Appunti Analisi e Progetto di Algoritmi

Domande e Risposte

Capitolo 1 – Nozioni introduttive

1) Cosa si intende per alfabeto?

Un alfabeto è un insieme finito e non vuoto di simboli che vengono identificati mediante la lettera maiuscola ∑ (sigma).

Esempio:

→ $\sum = \{0, 1\}$, l'alfabeto binario;

→ $\sum = \{a, b, ..., z\}$, l'alfabeto di tutte le lettere dell'alfabeto;

→ $\sum = \{0, 1, ...,9\}$, l'alfabeto delle cifre da 0 a 9.

2) Cosa si intende per stringa?

Una stringa è una sequenza finita di simboli scelti da un alfabeto.

Esempio: Sia l'alfabeto $\sum = \{0, 1\}$ binario, le possibili stringhe sono:

→ $w_1 = 10110$

→ $w_2 = 1101$

→ $w_3 = 100101$

→ $w_1 = <1,0,1,1,0>$

→ $w_2 = <1,1,0,1>$

→ $w_3 = <1,0,0,1,0,1>$

Uso le ultime lettere x, y, z dell'alfabeto $\sum = \{a, b, ..., x, y, z\}$ per indicare una generica sequenza (o stringa).

Esempio 1:

$X = <x_1, ..., x_n>$

x_1 = primo simbolo di X

x_n = ultimo simbolo di X

Esempio 2:

$X = <G, A, T, T, O>$

$G = x_1$

$A = x_2$

$T = x_3$

$T = x_4$

$O = x_5$

Esempio 3:

Con n = 6, ho che

$Y = <y_1, y_2, y_3, y_4, y_5, y_6>$

Capitolo 2 – LCS (Longest Common Subsequence)

3) Cosa si intende per sequenza?

Si definisce sequenza una successione di elementi topologicamente ordinati presi da un insieme ∑ (esempio: $X = \langle 2, 4, 10, 5, 9, 11\rangle$)

4) Cosa si intende per prefisso di lunghezza k?

Si definisce prefisso di lunghezza k di una sequenza i primi k elementi della sequenza.

Esempi

$X_3 = \langle 2, 4, 10 \rangle$
$X_6 = \langle 2, 4, 10, 5, 9, 11 \rangle = X$
$X_0 = \langle \rangle$

5) Cosa si intende per sottosequenza comune di due sequenze X e Y?

Si definisce sottosequenza comune di due sequenze X e Y, la sottosequenza sia di X che di Y.

Esempio:

$X = \langle 1, 13, 5, 3, 1, 12, 8, 11, 6, 10, 10 \rangle$
$Y = \langle 1, 5, 5, 2, 3, 1, 12, 8, 8, 10 \rangle$
$\langle 5, 3, 1, 8, 10 \rangle$ è sottosequenza comune di X e Y
$\langle 1, 5, 3, 1, 12, 8, 10 \rangle$ è sottosequenza comune di X e Y

6) LCS di due sequenze: teorema (sottostruttura ottima)

Sia (i, j) un generico problema di LCS con $i > 0 \wedge j > 0$, per ogni sottoproblema più piccolo $(\bar{i}, \bar{j})$ sia $Z^{(\bar{i}, \bar{j})}$ una sua soluzione.
Sia $Z_k^{(i,j)} = \langle z_1, z_2, \dots, z_k \rangle$ una soluzione del problema (i, j), allora:
→ se $x_i = y_j$, allora $z_k = x_i = y_j \wedge Z_{k-1}^{(i,j)} = Z^{(i-1, j-1)}$
→ se $x_i \neq y_j \wedge z_k \neq x_i$, allora $Z_k^{(i,j)} = Z^{(i-1, j)}$
→ se $x_i \neq y_j \wedge z_k \neq y_j$, allora $Z_k^{(i,j)} = Z^{(i, j-1)}$

7) Dimostrazione LCS

A – Per assurdo $x_i = y_j$, allora $(z_k \neq x_i \vee z_k \neq y_j) \vee \left(Z_{k-1}^{(i,j)} \neq Z^{(i-1,j-1)}\right)$.
→ Se $z_k \neq x_i$ allora $Z_k^{(i,j)} | x_i$ è una LCS di X_i e Y_j. Quindi $Z_k^{(i,j)}$ non è soluzione del problema (i, j) perchè se $x_i = y_j \wedge z_k \neq x_i$, aggiungendo x_i si ottiene una sottosequenza più lunga, che è assurdo.
→ Se $z_k \neq y_j$ allora $Z_k^{(i,j)} | y_j$ è una LCS di X_i e Y_j. Quindi $Z_k^{(i,j)}$ non è soluzione del problema (i, j) perchè se $x_i = y_j \wedge z_k \neq y_j$, aggiungendo y_j si ottiene una sottosequenza più lunga, che è assurdo.
→ Se $Z_{k-1}^{(i,j)} = Z^{(i-1,j-1)}$ allora $Z_{k-1}^{(i,j)}$ non è soluzione del problema $(i-1, j-1)$. Si pone allora che W sia soluzione di $(i-1, j-1)$.
Si ha $|W| > k - 1$ e $W | x_i$ è una sottosequenza di X_i e Y_j e ha $|W | x_i| > k$.
Allora $Z_k^{(i,j)}$ non è soluzione del sottoproblema (i, j) che è assurdo.
B - Per assurdo $Z_k^{(i,j)}$ non è soluzione del sottoproblema $(i-1, j)$. Sia allora W la soluzione a tale sottoproblema con $|W| > k$. Risulta quindi assurdo che $Z_k^{(i,j)}$ sia

soluzione del problema (i,j) dato che W è sottosequenza di X_i e Y_j e sottosequenza di X_{i-1} e Y_j.
C - Per assurdo $Z_k^{(i,j)}$ non è soluzione del sottoproblema $(i, j-1)$. Sia allora W la soluzione a tale sottoproblema con $|W| > k$. Risulta quindi assurdo che $Z_k^{(i,j)}$ sia soluzione del problema (i,j) dato che W è sottosequenza di X_i e Y_j e sottosequenza di X_i e Y_{j-1}.

8) Stampa LCS ricorsiva

Problema: data la matrice C e gli indici i e j stampare gli elementi della sequenza ricavata mediante LCS.
Input: matrice C e indici i e j di una cella di C
Output: LCS(X_i , Y_j)
Pseudocodice

```
function stampa_LCS(C, i, j)
    if i > 0 and j > 0 then
      if xi = yj then
         stampa_LCS(C, i-1, j-1)
         print xi
   else
     if C[i,j] = C[i-1,j] then
        stampa_LCS(C, i-1, j)
     else
        stampa_LCS(C, i, j-1)
```

Complessità Computazionale
$T(m,n) = O(m * n)$

Capitolo 3 – LIS (Longest Increasing Subsequence)

9) Cosa si intende per Increasing Subsequence?

