{ij} = d^{k-1}_{ik} \wedge d^{k-1}_{kj}$

Quindi, nel passo ricorsivo con $k > 0$:

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

Soluzione del problema

La soluzione del problema è costituita da tutti i valori contenuti in $D^{|V|}$

2.4 Esercizio Arco RED mai seguito da arco BLUE

Dato un grafo $G(V, E, W)$ orientato e senza cappi in cui ad ogni arco è associato un colore tramite la funzione $col : E \rightarrow \{red, blue\}$ calcolare $\forall (i,j) \in V^2$ il peso di un cammino minimo da i a j nella quale un arco *red* non è mai seguito da un arco *blue*

Definiamo il Problema Ausiliario P' Dato G, $\forall (a,b) \in C^2$ calcolare $\forall (i,j) \in V^2$ il peso di un cammino minimo da i a j con colore del primo arco pari ad a e dell'ultimo a b

Sottoproblema k**-esimo di** P' con $k \in \{0, ..., n\}$.
$\forall (a,b) \in C^2$ calcolare $\forall (i,j) \in V^2$ il peso di un cammino minimo da i a j con colore del primo arco pari ad a e dell'ultimo a b e con vertici intermedi $\in \{0, ..., k\}$

Introduco la variabile $D^{(k,a,b)}$ $\forall(i,j) \in V^2$, $d_{ij}^{(k,a,b)}$ è il peso di un cammino minimo da i a j con colore del primo arco pari ad a e dell'ultimo a b e con vertici intermedi $\in \{0,...,k\}$
Introduco la variabile $D^{(k,a,b)}$

Caso Base sottoproblema di P' $k = 0$

$$d_{ij}^{(0,a,b)} \begin{cases} \infty & \text{se } i = j \\ w_{ij} & \text{se } i \neq j \wedge (i,j) \in E \wedge col(i,j) = a = b \\ \infty & \text{altrimenti} \end{cases}$$

Passo ricorsivo sottoproblema di P' $k > 0$
Abbiamo due casi
se $k \notin cammino$ allora $d_{ij}^{(k,a,b)} = d_{ij}^{(k-1,a,b)}$
se $k \in cammino$ allora
grafo: i [a]- - -[c] k [d]- - -[b] j con $(c,d) \neq (red, blue)$ tradotto: Se k appartiene al cammino minimo, allora il colore dell'ultimo arco entrante in k deve essere diverso da red, oppure il colore del primo arco uscente da k deve essere diverso da blue. quindi: $d_{ij}^{(k,a,b)} = d_{ik}^{(k-1,a,c)} + d_{kj}^{(k-1,d,b)}$ con $c,d \in C^2$ tc $c \neq red \vee b \neq blue$
Quindi l'equazione di ricorrenza del passo ricorsivo è:
$d_{ij}^{(k,a,b)} = min\{d_{ij}^{(k-1,a,b)}, d_{ik}^{(k-1,a,c)} + d_{kj}^{(k-1,d,b)}\}$ con $c,d \in C^2$ tc $c \neq red \vee b \neq blue$

Soluzione Problema Ausiliario P' è formata dalle matrici:
$D^{(n,red,red)}$ $D^{(n,red,blu)}$ $D^{(n,blu,red)}$ $D^{(n,blu,blu)}$
Ovvero tutte le combinazioni possibili di cammini minimi con gli archi a e b = red or blu

Soluzione Problema di partenza PB

$$d_{ij}^{prob} = \begin{cases} 0 & \text{se } i = j \\ min\{d_{ij}^{(n,a,b)} \forall (a,b) \in C^2\} & \text{altrimenti} \end{cases}$$

Capitolo 3

Esercizi DFS

3.1 Esercizio k alberi e almeno h vertici

Esercizio 2 esame 12/09/2016
Scrivere un algoritmo che, dati un grafo non orientato e due interi positivi $h > 0$ e $k > 0$, stabilisce se ENTRAMBE le seguenti condizioni sono verificate

- Esattamente k componenti connesse del grafo sono alberi
- Ogni componente connessa del grafo ha almeno h vertici

Spiegazione

Per creare questo algoritmo occorre modificare DFS:

- Un grafo è un albero se G *è aciciclo* $\vee$ $|E| = |V| - 1$.
  Per capire quanti alberi ci sono, inizializzo un boolean TREE a True. In visit controllo gli archi all'indietro e metto TREE = False se ne trovo. Al ritorno in DFS, se TREE è True incremento un contatore che alla fine di DFS controllerò che equivalga a k.
- Per capire il numero di Vertici in una CC, in VISIT incremento un contatore ricorsivamente ogni volta che incontro un nodo white. Una volta tornato in DFS controllo che la CC abbia almeno h vertici.

Pseudocodice

```
1   DFS-CONDITIONS(G,h,k)
2       for ogni u ∈ V
3           col[u] = WHITE
```

```
4           π[u] = NIL
5       tree = True #variabile globale
6       n_trees = 0
7       for ogni u ∈ V
8           if col[u] == White
9               n_vertici = DFS-VISIT-CONDITIONS(u)
10              if n_vertici < h
11                  RETURN False
12              if tree == True
13                  n_trees = n_trees + 1
14              else
15                  tree = true
16                  #reset senza incrementare n_trees
17      if n_trees == k
18          RETURN True
19      else
20          RETURN False
```

```
1   DFS-VISIT-CONDITIONS(u)
2       col[u] = Gray
3       n_vertici = 0
4       for ogni w ∈ Adj[u]\π[u]
5           if col[w]==White
6               π[w] = u
7               n_vertici = n_vertici + DFS-VISIT-CONDITIONS(w)
8           else if col[w] == Gray and tree = True
9               tree = False
10      col[u] = Black
11      return n_vertici + 1
```

3.2 h vertici "pari" e k cc complete

Esercizio 20 giugno 2018 Dati un grafo a cui ogni vertice è associato un numero intero stabilire se:

- ogni componente connessa del grafo ha almeno h vertici a cui è associato un numero pari
- ci sono k componenti connesse del grafo che, singolarmente, sono grafi completi.

```
DFS(G,h,k)
  for u ∈ V
    col[u] = white
  n_comp = 0 #globale
  foreach u ∈ V
    if col[u] == white
      nv,np,na = 0 #globali
      DFS_VISIT(G,u)
      if np < h
        return False
      na = na/2
      if na = nv(nv-1)/2
        n_comp ++
  if n_comp == k
    return True
  return False
```

```
DFS_VISIT(G,u)
  color[u]=gray
  for w ∈ adj[u]
    na ++
    if w==white
      nv ++
      DFS_visit(G,w)
  if u is pari
    np ++
  col[u]=black
  nv++
```

3.3 Esercizio k grafi completi e h vertici

Dati due interi $h, k > 0$ e un grafo non orientato in cui ad ogni vertice è associato un simbolo ($ e *) stabilisci se: ogni CC del grafo ha almeno h vertici a cui è associato il simbolo $ e ci sono al più k componenti connesse del grafo che prese singolarmente sono grafi completi.

spiegazione

- *h vertici per cc...*, ogni volta che trovo un vertice white aumento un counter se è associato al dollaro, e alla fine controllo se è $> h$.

- *k cc sono....*, un grafo è completo se ci sono $n_{vert} \cdot (n_{vert} - 1)/2$ archi. per contare gli archi si incrementa un counter ogni volta che si trova un vertice nella lista di adiacenza.