Si definisce Increasing Subsequence la sequenza $Z = \langle z_1, z_2, ..., z_k \rangle$ tale che $z_i < z_{i+1}$ per ogni indice i da 1 a $k - 1$.
Esempi di sequenze crescenti:
→ ⟨2, 4, 10⟩
→ ⟨2, 4, 7, 13, 21⟩
→ ⟨2⟩

10) Cosa si intende per Longest Increasing Subsequence (LIS)

Data $X = \langle x_1, x_2, ..., x_m \rangle$, la più lunga sottosequenza di X che sia crescente è $Z = LIS(X)$
Esempio:
X = ⟨14, 2, 4, 2, 7, 0, 13, 21, 11⟩
LIS(X) = ⟨2, 4, 7, 13, 21⟩

11) Problema: Longest Increasing Subsequence (LIS)

P: Data una sequenza $X = \langle x_1, x_2, \dots, x_m \rangle$, trovare la più lunga sottosequenza crescente $Z = \mathrm{LIS(X)}$. P è un problema di ottimizzazione di massimo, dove:
→ (m): dimensione del problema
→ Soluzioni possibili: tutte le sottosequenze crescenti di X
→ Funzione obiettivo: lunghezza
→ |Z| è il valore ottimo del problema
→ Z è una soluzione ottimale

12) Sottoproblemi e variabili associate

Il sottoproblema di dimensione (i) é definito come segue:
P: Data una sequenza X di m numeri interi, si determini la lunghezza di una tra le più lunghe sottosequenze crescenti di X.
Dato che $0 \leq i \leq m$, si ottengono $m + 1$ sottoproblemi e ad ogni sottoproblema di PBR è associata una variabile. Considerato il sottoproblema di dimensione (i), la variabile ad esso associata è c_i ed è così definita:
c_i = lunghezza di una tra le più lunghe sottosequenze crescenti di X_i.

13) LIS: Teorema e dimostazione della proprietà della sottostruttura ottima

Teorema: Sia X una sequenza di m numeri interi e sia X_i un suo prefisso di lunghezza i con $1 \leq i \leq m$. Sia Z^i una tra le più lunghe sottosequenze crescenti di X_i e che termina con x_i.
Allora vale che $Z^i = Z^* \mid x_i$, con $Z^* \in W_i$ e $|Z^*| = max\{|W|: W \in W_i\}$ dove W_i è l'insieme di tutte le sottosequenze crescenti di X_j che finiscono con x_j e a cui è possibile concatenare x_i.

Dimostrazione: Per assurdo ora si supponga che $Z^* \mid x_i$ non si la soluzione del problema $i - esimo$. Allora, relativamente alla soluzione Z^i del problema valgono le seguenti affermazioni:
→ $Z^i = Z' \mid x_i$
→ $|Z'| > |Z^*|$
dove Z' è una qualche sottosequenza crescente di un prefisso più piccolo di X_i.
Sia ora z' l'ultimo elemento di Z'. Sia $h < i$ il più grande indice tale che $x_h = z'$.
Di conseguenza, per come è stato definito W_i, si ottiene che $Z' \in W_i$. Infatti Z' è una sottosequenza crescente di X_h, la quale termina con $x_h < x_i$. Ciò però porta ad una contraddizione tra l'affermazione $|Z'| > |Z^*|$ e l'ipotesi $|Z^*| = max\{|W|: W \in W_i\}$.

14) Equazione di ricorrenza

Un'equazione di ricorrenza è composta da:

→ caso base: definisce i casi più semplici che possono essere subito risolti senza ricorrere alle soluzioni dei sottoproblemi più piccoli e lo si ha per un qualunque sottoproblema di dimensione (i) con $i = 0 \vee i = 1$, ossia quando il prefisso considerato è la sequenza vuota oppure è una sequenza composta da un singolo elemento

$c_i = 1 \quad \text{se } i = 1$

→ passo ricorsivo: lo si ha per un qualunque sottoproblema di dimensione (i) tale che $i > 1$, ossia quando si considera un prefisso della sequenza X in input di almeno due elementi e i dati disponibili per calcolare c_i sono: l'input X ed in particolare l'elemento x_i e tutte le variabili $\{c_0, \dots, c_{i-1}\}$.

Il passo ricorsivo è quindi scrivibile come:

$c_i = 1 + \max\{c_h \mid 1 \leq h < i \wedge x_h < x_i\}$

i	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
x_i	ε	2	4	7	6	11	13	21	14	1
	0	1	2	3	3	4	5	6	6	1

15) Algoritmo ricorsivo

```
function LISRic(i):
    if i = 1 then
        return 1
    else
        max := 0
        for h ← 1 to i – 1do
            if xh < xi then
                S ← LISRic(h)
                if S > max then
                    max ← S
        return 1 + max
```

16) Implementazione bottom up

Con la tecnica bottom-up tutti i valori vengono calcolati in modo tale da risolvere ogni sottoproblema una volta sola. Questo permette di risolvere il problema in $O(m^2)$ occupando $\Theta(m)$ memoria.

Procedura

```
function LIS(X):
    c[1] ← 1
    max ← c[1]
    for i ← 2 to m do
        temp ← 0
        for h ← 1 to i – 1 do
            if (xh < xi) ∧ (c[h] > temp) then
                temp ← c[h]
        c[i] ← 1 + temp
        if c[i] > max then
            max ← c[i]
    return max
```

Capitolo 4 – LICS (Longest Increasing Common Subsequence)

17) Problema: Longest Increasing Common Subsequence (LICS)

P1: Date due sequenze X e Y, rispettivamente di m e n numeri interi, si determini una tra le più lunghe sottosequenze crescenti comuni a X e Y.

Esempio: $X = \langle 2, 4, 7, 11, 21, 14, 1 \rangle, Y = \langle 2, 7, 4, 23, 21, 14, 1, 8 \rangle$
$Z = \langle 2, 4, 21 \rangle = \text{LICS}(X, Y)$

P2: Date due sequenze X e Y, rispettivamente di m e n numeri interi, si determini la lunghezza tra le più lunghe sottosequenze crescenti comuni a X e Y.

Esempio:
$X = \langle 2, 4, 7, 11, 21, 14, 1 \rangle$
$Y = \langle 2, 7, 4, 23, 21, 14, 1, 8 \rangle$
$|Z| = 3$

18) Sottoproblemi e variabili associate

Il sottoproblema di dimensione (i, j) è definito come segue:
"Date due sequenze X e Y, rispettivamente di m e n numeri interi, si determini la lunghezza di una tra le più lunghe sottosequenze crescenti comuni al prefisso X_i e al prefisso Y_j" .
Dato che $0 \leq i \leq m$ e $0 \leq j \leq n$, si ottengono $(m + 1) \cdot (n + 1)$ sottoproblemi (i e j possono valere 0 in quanto si deve considerare anche il caso in cui un prefisso sia la sequenza vuota). Considerato il sottoproblema di dimensione (i, j), la variabile ad esso associata è $c_{i,j}$ ed è così definita:

$$c_{i,j} = \textit{lunghezza di una tra le più lunghe sottosequenze crescenti comuni a } X_i \textit{ e } Y_j$$

19) Teorema e dimostazione della proprietà della sottostruttura ottima

Teorema: Sia X una sequenza di m numeri interi e sia X_i un suo prefisso di lunghezza i con $1 \leq i \leq m$. Sia Y una sequenza di n numeri interi e sia Y_j un suo prefisso di lunghezza j con $1 \leq j \leq n$. Sia $Z^{i,j}$ una tra le più lunghe sottosequenze crescenti di X_i e Y_j tale che termini con $x_i = y_j$. Allora vale che $Z^{i,j} = Z^* | \, x_i$, con $Z^* \in W_{i,j}$ e $|Z^*| = max\{|W| : W \in W_{i,j}\}$ dove $W_{i,j}$ è l'insieme di tutte le sottosequenze crescenti comuni di X_h e Y_k che finiscono con $x_h = y_k$ e a cui è possibile concatenare x_i (o y_j).

Dimostrazione: Per assurdo supponiamo che $Z^* | \, x_i$ non si la soluzione del problema $(i, j) - esimo$. Allora, relativamente alla soluzione Z^i del problema valgono le seguenti affermazioni:
→ $Z^i = Z' \, | \, x_i$
→ $|Z'| > |Z^*|$
dove Z' è una qualche sottosequenza crescente di un prefisso più piccolo di X_i.