Pseudocodice

```
1  DFS_CUSTOM(G,h,k)
2      for ogni u ∈ V
3          col[u] = WHITE
4          π[u] = NIL
5      n_completi = 0
6      n_archi = 0
7      for ogni u ∈ V
8          if col[u] == WHITE
9              n_dollari = DFS_VISIT_CUSTOM(u)
10             if n_dollari < h
11                 RETURN False
12             if n_archi = n_vertici · (n_vertici - 1)/2
13                 n_completi = n_completi + 1
14             n_archi = 0
15     if n_completi > k
16         RETURN False
17     RETURN True
```

```
1  DFS_VISIT_CUSTOM(u)
2      col[u] = GRAY
3      if f(u) == "dollar"
4          n_dollar = 1
5      else
6          n_dollar = 0
7      for ogni w ∈ Adj[u]\π[u]
8          n_archi = n_archi + 1
9          if col[w] == WHITE
10             π[w] = u
11             n_dollar = n_dollar + DFS_VISIT_CUSTOM(w)
12     RETURN n_dollar
```

3.4 3 alberi con k archi e h cc complete

Testo Scriver un algoritmo che, dati un grafo non orientato e due interi positivi $h > 0$ e $k > 0$, stabilisce se ENTRAMBE le seguenti condizioni sono verificate:

- Esattamente 3 componenti connesse del grafo sono alberi con k archi.
- Ci sono al più h componenti connesse del grafo che, prese singolarmente sono grafi completi.

Spiegazione

- Per controllare che la cc è un albero, controllo se trovo degli archi all'indietro. In quel caso sono sicuro che non sia un albero.
- Per controllare che un grafo è connesso, devo avere il numero degli archi uguale a: $|E| = |V| \cdot (|V| - 1)/2$

```
DFS(G)
  for ogni u ∈ V
    col[u] = White
    π[u] = NIL
  NT = 0, NC= 0, tree = True #globale
  for ogni v ∈ V
    if col[v] == White
      n_vertici, n_archi = DFS_VISIT(G,v)
    n_archi = n_archi/2
    if tree == True
      if n_archi == k
        NT ++
    elif n_archi == n_vertici * (n_vertici-1)/2
      NC++
    tree = True
  if NT == 3 AND NC == h
    return True
  return False
```

```
DFS_VISIT(G,v)
  col[v] = Gray
  n_vertici = 0
  n_archi = 0
```

```
  for ogni w ∈ adj[v]\π[v]
    n_archi ++
    if col[w] == White
      n_vertici ++
      n_vertici n_archi += DFS_VISIT(G,w)
    elif col[w] == Gray
      tree = False
  col[v] = Black
  return n_vertici+1, n_archi
```

3.5 Esercizio conta CC di un grafo

Scrivere un algoritmo che determina il numero di componenti connesse di un grafo G = (V, E) non orientato.

Spiegazione

Ogni volta che DFS invoca DFS-VISIT significa che ha trovato una componente connessa, quindi basta mettere un contatore che incrementa ogni volta che si chiama DFS-VISIT da DFS

Pseudocodice

```
1  DFS_COUNT_CC(G)
2      for ogni u ∈ V
3          col[u] = WHITE
4      count = 0
5      for ogni u ∈ V
6          if col[u] == WHITE
7              count = count + 1
8              DFS_VISIT(u)
9      RETURN count
```

3.6 Esercizio Controlla se G è Aciclico

Modificare l'algoritmo DFS di visita di un grafo orientato G in maniera tale che stabilisca se G è acicilico, ossia se non contiene cicli.

Spiegazione

Un grafo è aciclico se non contiene nessun arco all'indietro (btw se un grafo non orientato è aciclico allora è un albero).
Quindi bisogna semplicemente iniziallizzare un Boolean "acyclic" a TRUE e farlo diventare FALSE qual'ora si incontrasse un arco all'indietro

Pseudocodice

```
1  DFS-ACYCLIC(G)
2      for ogni u ∈ V
3          col[u] = WHITE
4          π[u] = NIL
5      time = 0
6      acyclic = true
7      for ogni u ∈ V
8          if col[u] == WHITE
9              DFS-VISIT-ACYCLIC(u)
10     return acyclic
```

```
1  DFS-VISIT-ACYCLIC(u)
2      col[u] = GRAY
3      time++
4      d[u] = time
5      for ogni w ∈ Adj[u]
6          if color[w] == WHITE
7              π[w] = u
8              DFS-VISIT-ACYCLIC(w)
9          else if col[w] == GRAY and acyclic == true
10             acyclic = FALSE
11     col[u] = black
12     time ++
13     f[u] = time
```

In caso di grafo NON ORIENTATO

Se il grafo non è orientato, l'algorimto è lo stesso ma nel controllo degli adiacenti di u in visit bisogna togliere il padre di u

Capitolo 4

Esami

4.1 20 Giugno 2018

Esercizio 1: FW

testo Sia dato un grafo $G(V, E, f)$ (senza cappi) in cui ad ogni vertice è associato un colore tramite la funzione $f : V \to C$, dove C è l'insieme dei colori R, B, N. Mediante la tecinca della programmazione dinamica, si vuole stabilire, per ogni coppia di vertici (i, j), se esiste un cammino da i a j che contenga esattamente 3 vertici rossi.

Sottoproblema di dimensione $k \in 0, ..., |V|$ e $r \in 0, ..., 3$. dato un grafo si vuole stabilire se esiste un cammino p da i a j con esattamente r vertici rossi che usa archi intermedi appartenenti a $\{0, ..., k\}$.
A ogni sottoproblema è associata la variabile d_{ij}^{kr} che assume valore true sse ...

Caso Base con $k = 0$, ho al più due vertici, quindi:

$$d_{ij}^{00} = \begin{cases} True & i \neq j \wedge f(i) \neq r \wedge f(j) \neq r \\ True & i = j \wedge f(i) \neq r \\ False & \text{Altrimenti} \end{cases}$$

$$d_{ij}^{01} = \begin{cases} True & i \neq j \wedge f(i) = r XOR f(j) = r \\ True & i = j \wedge f(i) = r \\ False & \text{Altrimenti} \end{cases}$$

$$d_{ij}^{02} = \begin{cases} True & i \neq j \wedge f(i) = r \wedge f(j) = r \\ False & \text{Altrimenti} \end{cases}$$

In tutti i casi in cui $r = 3$ con $k = 0$ la variabile vale false.

Passo ricorsivo Se $k > 0$ ho tre possibilità:

$$k \notin p \implies d_{ij}^{kr} = d_{ij}^{k-1,r}$$

Chiameremo questo caso e_1
Se $k \in p$ distinguo due casi:
$k \in p \wedge f(k) = r$

$$d_{ij}^{kr} = \vee\{d_{ik}^{k-1,r_1} \wedge d_{kj}^{k-1,r_2}\}$$

$(r_1, r_2) \in r^2 | r_1 + r_2 = r + 1$
$k \in p \wedge f(k) \neq r$

$$d_{ij}^{kr} = \vee\{d_{ik}^{k-1,r_1} \wedge d_{kj}^{k-1,r_2}\}$$

$(r_1, r_2) \in r^2 | r_1 + r_2 = r$
Per semplicità chiamerò questi casi e_{2a} e e_{2b}
L'equazione di ricorrenza per il caso passo quindi sarà:

$$d_{ij}^{kr} = \begin{cases} e_1 \vee e_{2a} & f(k) = r \\ e_1 \vee e_{2b} & f(k) \neq r \end{cases}$$

soluzione Per ogni (i,j) la soluzione sarà in d_{ij}^{n3}.

4.2 12 Settembre 2016

4.2.1 Esercizio 1

Testo Sia A l'insieme dei numeri interi escluso 0.
Dato un grafo non orientato (V, E, f) in cui ad ogni vertice è associato un numero intero diverso da zero (mediante la funzione $f : V \to A$), mediante la tecnica della programmazione dinamica, si vuole stabilire, per ogni coppia di vertici (i, j), se esiste un cammino da i a j avente vertici che danno luogo ad una alternanza del segno dei numeri ad essi associati. RISPONDERE PER PUNTI alle seguenti richieste

1. Esplicitare e definire le variabili che servono per risolvere il problema
2. Scrivere l'equazione di ricorrenza per il CASO BASE, giustificando perchè è fatta in quel modo
3. Scrivere la/le equazione/i di ricorrenza per il PASSO RICORSIVO, giiustificando perchè è/sono fatte in quel modo

4. scrivere qual'è la soluzione del problema, espressa rispetto alle variabili introdotte.