Sia ora z' l'ultimo elemento di Z'. Vale quindi che $z' < x_i = y_j$, poichè è stato possibile concatenare x_i (o y_j) a Z'. Inoltre, siano $r < i$ e $s < j$ i più grandi indici tale che $x_r = y_s = z'$. Di conseguenza, per come è stato definito W_i, si ottiene che $Z' \in W_{i,j}$. Infatti Z' è una sottosequenza comune crescente di X_r e Y_s, la quale termina con $x_r < x_i$. Ciò però porta ad una contraddizione: infatti dal punto 2 vale che $|Z'| > |Z^*|$, ma ciò è in contraddizione con l'ipotesi che $|Z^*| = max\{|W|: W \in W_{i,j}\}$.

20) Equazione di ricorrenza

Un'equazione di ricorrenza è composta da:

→ caso base: definisce i casi più semplici che possono essere subito risolti senza ricorrere alle soluzioni dei sottoproblemi più piccoli e lo si ha per un qualunque sottoproblema di dimensione (i, j) con $i = 0 \vee j = 0$ ma anche $x_i \neq y_j$ossia quando i due prefissi considerati terminano con due elementi diversi

$c_{i,j} = 0$

→ passo ricorsivo: lo si ha per un qualunque sottoproblema di dimensione (i, j) tale che $x_i = y_j$, ossia quando i due prefissi X_i e Y_j è uguale alla lunghezza della più lunga sottosequenza crescente comune calcolata per un sottoproblema di dimensione minore di $x_i = y_j$ aumentata di uno. Il passo ricorsivo è quindi scrivibile come:

$c_{i,j} = 1 + \max\{c_{h,k} \mid 1 \leq h < i, 1 \leq k < j, x_h < x_i\}$

		1	2	3	4	5	6	7	8	j
		2	7	4	23	21	14	1	8	y_j
1	2	1	0	0	0	0	0	0	0	
2	4	0	0	2	0	0	0	0	0	
3	7	0	2	0	0	0	0	0	0	
4	11	0	0	0	0	0	0	0	0	
5	21	0	0	0	0	3	0	0	0	
6	14	0	0	0	0	0	3	0	0	
7	1	0	0	0	0	0	0	1	0	
i	x_i									

21) Algoritmo ricorsivo

```
procedure LGCSRic(i, j)
    if i = 0 ∨ j = 0 ∨ x_i ≠ y_j then
        return 0
    else
        max ← 0
        for h ← 1 to i − 1 do
            for k ← 1 to j − 1 do
                if x_h < x_i then
                    S ← LGCSRic(h, k)
                    if S > max then
                        max ← S
        return 1 + max
```

22) Implementazione bottom up

Con la tecnica bottom-up tutti i valori vengono calcolati in modo tale da risolvere ogni sottoproblema una volta sola. Questo permette di risolvere il problema in $O(m^2 * n^2)$ occupando $\Theta(m * n)$.

Procedura

```
procedure LGCS(X, Y)
    max ← 0
    for i ← 1 to m do
        for j ← 1 to n do
            if x_i ≠ y_j then
                c[i, j] ← 0
            else
                temp ← 0
                for h ← 1 to i − 1 do
                    for k ← 1 to j − 1 do
                        if (x_h < x_i) ∧ (c[h, k] > temp) then
                            temp ← c[h, k]
                c[i, j] ← 1 + temp
            if c[i, j] > max then
                max ← c[i, j]
    return max
```

Capitolo 5 – Longest Common Subsequence (LCS) con al più K elementi rossi

23) Input e Output del Problema

Input: $X = \langle x1, x2, x3, \ldots, xm \rangle,\ Y = \langle y1, y2, y3, \ldots, yn \rangle$: $|X| = m\ \wedge\ |Y| = n\ \wedge$ $K \in Z+$.

Sia ora la funzione

$col: N \rightarrow C\ \subseteq\ \sum *$

$C = \{\text{'Rosso'}, \text{'Blu'}, \text{'Nero'}\}$
Output: $|LCS_k(X, Y)| = |\langle z_1, z_2, z_3, \ldots, z_k\rangle|$ con al più k elementi rossi

24) Definizione dei sottoproblemi e variabili associate

Trovare la LCS dei prefissi X_i e Y_j che ha al più k elementi rossi → $LCS_k(X_i, Y_j)$
$i \in \{0, 1, 2, \ldots, m\}$
$j \in \{0, 1, 2, \ldots, n\}$
$k \in \{0, 1, 2, \ldots, K\}$
Numero di sottoproblemi: $(m + 1) * (n + 1) * (K + 1)$
Coefficiente del sottoproblema di dimensione (i, j, k): $c_{i,j,k}$ è la lunghezza della LCS dei prefissi X_i e Y_j che ha al più k elementi rossi.

25) Equazione di ricorrenza: caso base

Se $i = 0 \vee j = 0$ → $c_{i,j,k} = 0$, t.c. $k \in Z^+$

26) Equazione di ricorrenza: passo ricorsivo

Se $i > 0 \wedge j > 0$, allora
se $x_i \neq y_j$ → $c_{i,j,k} = \max\{c_{i-1,j,k}, c_{i,j-1,k}\}$
se $x_i = y_j \wedge col(x_i) \neq \text{'Rosso'}$ → $c_{i,j,k} = c_{i-1,j-1,k} + 1$
se $x_i = y_j \wedge col(x_i) = \text{'Rosso'} \wedge k > 0$ → $c_{i,j,k} = c_{i-1,j-1,k-1} + 1$
se $x_i = y_j \wedge col(x_i) = \text{'Rosso'} \wedge k = 0$ → $c_{i,j,k} = c_{i-1,j-1,k-1}$

27) Soluzione del problema

La soluzione di tale problema è il coefficiente $c_{m,n,k}$

28) Algoritmo Bottom Up per il calcolo del valore ottimo

```
function LCS_at_most_k_red(X, Y, K)
        m := X.length
        n := Y.length

        for k := 0 to K do
                for i := 0 to m do
                        ck[i, 0] := 0
        for k := 0 to K do
                for j := 0 to n do
                        ck[0, j] := 0
        for k := 0 to K do
                for i := 1 to m do
                        for j := 1 to n do
                                if xi ≠ yj then
                                        ck[i, j] := max{ck[i - 1, j], ck[i, j - 1]}
                                else
                                        if col(xi) ≠ 'Rosso' then
                                                ck[i, j] := ck[i - 1, j - 1] + 1
                                        else
                                                if k > 0 then
                                                        ck[i, j] := ck-1[i - 1, j - 1] + 1
                                                if k = 0 then
                                                        ck[i, j] := ck-1[i - 1, j - 1]
             return ck[m, n]
```

29) Ricostruzione dell'algoritmo

```
function LCS_most_k_red(i, j, k)
        if i > 0 ∧ j > 0 then
                if xi ≠ yj then
                        return max{LCS_most_k_red(i - 1, j, k), LCS_most_k_red(i, j - 1, k)}
                else
                        if col(xi) ≠ 'Rosso' then
                                LCS_most_k_red(i - 1, j - 1, k)
                                print(xi)
                        else
                                if k > 0 then
                                        LCS_most_k_red(i – 1, j – 1, k – 1)
                                        print(xi)
                                if k = 0 then
                                        LCS_most_k_red(i – 1, j – 1, k – 1)
```

Capitolo 6 – Longest Common Subsequence (LCS) con ingombro minore o uguale di K

30) Input e Output del Problema

Input: $X = \langle x1, x2, x3, \ldots, xm \rangle,\ Y = \langle y1, y2, y3, \ldots, yn \rangle \colon |X| = m \ \wedge\ |Y| = n \ \wedge$ $K \in Z+.$

Sia ora la funzione
$w: \sum \rightarrow N$
Output: $|LCS_k(X, Y)| = |\langle z_1, z_2, z_3, ..., z_k \rangle|$ con ingombro al più K

31) Definizione dei sottoproblemi e variabili associate

Trovare la LCS dei prefissi X_i e Y_j con ingombro al più K $\rightarrow LCS_k(X_i, Y_j)$
$i \in \{0, 1, 2, ..., m\}$
$j \in \{0, 1, 2, ..., n\}$
$k \in \{0, 1, 2, ..., K\}$
Numero di sottoproblemi: $(m + 1) * (n + 1) * (K + 1)$
Coefficiente del sottoproblema di dimensione (i, j, k): $c_{i,j,k}$ è la lunghezza della LCS dei prefissi X_i e Y_j con ingombro minore.

32) Equazione di ricorrenza: caso base

Se $i = 0 \vee j = 0 \vee k = 0 \rightarrow c_{i,j,k} = 0$, t.c. $k \in Z^+$

33) Equazione di ricorrenza: passo ricorsivo

Se $i > 0 \wedge j > 0$, allora
se $x_i \neq y_j \rightarrow c_{i,j,k} = \max\{c_{i-1,j,k}, c_{i,j-1,k}\}$
se $x_i = y_j \wedge w(x_i) \leq k \rightarrow c_{i,j,k} = \max\{c_{i-1,j-1,k-w(xi)} + 1, c_{i-1,j-1,k-1}\}$
se $x_i = y_j \wedge w(x_i) > k \wedge k > 0 \rightarrow c_{i,j,k} = c_{i-1,j-1,k}$

34) Soluzione del problema

La soluzione di tale problema è il coefficiente $c_{m,n,K}$

35) Algoritmo Bottom Up

```
function LCS_ingombro_most_k(X, Y, K)
    m := X.length
    n := Y.length

    for k := 0 to K do
        for i := 0 to m do
            c_k[i, 0] := 0
    for k := 0 to K do
        for j := 0 to m do
            c_k[0, j] := 0
    for k := 0 to K do
        for i := 1 to m do
            for j := 1 to n do
                if x_i ≠ y_j then
                    c_k[i, j] := max{c_k[i – 1, j], c_k[i, j – 1]}
                else
                    if w(x_i) <= k then
                        c_k[i, j] := max{c_k-w(xi)[i – 1, j – 1] + 1 , c_k[i – 1, j – 1]}
                    else
                        c_k[i, j] := c_k[i – 1, j – 1]
    return c_k[m, n]
```

36) Ricostruzione dell'algoritmo

```
function LCS_ing_most_k(i, j, k)
    if i > 0 ∧ j > 0 then
        if x_i ≠ y_j then
            return max{LCS_ing_most_k(i – 1, j, k), LCS_ing_most_k(i, j – 1, k)}
        else
            if w(x_i) <= k then
                return max{LCS_ing_most_k(i – 1, j, k – w(x_i)),
                    LCS_ing_most_k(i – 1, j – 1, k – 1)}
                print(xi)
            else
                return LCS_ing_most_k(i – 1, j – 1, k)
```

Capitolo 7 – Longest Alternative Common Subsequence (LACS) con colori alternanti

37) Input e Output del Problema

Input: $X = \langle x1, x2, x3, \dots, xm \rangle,\ Y = \langle y1, y2, y3, \dots, yn \rangle$: $|X| = m \ \wedge\ |Y| = n$

Sia ora:

col: N → C = {R, N, B}

Output: $|LCS_k(X, Y)| = |\langle z_1, z_2, z_3, \dots, z_k \rangle|$ con colori alternanti.

Considerare che

$$\phi(x) = \begin{cases} \text{rosso} & x < 5 \\ \text{blu} & 5 \le x \le 10 \\ \text{verde} & x > 10 \end{cases}$$

38) Definizione dei sottoproblemi e variabili associate

Trovare la LCS dei prefissi X_i e Y_j con colori alternanti associati.
$i \in \{0, 1, 2, \dots, m\}$
$j \in \{0, 1, 2, \dots, n\}$
Numero di sottoproblemi: $(m + 1) * (n + 1)$
Coefficiente del sottoproblema di dimensione (i, j, k): $c_{i,j}$ è la lunghezza della LCS dei prefissi X_i e Y_j con con colori alternanti associati.

39) Equazione di ricorrenza: caso base

Se $i = 0 \lor j = 0 \rightarrow c_{i,j} = 0$.

40) Equazione di ricorrenza: passo ricorsivo

Se $i > 0 \land j > 0$, allora
se $x_i \neq y_j \rightarrow c_{i,j} = \max\{c_{i-1,j}, c_{i,j-1}\}$
se $x_i = y_j \rightarrow c_{i,j} = 1 + \max\{c_{h,k} \mid 1 \leq h < i, 1 \leq k < j, \phi(x_i) = \phi(x_h)\}$

41) Soluzione del problema

La soluzione di tale problema è $\max\{ci,j \mid 1 \leq i \leq m, 1 \leq j \leq n\}$

42) Algoritmo Bottom Up

```
function LACS(X, Y)
    max := 0
    for i := 1 to m do
        for j := 1 to n do
            if xi ≠ yj then
                c[i, j] := 0
            else
                temp := 0
                for h := 1 to i−1 do
                    for k := 1 to j−1 do
                        if (ϕ(xh) = ϕ(xi)) ∧ (c[h,k] > temp) then
                            temp ←c[h,k]
                c[i, j] := 1 + temp
            if c[i, j] > max then
                max := c[i,j]
    return max
```

43) Algoritmo di Stampa

```
function LACSric(i, j)
    if i = 0 ∨ j =0 ∨  xi ≠ yj then
        return 0
    else
        max ←0
        for h ←1 to i−1 do
            for k ←1 to j−1 do
                if ϕ(xh)= ϕ(xi) then
                    S ←LACSric(h,k)
        if S > max then
            max ←S
    return 1+max
```

Capitolo 8 – Weighted Interval Scheduling

44) Input e Output del Problema

Input: $A = \{1, \ldots, n\}$ con $([si, ei), vi), \forall i \in \{1, \ldots, n\}$, dove

s_i → indica il tempo di inizio dell'attività;

e_i → tale che $e_i \leq s_i$, indica il tempo di fine dell'attività (si noti che ei non appartiene all'intervallo che specifica la durata dell'attività);

v_i → indica il valore dell'attività

Due attività vengono definite compatibili se non si sovrappongono:

$[s_i, e_i) \cap [s_j, e_j) = \emptyset$

La funzione

$\mathtt{COMP} : \mathcal{P}(\{1, \ldots, n\}) \rightarrow \{True, False\}$

determina se l'insieme A contiene tutte attività compatibili

$$\mathtt{COMP}(A) = \begin{cases} True & \text{se } \forall i, j \in A \text{ con } i \neq j, [s_i, e_i) \cap [s_j, e_j) = \emptyset \\ False & \text{altrimenti} \end{cases}$$

La funzione

$\mathtt{V} : \mathcal{P}(\{1, \ldots, n\}) \rightarrow \mathbb{R}_+$

determina il valore complessivo del sottoinsieme S di attività A.