1: Introduco la funzione $s : A \to \{+, -\}$.

Sottoproblema k-esimo Dato un grafo (V, E, f) e $k \in \{0, ..., n\}, \forall (i, j) \in V \times V$ si vuole stabilire se esiste un cammino p da i a j avente vertici intermedi appartenenti a $\{1, ..., k\}$ e in cui non ci siano due vertici consecutivi a cui è associato un numero con lo stesso segno.
Ad ogni sottoproblema di dimensione k è associata una variabile:

Variabile Sia $k \in \{0, ..., n\}$, $\forall (i, j) \in V \times V$, introduciamo la variabile $d^{(k)}_{(i,j)}$ così definita:

- TRUE, se esiste un cammino p da i a j avente vertici intermedi appartenenti a $\{ ..., k\}$ e in cui non ci siano due vertici consecutivi a cui è associato un numero con lo stesso segno.
- FALSE, altrimenti

Pertanto al sottoproblema k-esimo è associata la macrovariabile $D^{(k)}$ il cui generico elemento è $d^{(k)}_{(i,j)}$.

2: CASO BASE $k = 0$.
Si consideri la generica coppia di vertici (i, j). Nel problema si vuole stabilire se esiste un cammino p da i a j avente vertici intermedi appartenenti a $\{1, ..., k\}$ e in cui non ci siano due vertici consecutivi a cui è associato lo stesso segno. Se k è pari a 0, allora non esistono vertici intermedi. Per cui sono possibili due situazioni:

- Può esistere al massimo un arco che collega i e j (se essi sono due vertici distinti), In tal caso la variabile vale TRUE sse $s(f(i)) \neq s(f(j))$.
- Se non esiste alcun arco tra i e j e sono nel caso degenere per cui $i = j$, allora la variabile vale TRUE.

In tutti gli altri casi la variabile vale false.

$$d^{(0)}_{(i,j)} = \begin{cases} TRUE & \text{se } i \neq j \wedge (i, j) \in E \wedge s(f(i)) \neq s(f(j)) \vee i = j \\ FALSE & \text{Altrimenti} \end{cases}$$

3 PASSO RICORSIVO $k \geq 0$ Si consideri la generica coppia (i, j) e sia $k \in \{1, ..., n\}$.
Si possono presentare due diverse situazioni:

- $k \notin p$: Se k non è un vertice intermedio di p (tra i e j), allora tutti i vertici intermedi di p appartengono all'insieme $\{1, ..., k-1\}$. Esiste un cammino p da i a j avente vertici intermedi appartenenti a $\{1, ..., k\}$ e in cui non si siano due vertici consecutivi a cui è associato un numero con lo stesso segno se esiste un cammino p da i a j avente vertici intermedi appartenenti a $\{1, ..., k-1\}$ con le medesime condizioni

  $$\forall (i, j) \in V \times V, d^{(k)}_{(i,j)} = d^{(k-1)}_{(i,j)} \text{ sse } k \notin p$$

  Per semplicità chiamo questo caso C_1

- $k \in p$: se k è un vertice intermedio di p (tra i e j), suddivido p in due sottocammini p_1 da i a k e p_2 da k a j, entrambi aventi vertici intermedi appartenenti all'insime $\{1, ..., k-1\}$. Per la proprietà della sottostruttura ottima si avrà:

  $$\forall (i, j) \in V \times V, d^{(k)}_{(i,j)} = d^{(k-1)}_{(i,k)} \wedge d^{(k-1)}_{(k,j)} \text{ sse } k \in p$$