$$\mathtt{V}(A) = \begin{cases} \sum_{i \in A} v_i & \text{se } A \neq \emptyset \\ 0 & \text{se } A = \emptyset \end{cases}$$

Output: Determinare $S \subseteq \{1, \ldots, n\}$ tale che

$$\mathtt{COMP}(S) = True \wedge \mathtt{V}(S) = \max_{A \subseteq \{1, \ldots, n\} : \, \mathtt{COMP}(A) = True} \{\mathtt{V}(A)\}$$

45) Definizione dei sottoproblemi e variabili associate

In questo caso il sottoproblema i – esimo viene associato al sottoinsieme i – esimo tale che

$$\mathtt{COMP}(S) = True \wedge \mathtt{V}(S) = \max_{A \subseteq \{1, \ldots, n\} : \, \mathtt{COMP}(A) = True} \{\mathtt{V}(A)\}$$

La variabile associata è la coppia (OPT_i, S_i) tale che:

$OPT_i = V(S_i)$, ossia il valore di un sottoinsieme di $\{1, \ldots, i\}$ che contiene attività mutualmente compatibili di peso massimo S_i

Numero di sottoproblemi: $(m+1)$

46) Equazione di ricorrenza: caso base

Se $i = 0 \rightarrow S_i = \emptyset \wedge OPT_i = 0$

47) Equazione di ricorrenza: passo ricorsivo

Se $i > 0$, allora si definisce

$\psi(i) = max\{j \mid j < i \wedge comp(\{i,j\}) = \ true\}$

assumendo che $\max\{\emptyset\} = 0$.

Per determinare il passo ricorsivo si distinguono i due casi:

→ $i \notin S_i$: se l'attività i non appartiene alla soluzione, allora è sufficiente considerare l'insieme di attività $\{1,\ldots, i-1\}$ e si può dunque considerare la soluzione del sottoproblema di dimensione subito minore, ossia

$\mathtt{OPT}_i = \mathtt{OPT}_{i-1}$ e $S_i = S_{i-1}$

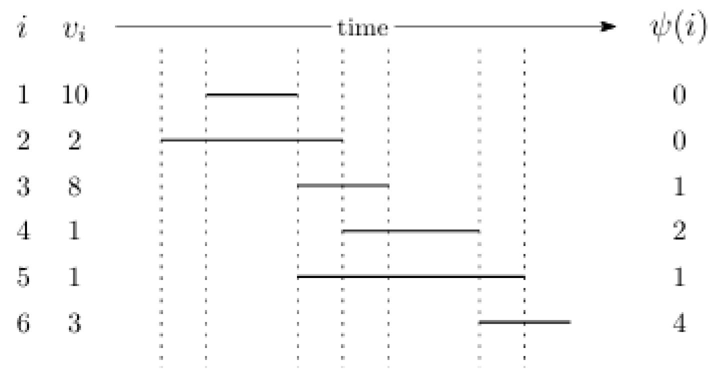
→ $i \in S_i$: se l'attività i appartiene alla soluzione, allora è necessario andare a determinare la soluzione del sottoproblema di dimensione minore ad i di peso massimo e il cui insieme di attività risulta essere compatibile con l'attività i. Risulta dunque possibile considerare la soluzione del sottoproblema di dimensione $\psi(i)$ e aggiungere a questa l'attività i. Si ottiene dunque:

$\mathtt{OPT}_i = \mathtt{OPT}_{\psi(i)} + v_i$ e $S_i = S_{\psi(i)} \cup \{i\}$

Risulta quindi

$$\mathtt{OPT}_i = \max\{\mathtt{OPT}_{i-1}, \mathtt{OPT}_{\psi(i)} + v_i\}$$

$$S_i = \begin{cases} S_{i-1} & \text{se } \mathtt{OPT}_{i-1} \geq \mathtt{OPT}_{\psi(i)} + v_i \\ S_{\psi(i)} \cup \{i\} & \text{altrimenti} \end{cases}$$



OPT							
	0	10	2	18	18	18	21
	0	*1*	*2*	*3*	*4*	*5*	*6*

$S = \{1, 3, 6\}$

48) Soluzione del problema

La soluzione di tale problema è la coppia (OPT_n, S_n)

49) Algoritmo Bottom Up

```
function WIS(n)
        OPT[0] ← 0
        S ←∅

        for i ← 1 to n do
                V1 ←OPT[i−1]
                V2 ←OPT[ψ(i)] +vi
                OPT[i] ← max{V1, V2}
                if V1 ≥ V2 then
                        S[i] ← S[i −1]
                else
                        S[i] ← S[ψ(i)] ∪ {i}

        return (OPT[n], S[n])
```

Capitolo 9 – Hateville

50) Input e Output del Problema

Input: $X = \{1, \dots, n\} \wedge d_i, \forall i \in \{1, \dots, n\}$

Output: $S \subseteq \{1, \dots, n\} : COMP(S) = true \wedge D(s) = max\{D(A), A \subseteq X_n : COMP(A)\}$

51) Definizione dei sottoproblemi e variabili associate

Dato un insieme $X = \{1, \dots, i\}$ rappresentante i primi i abitanti di Hateville, trovare un suo sottoinsieme, che rispetti la compatibilità, che massimizzi il denaro raccolto.

Input: $\{1, \dots, i\} \wedge d_j, \forall j \in \{1, \dots, j\}$

Output: $S_i \subseteq \{1, \dots, i\} : COMP(S_i) = true \wedge D(S_i) = max\{D(A), A \subseteq X_n : COMP(A)\}$

52) Equazione di ricorrenza: caso base

Se $i = 0$, allora significa che non ci sono abitanti in Hateville e quindi non c' è nessuno che può partecipare alla colletta. Se $i = 1$, allora significa che c'`e un solo abitante in Hateville e quindi non è necessario fare nessuna scelta per evitare di inserire i vicini.

Il caso base è:

$$\mathrm{OPT}_i = \begin{cases} 0 & \text{se } i = 0 \\ d_1 & \text{se } i = 1 \end{cases}$$

$$S_i = \begin{cases} \emptyset & \text{se } i = 0 \\ \{1\} & \text{se } i = 1 \end{cases}$$

53) Equazione di ricorrenza: passo ricorsivo

Il passo ricorsivo si ha per un qualunque sottoproblema di dimensione (i) con

$i > 1$, ossia quando ci sono almeno due abitanti in Hateville.
Si ha quindi:

$$\mathtt{OPT}_i = \max\{\mathtt{OPT}_{i-1}, \mathtt{OPT}_{i-2} + d_i\}$$

$$S_i = \begin{cases} S_{i-1} & \text{se } \mathtt{OPT}_{i-1} \geq \mathtt{OPT}_{i-2} + d_i \\ S_{i-2} \cup \{i\} & \text{altrimenti} \end{cases}$$

54) Soluzione del problema

La soluzione del problema é (OPT_n, S_n).

55) Algoritmo Bottom Up

```
function HatevilleRic(i)
    if i = 0 then
        return (0, Ø)
    else
        (V1, S1) := HatevilleRic(i – 1)
        (V2, S2) := HatevilleRic(i – 2)
        V2 := V2 + di
        S2 := V2 ∪ {i}
        if V1 ≥ V2 then
            return (V1, S1)
        else
            return (V2, S2)
```

56) Ricostruzione dell'algoritmo

```
function HatevilleRic(n)
    OPT[0] := 0
    S[0] := 0
    OPT[1] := d1
    S[1] := {1}
    for i := 2 to n do
        V1 := OPT[i – 1]
        V2 := OPT[i – 2] + di
        if V1 ≥ V2 then
            S[i] := S[i – 1]
            OPT[i] := V1
        else
            S[i] := S[i – 2] ∪ {i}
            OPT[i] := V2
    return (OPT[n], S[n])
```

57) Dimostrazione di WIS/Hateville

A - Nel caso $i \notin S_i$, per assurdo. Se S_{i-1} non fosse soluzione del problema $i - esimo$, allora vale $S_i \neq S_{i-1}$. Inoltre $V(S_i) > V(S_{i-1})$ per definizione.
Dato che $i \notin S_i$, allora $S_i \subseteq \{1, \dots, i-1\} = X_{i-1}$, con $COMP(S_i) = T$ per definizione. Quindi S_i è candidato soluzione di X_{i-1}, la cui soluzione è S_{i-1}.
Ma se S_{i-1} non è soluzione, allora ciò risulterebbe assurdo.

B - Nel caso $i \in S_i$, per assurdo.
Se $S_{p(i)} \cup \{i\}$ non fosse soluzione del problema $i - esimo$, allora $S_i \neq S_{p(i)} \cup \{i\}$.
Inoltre $V(S_i) > V\left(S_{p(i)}\right) + v_i$.
Allora $S_i = S' \cup \{i\}$ e $COMP(S_i) = T$, con $S' \subseteq \{1, \dots, p(i)\}$. Quindi S' è candidato soluzione per $p(i)$. Si può allora scrivere:
→ $V(S' \cup \{i\}) > V\left(S_{p(i)}\right) + v_i$
→ $V(S') + v_i > V\left(S_{p(i)}\right) + v_i$
→ $V(S') > V\left(S_{p(i)}\right)$
Ciò risulta assurdo dato che si è in contraddizione con $S_{p(i)} \cup \{i\}$ non soluzione del problema $p(i)$.

Capitolo 10 – Edit Distance

58) Definizione del problema

Date due sequenze $X = \langle x_1, \dots, x_m \rangle$ e $Y = \langle y_1, \dots, y_n \rangle$, rispettivamente di lunghezza m ed n definite su un alfabeto Σ, si determini il minimo numero di operazioni elementari che permette di trasformare X in Y.

59) Definizione dei sottoproblemi e variabili associate

Date due sequenze $X = \langle x_1, \dots, x_m \rangle$ e $Y = \langle y_1, \dots, y_n \rangle$, rispettivamente di lunghezza m ed n definite su un alfabeto Σ, si determini il minimo numero di operazioni elementari che permette di trasformare X_i in Y_j. L'equazione $\delta_{i,j}$ è il numero minimo di operazioni elementari che permette di trasformare X_i in Y_j.

60) Equazione di ricorrenza: caso base

Il caso base si ha per un qualunque sottoproblema di dimensione (i, j) con $i = 0 \wedge j = 0$, ossia quando uno dei due prefissi considerati è la sequenza vuota.

$$\delta_{i,j} = \begin{cases} 0 & \text{se } i = 0 \wedge j = 0 \\ j & \text{se } i = 0 \wedge j > 0 \\ i & \text{se } i > 0 \wedge j = 0 \end{cases}$$

61) Equazione di ricorrenza: passo ricorsivo

Il passo ricorsivo si ha per un qualunque sottoproblema di dimensione (i, j) con $i > 0 \land j > 0$, ossia quando entrambi i prefissi considerati non sono vuoti.

$$\delta_{i,j} = \min \begin{cases} \delta_{i-1,j-1} & \text{se } x_i = y_j \\ 1 + \min\{\delta_{i,j-1}, \delta_{i-1,j}, \delta_{i-1,j-1}\} & \text{se } x_i \neq y_j \end{cases}$$

62) Soluzione del problema

La soluzione del problema è $\partial_{m,n}$.

63) Algoritmo Ricorsivo

```
procedure EDRic(i, j)
    if i = 0 ∨ j = 0 then
        if i = 0 then
            return j
        else
            return i
    else
        if x_i = y_j then
            return EDRic(i − 1, j − 1)
        else
            ins ← 1 + EDRic(i, j − 1)
            del ← 1 + EDRic(i − 1, j)
            rep ← 1 + EDRic(i − 1, j − 1)
            return MIN(ins, del, rep)
```

64) Algoritmo Bottom Up

```
procedure ED(X, Y)
    for i ← 0 to m do
        δ[i, 0] ← i
    for j ← 1 to n do
        δ[0, j] ← j
    for i ← 1 to m do
        for j ← 1 to n do
            if x_i = y_j then
                δ[i, j] ← δ[i − 1, j − 1]
            else
                δ[i, j] ← 1 + MIN(δ[i, j − 1], δ[i − 1, j], δ[i − 1, j − 1])
    return δ[m, n]
```

Capitolo 11 – Interleaving

65) Definizione del problema

Date tre sequenze, definite su un alfabeto Σ:

$X = \langle x_1, \dots, x_m \rangle$, tale che $|X| = m$

$Y = \langle y_1, \dots, y_n \rangle$, tale che $|Y| = n$

$W = \langle w_1, \dots, w_{m+n} \rangle$, tale che $|W| = m + n$
stabilire se W è un interleaving di X e Y, ovvero se X e Y si possono trovare come due sottosequenze disgiunte in W.

66) Definire il coefficiente che risolve i vari sottoproblemi

Il coefficiente $s_{i,j}$ determina se W_{i+j} è interleaving dei prefissi X_i e Y_j.

67) Equazione di ricorrenza: caso base

Se $i = 0 \vee j = 0$, ossia uno dei prefissi è vuoto, allora risulta necessario considerare i seguenti casi:

→ se $i = 0 \wedge j = 0$, allora si avrebbe che W_{i+j} rappresenta la sequenza vuota e quindi vale che $s_{i,j} = True;$

→ se $i = 0 \wedge j > 0 \wedge w_j = y_j$, allora w_j è interleaving solo se lo è anche $W_{i,j-1}$ e quindi $s_{i,j} = s_{i,j-1};$

→ se $i = 0 \wedge j > 0 \wedge w_j \neq y_j$, allora w_j non può essere interleaving e quindi $s_{i,j} = False;$

→ se $i = 0 \wedge j > 0 \wedge w_i \neq x_i$, allora w_j non può essere interleaving e quindi $s_{i,j} = False;$

→ se $i = 0 \wedge j > 0 \wedge w_i = x_i$, allora w_j è interleaving solo se lo è anche $W_{i-1,j}$ e quindi $s_{i,j} = s_{i-1,j};$

Riassumendo si avrebbe:

$$s_{i,j} = \begin{cases} True & \text{se } i = 0 \wedge j = 0 \\ s_{i,j-1} & \text{se } i = 0 \wedge j > 0 \wedge w_j = y_j \\ False & \text{se } i = 0 \wedge j > 0 \wedge w_j \neq y_j \\ s_{i-1,j} & \text{se } i > 0 \wedge j = 0 \wedge w_i = x_i \\ False & \text{se } i > 0 \wedge j = 0 \wedge w_i \neq x_i \end{cases}$$

68) Equazione di ricorrenza: passo ricorsivo

Se $i > 0 \wedge j > 0$, ossia nessuno dei due prefissi è vuoto, allora risulta necessario considerare i seguenti casi:

→ se $w_{i+j} \neq x_i \wedge w_{i+j} \neq y_j$, allora w_{i+j} non può essere interleaving e quindi $s_{i,j} = False;$

→ se $w_{i+j} = x_i \wedge w_{i+j} \neq y_j$, allora w_{i+j} è interleaving solo se lo è anche $W_{i-1,j}$ e quindi $s_{i,j} = s_{i-1,j};$

→ se $w_{i+j} \neq x_i \wedge w_{i+j} = y_j$, allora w_{i+j} è interleaving solo se lo è anche $W_{i,j-1}$ e quindi $s_{i,j} = s_{i,j-1};$

→ se $w_{i+j} = x_i \wedge w_{i+j} = y_j$, allora w_{i+j} è interleaving solo se lo è $W_{i-1,j}$ e $W_{i,j-1}$ e quindi $s_{i,j} = s_{i-1,j} \vee s_{i,j-1}$

$$s_{i,j} = \begin{cases} False & \text{se } w_{i+j} \neq x_i \wedge w_{i+j} \neq y_j \\ s_{i-1,j} & \text{se } w_{i+j} = x_i \wedge w_{i+j} \neq y_j \\ s_{i,j-1} & \text{se } w_{i+j} \neq x_i \wedge w_{i+j} = y_j \\ s_{i-1,j} \vee s_{i,j-1} & \text{se } w_{i+j} = x_i \wedge w_{i+j} = y_j \end{cases}$$

69) Soluzione del problema

La soluzione del problema è $s_{m,n}$.