  Per semplicità chiamo questo caso C_2

Poichè non posso sapere a priori se k sia o meno vertice intermedio di p, avrò il caso passo:

$$\forall (i, j) \in V \times V, d^{(k)}_{(i,j)} = C_1 \vee C_2$$

4 Soluzione: La soluzione del problema PB è costituita da tutti i valori contenuti in $D^{(n)}$.

4.3 20 Settembre 2018

4.3.1 E1: FW

Testo Sia dato un grafo colorato $G = (V, E, col)$, dove la funzione $col : E \to \{rosso, nero\}$ associa ad ogni arco di G il colore rosso oppure il colore nero. Si vuole valutare con un algoritmo di programmazione dinamica se, per ogni coppia di vertici, esiste un cammino nel quale non vi sono archi consecutivi neri. Si chiede di:

1. Esplicitare quali sono le variabili che vanno definite

2. Scivere l'equazone di ricorrenza per il caso BASE, giustificando perchè è fatta in quel modo.

3. Scrivere la/le equazione/i di ricorrenza per il passo ricorsivo, giustificando perchè è/sono fatta/e in quel modo

4. Scrivere qual'è la soluzione del problema, espressa rispetto alle variabili introdotte.

Sottoproblema k-esimo Introduco il sottoproblema k-esimo, in cui dato un grafo colorato $G = (V, E, col)$, dove la funzione $col : E \to \{rosso, nero\}$ associa ad ogni arco di G il colore rosso oppure il colore nero, si vuole stabilire $\forall k \in \{0, ..., |V|\}$ se per ogni coppia di vertici $(i, j) \in V^2$ esiste un cammino p da i a j che utilizza vertici appartenenti a $\{0, ..., k\}$ in cui non esisto archi consecutivi di colore nero.

Variabile Introduco quindi la variabile d_{ij}^k che $\forall (i, j) \in V^2$ e $\forall k \in 0, ..., |V|$ vale True sse esiste un cammino p da i a j che utilizza vertici intermedi appartenenti a $\{1, ..., k\}$ in cui non vi sono due archi consecutivi di colore nero.

Problema Ausiliario Siccome mi mancano informazioni, introduco il problema ausiliario P': Sia dato un grafo colorato $G = (V, E, col)$, dove la funzione $col : E \to \{rosso, nero\}$ associa ad ogni arco di G il colore rosso oppure il colore nero, si vuole determinare per ogni coppia di $(i, j) \in V^2$ e per ogni coppia di $(a, b) \in \{rosso, nero\} \times \{rosso, nero\}$ se esiste un cammino p da i a j tale che non vi siano archi consecutivi neri, e che il colore del primo arco sia uguale ad a e quello dell'ultimo arco sia uguale a b.

Sottoproblema k-esimo di P' Si vuole stabilire $\forall k \in 0, ..., |V|$ se dato un grafo colorato $G = (V, E, col)$, dove la funzione $col : E \to \{rosso, nero\}$ associa ad ogni arco di G il colore rosso oppure il colore nero, per ogni coppia di $(i, j) \in V^2$ e per ogni coppia di $(a, b) \in \{rosso, nero\} \times \{rosso, nero\}$, se esiste un cammino p da i a j che utilizza vertici intermedi appartenenti a $\{1, ..., k\}$ tale che non vi siano archi consecutivi neri, e che il colore del primo arco sia uguale ad a e quello dell'ultimo arco sia uguale a b.

Variabile Introduco quindi la variabile $d_{ij}^{k,a,b}$ che vale true sse...

Caso Base con $k = 0$, abbiamo al massimo un arco intermedio. Se $i = j$, non abbiamo archi intermedi quindi la condizione non è mai rispettata. Se invece $i \neq j$ abbiamo un arco intermedio, e la condizione è rispettata soltanto se quell'arco esiste e $col(i, j) = a = b$. Quindi, con $k = 0$

$$d_{ij}^{0,a,b} = \begin{cases} True & \text{sse } i \neq j \wedge (i,j) \in E \wedge col(i,j) = a = b \\ False & \text{Altrimenti} \end{cases}$$

Passo Ricorsivo con $k > 0$
Quando $k > 0$ dobbiamo riconoscere due casi distinti:

- Se $k \notin p$, allora il valore di $d_{ij}^{k,a,b}$ è dato dal valore della variabile con $k - 1$
- Se $k \in p$, il valore di $d_{ij}^{k,a,b}$ è dato dall'unione dei valori dei cammini da i a k e da k a j se rispetta le condizioni date.