Capitolo 12 – Knapsack

70) Input e Output del Problema

Input: $X = \{1, \dots, n\}: (v_i, w_i), \forall i \in \{1, \dots, n\}$

Output: $S \subseteq \{1, \dots, n\}: W(S) \leq C \wedge V(S) = max_{A \subseteq \{1,\dots,n\}:\, W(A) \leq C}\{V(A)\}$

71) Definizione dei sottoproblemi e variabili associate

Dato un insieme $\{1, \dots, i\}$ di oggetti, trovare un suo sottoinsieme di ingombro complessivo $\leq C$ e di massimo valore complessivo.

Si ha quindi:

Input: $X_i = \{1, \dots, i\}: (v_j, w_j), \forall j \in \{1, \dots, i\}$

Output: $S \subseteq \{1, \dots, i\}: W(S_{i,c}) \leq c \wedge V(S_{i,c}) = max_{A \subseteq \{1,\dots,i\}:\, W(A) \leq c}\{V(A)\}$

Dato che $0 \leq i \leq n$ si ottengono $(n+1)$ sottoproblemi, ad ognuno dei quali è associata una coppia di variabili. Si ha quindi (OPT_i, S_i), dove:

$OPT_{i,c} = V(S_{i,c})$, ossia il valore di un sottoinsieme di $\{1, \dots, i\}$ di ingombro complessivo $\leq c$ e di massimo valore complessivo.

72) Equazione di ricorrenza: caso base

Se $i = 0 \vee c = 0$, allora vale

$OPT_{i,c} = 0 \wedge S_{i,c} = \emptyset$ se $i = 0 \vee c = 0$

73) Equazione di ricorrenza: passo ricorsivo

Quando $i > 0 \wedge c > 0$, allora risulta necessario considerare i seguenti casi:

→ $w_i > c$: se l'oggetto ha ingombro maggiore della capacità, allora l'oggetto non può appartenere alla soluzione e quindi si va a considerare la soluzione del sottoproblema di dimensione minore $i-1, c$. In questo caso

$OPT_{i,c} = 0PT_{i-1,c} \wedge S_{i,c} = S_{i-1,c}$ se $w_i > c$

→ $w_i \leq c$: allora risulta necessario tenere in considerazione due casi:

a. $i \notin S_i$: se l'oggetto i non appartiene alla soluzione allora risulta necessario considerare l'insieme $\{1, \dots, i-1\}$ e quindi:

$OPT_{i,c} = OPT_{i-1,c} \wedge S_{i,c} = S_{i-1,c}$ se $i \notin S_i$

b. $i \in S_i$: se l'oggetto i appartiene alla soluzione allora vale:

$OPT_{i,c} = OPT_{i-1,c-w_i} + v_i \wedge S_{i,c} = S_{i-1,c-w_i} \cup \{i\}$ se $i \in S_i$

Riassumendo si ha:

$$\mathtt{OPT}_{i,c} = \begin{cases} \mathtt{OPT}_{i-1,c} & \text{se } w_i > c \\ \max\{\mathtt{OPT}_{i-1,c}, \mathtt{OPT}_{i-1,c-w_i} + v_i\} & \text{altrimenti} \end{cases}$$

e

$$S_{i,c} = \begin{cases} S_{i-1,c} & \text{se } w_i > c \\ \begin{cases} S_{i-1,c} & \text{se } \mathtt{OPT}_{i-1,c} \geq \mathtt{OPT}_{i-1,c-w_i} + v_i \\ S_{i-1,c-w_i} \cup \{i\} & \text{altrimenti} \end{cases} & \text{altrimenti} \end{cases}$$

74) Soluzione del problema

La soluzione del problema è $(OPT_{n,C}, S_{n,C})$.

75) Algoritmo Bottom Up

```
procedure KP(n, C)
    for i ← 0 to n do
        OPT[i, 0] ← 0
        S[i, 0] ← ∅
    for c ← 0 to C do
        OPT[0, c] ← 0
        S[0, c] ← ∅
    for i ← 1 to n do
        for c ← 1 to C do
            if w_i > c then
                OPT[i, c] ← OPT[i − 1, c]
                S[i, c] ← S[i − 1, c]
            else
                V_1 ← OPT[i − 1, c]
                V_2 ← OPT[i − 1, c − w_i] + v_i
                OPT[i, c] ← MAX(V_1, V_2)
                if V_1 ≥ V_2 then
                    S[i, c] ← S[i − 1, c]
                else
                    S[i, c] ← S[i − 1, c − w_i] ∪ {i}
    return (OPT[n, C], S[n, C])
```

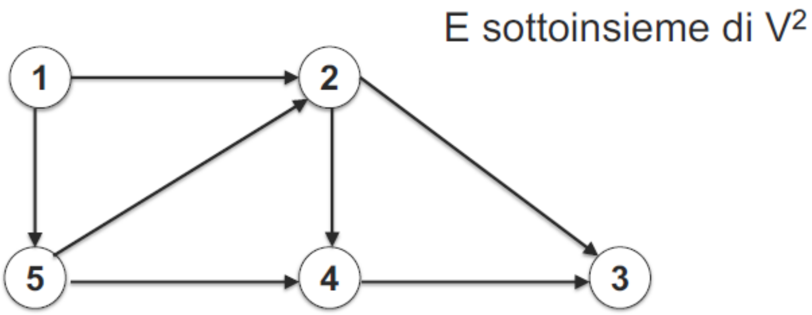
76) Algoritmo Ricorsivo

```
procedure KPRIC(i,c)
    if i = 0 ∨ c = 0 then
        return (0, ∅)
    else
        if w_i > c then
            return KPRIC(i − 1, c)
        else
            (V_1, S_1) ← KPRIC(i − 1, c)
            (V_2, S_2) ← KPRIC(i − 1, c − w_i)
            V_2 ← V_2 + v_i
            if V_1 ≥ V_2 then
                return (V_1, S_1)
            else
                return (V_2, S_2 ∪ {i})
```

Capitolo 13 – Grafi: Floyd – Warshall

77) Input e Output del Problema

Input: Si ha un grafo $G = (V, E, W)$ (senza cappi) orientato e pesato tale che:
→ $V = \{v_1, v_2, \dots, v_n\}$
→ $E \subseteq V^2$
→ $W: E \rightarrow R^+ : w(i, j) = w_{ij}$



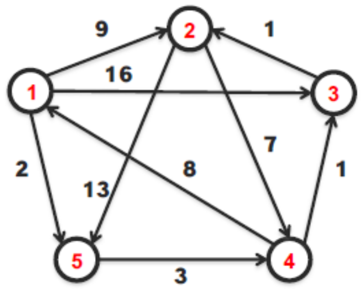
V = {1, 2, 3, 4, 5}

E = {(1,2), (1,5), (2,3), (2,4), (4,3), (5,2), (5,4)}

Output: Per ogni coppia di vertici i e j, trovare il cammino di peso minimo (cammino minimo) che parte da i e finisce in j.