Quindi, per $k > 0$:
Se $k \notin p$

$$d_{ij}^{k,a,b} = \Big\{ d_{ij}^{k-1,a,b}$$

per comodità chiameremo questo caso e_1
Se invece $k \in p$

$$d_{ij}^{k,a,b} = \Big\{ d_{ik}^{k,a,c} \wedge d_{kj}^{k,d,b} \quad \text{se } i \neq jc \neq nero \vee d \neq nero$$

Con $a, b, c, d \in rosso, nero$. Chiameremo questo caso e_2.
Quindi il passo ricorsivo è:

$$d_{ij}^{k,a,b} = \Big\{ e_1 \vee e_2$$

Soluzione P' La soluzione del problema ausiliario è contenuta nelle quattro macrovariabili:

$$D^{rr}, D^{rn}, D^{nn}, D^{nr}$$

Solzuone Problema La soluzione del problema iniziale è formata da tutte le soluzioni del problema ausiliario, quindi:

$$D^{rr} \vee D^{rn} \vee D^{nn} \vee D^{nr}$$

4.3.2 E2: DFS

Testo Scrivere un algoritmo che, dati un grafo non orientato e due interi positivi $V1$ e $V2$, decide se il grafo consiste di al più $V1$ componenti connesse ciascuna delle quali, presa singolarmente, è un grafo completo con massimo $V2$ vertici.

Specifiche Si chiede quindi di modificare DFS, controllando che abbia al più $V1$ componenti connesse che sono grafi completi con un massimo di $V2$ vertici. Un *grafo completo* è un grafo semplice nel quale ogni vertice è collegato a tutti gli altri. Per calcolarlo, con a archi e e v vertici: $a = v \cdot \frac{v-1}{2}$

```
DFS(G,V1,V2)
  for u ∈ V
    col[u] = white
  ncc = 0 #globale
  for u ∈ V
    if u == white
      nv=0,na=0
      dfs_visit(G,u)
      na=na/2
      if na=nv(nv-1)/2
        if nv<V2
          return false
      else
        return false
    ncc++
    if ncc > V1
    return false
  return true
```

```
DFS_VISIT(G,u)
  col[u]=gray
  nv++
  for w ∈ adj[u]
    na++
    if col[w] == white
      DFS_VISIT(G,w)
```

4.4 18 Gennaio 2022

4.4.1 Esercizio 1: LIS

Testo Si consideri la sequenza X sull'alfabeto A di lunghezza n ed una funzione $\phi : A \to Z$ che associa ad ogni carattere un numero intero. Si vuole calcolare, mediante la tecnica della programmazione dinamica, la lunghezza di una più lunga sottosequenza di X in cui i numeri associati ai caratteri appaiono alternando un pari ad un dispari. RISPONDERE PER PUNTI alle seguenti richieste

1. Esplicitare e definire le variabili che servono per risolvere il problema
2. Scrivere l'equazione di ricorrenza per il CASO BASE, giustificando perchè è fatta in quel modo
3. Scrivere la/le equazione/i di ricorrenza per il PASSO RICORSIVO, giiustificando perchè è/sono fatte in quel modo
4. Scrivere qual'è la soluzione del problema, espressa rispetto alle variabili introdotte.
5. Scrivere quindi, in pseudocodice, l'algoritmo relativo.

Sottoproblema e Variabili Per il sottoproblema di taglia i introduco la variabile S di dimensione n il cui generico elemento $S[i]$ con $i \in \{1, ..., n\}$ è la lunghezza della più lunga sottosequenza di X che termina con l'i-esimo elemento di X.

Caso Base $i = 1$.
Con una sottosequenza di dimensione 1 le condizioni sono sempre rispettate, quindi:

$$S[1] = \Big\{ 1$$

Passo Ricorsivo $i > 1$ Nel caso in cui $i > 1$ si introduce $0 < j < 1$. $S[i]$ sarà il massimo di tutti gli $S[j]$ in cui $S[j]\%2 \neq S[i]\%2$ sia rispettata, aumentata di 1. Quindi:

$$S[i] = \Big\{ 1 + max\{S[j] | 0 < j < i, S[j]\%2 \neq S[i]\%2\}$$

Soluzione La soluzione del problema è il massimo di tutti gli $S[i]$.

```
def E1(X):
  S[1] = 1
  for i = 2 to n
    max = 0
    for j = 0 to i
      if S[j] > max AND X[j]%2 ≠ X[i]%2
        max = S[j]
    S[i] = max + 1
  return max(S)
```

4.5 10 Febbraio 2022

4.5.1 Esercizio 1: LGCS

Testo Sono date due sequenze X e Y di interi di lunghezza n ed m. Si vuole calcolare, mediante la tecnica della programmazione dinamica, la lunghezza di una più lunga sottosequenza *crescente* comune a X e Y in cui due numeri consecutivi non siano entrambi pari.
Rispondere alle seguenti richieste:

1. Definire le variabili che servono per risolvere il problema.
2. Scrivere l'equazione di ricorrenza per il CASO BASE, giustificando perchè è fatta in quel modo.
3. Scrivere l'equazione di ricorrenza per il PASSO RICORSIVO, giustificando perchè è fatta in quel modo.
4. Dare la soluzione del problema, espressa rispetto alle variabili introdotte.
5. Scrivere in pseudocodice l'algoritmo.

Sottoproblema e variabile Per risolvere il problema introduco il sottoproblema di taglia (i, j) con $0 \leq i \leq n$,$0 \leq j \leq m$ così definito: Date due sequenze X e Y si determini la lunghezza di una delle più lunghe sottosequenze crescenti comuni al prefisso X_i e Y_j in cui non vi sono due numeri pari consecutivi.

Ad ogni sottoproblema è associata la varibile c_{ij} così definita:
c_{ij} = La lunghezza di una delle più lunghe sottosequenze crescenti comuni a X_i e Y_j in cui due numeri consecutivi non siano entrambi pari.

Problema Ausiliario Introduco il problema asuiliario:
Date due sequenze X e Y rispettivamente di n e m numeri interi, si determini la lunghezza di una delle più lunghe sottosequenze crescenti comuni in cui non vi sono due numeri consecutivi pari e che terminano con x_m e y_n *se questi coincidono.*
Di questo problema ausiliario considero i sottoproblemi di taglia (i, j) definiti come:
date due sequenze X e Y rispetivamente di n e m numeri interi, si determini la lunghezza di una delle più lunghe sottosequenze crescenti comuni ai prefissi X_i e Y_j in cui non vi sono due numeri pari consecutivi e che termini con x_i e y_j se essi coincidono.
Il sottoproblema è associato alla variabile c_{ij} così definito:
c_{ij}= La lunhgezza di una delle più lunghe sottosequenze crescenti comuni a X_i e Y_j in cui non vi sono due numeri consecutivi pari e che terminano con x_i e y_j se questi coincidono.

Caso Base Il caso base lo si ha sia nel caso in cui $i = 0 \vee j = 0$, che nel caso in cui x_i e y_j non coincidano. quindi:

$$c_{ij} = \begin{cases} 0 & \text{se } i = 0 \vee j = 0 \vee x_i \neq y_j \end{cases}$$

Passo Ricorsivo Il Passo ricorsivo lo si ha per ogni $i > 0 \wedge j > 0 \wedge x_i = y_j$. In questo caso la variabile cambia se x_i è pari o dispari, quindi:

$$c_{ij} = \begin{cases} 1 + max\{c_{hk} | 0 < h < i, 0 < k < j, x_h < x_i, x_h \%2 = 1 \vee x_i \%2 = 1\} \end{cases}$$

Soluzione La soluzione del problema è il massimo di tutti i c_{ij} con $0 < i \leq m \wedge 0 < j \leq n$