78) Definizione dei sottoproblemi e variabili associate

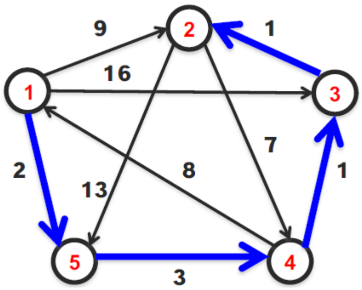
Il coefficiente $d_{i,j}^k$ determina il peso del cammino minimo dal vertice i al vertice j, con vertici intermedi $\in \{1, \dots, k\}$ se $k > 0$ senza vertici intermedi se $k = 0$.
Vale quindi $k \in \{0, 1, \dots, n\}, i \in \{1, \dots, n\}, j \in \{1, \dots, n\}$
Qui si mostra la matrice W dei pesi



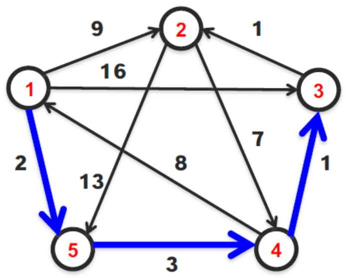
n=5

W	1	2	3	4	5
1	0	9	16	∞	2
2	∞	0	∞	7	13
3	∞	1	0	∞	∞
4	8	∞	1	0	∞
5	∞	∞	∞	3	0

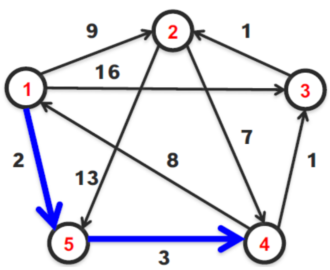
Qui si mostra alcuni passi della costruzione della matrice π dei predecessori



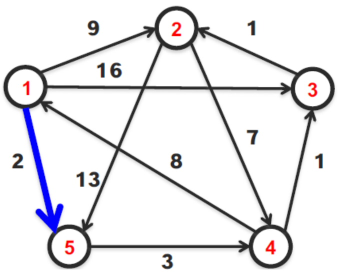
Π	1	2	3	4	5
1	NIL				
2					
3					
4					
5					



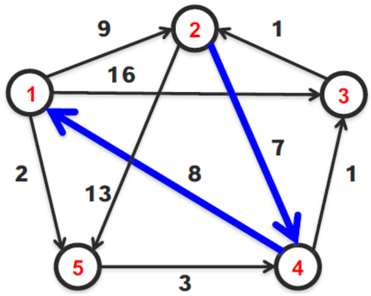
Π	1	2	3	4	5
1	NIL	3	4		
2					
3					
4					
5					



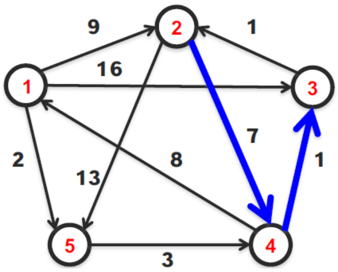
Π	1	2	3	4	5
1	NIL	3	4	5	
2					
3					
4					
5					



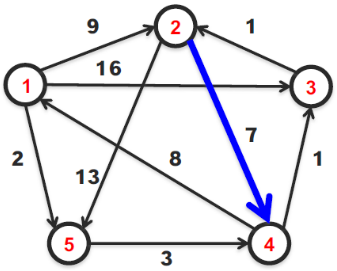
Π	1	2	3	4	5
1	NIL	3	4	5	1
2					
3					
4					
5					



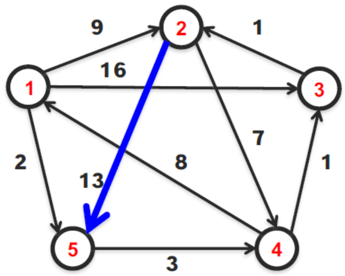
Π	1	2	3	4	5
1	NIL	3	4	5	1
2	4	NIL			
3					
4					
5					



Π	1	2	3	4	5
1	NIL	3	4	5	1
2	4	NIL	4		
3					
4					
5					

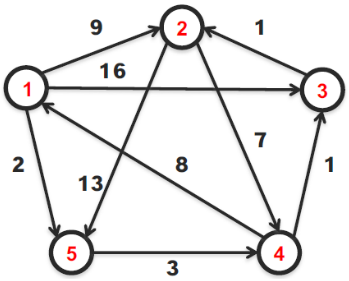


Π	1	2	3	4	5
1	NIL	3	4	5	1
2	4	NIL	4	2	
3					
4					
5					



Π	1	2	3	4	5
1	NIL	3	4	5	1
2	4	NIL	4	2	2
3					
4					
5					

Alla fine si ottiene



Π	1	2	3	4	5
1	NIL	3	4	5	1
2	4	NIL	4	2	2
3	4	3	NIL	2	2
4	4	3	4	NIL	1
5	4	3	4	5	NIL

79) Equazione di ricorrenza: caso base

Se $k = 0$, allora:

→ $d_{i,j}^{0} = 0$, se $i = j$;

→ $d_{i,j}^{0} = w_{ij}$, se $i \neq j \wedge (i,j) \in E$

→ $d_{i,j}^{0} = \infty$, se $i \neq j \wedge (i,j) \notin E$

Riassumendo

$$d_{i,j}^{0} = \begin{cases} 0 \\ w_{ij} \\ \infty \end{cases}$$

Per quanto riguarda i predecessori valgono le seguenti condizioni:

→ $\pi_{i,j}^{0} = NIL$, se $i = j$;

→ $\pi_{i,j}^{0} = i$, se $i \neq j \wedge (i,j) \in E$

→ $\pi_{i,j}^{0} = NIL$, se $i \neq j \wedge (i,j) \notin E$

Riassumendo

$$\pi_{i,j}^{0} = \begin{cases} NIL \\ i \\ NIL \end{cases}$$

80) Equazione di ricorrenza: passo ricorsivo

Se $k > 0$, allora vale che:

→ se $k \notin p$, allora

$d_{i,j}^{k} = d_{i,j}^{k-1}$ e $\pi_{i,j}^{k} = \pi_{i,j}^{k-1}$

→ se $k \in p$, allora

$d_{i,j}^{k} = d_{i,k}^{k-1} + d_{k,j}^{k-1}$ e $\pi_{i,j}^{k} = \pi_{k,j}^{k-1}$

Riassumendo

$$d_{i,j}^{k} = min\{d_{i,j}^{k-1}, d_{i,k}^{k-1} + d_{k,j}^{k-1}\}$$

$$\pi_{i,j}^{k} = \begin{cases} \pi_{i,j}^{k-1} \\ \pi_{k,j}^{k-1} \end{cases}$$

81) Soluzione del problema

Le soluzioni del problema sono $d_{i,j}^{n}$ e $\pi_{i,j}^{n}$

81) Algoritmo Bottom Up

```
Procedura calcola_valori_ottimi_FW(V,E,W)
    D⁰ ← W
    Π⁰ ← (n x n) matrix of NIL values
    for i ← 1 to n do
        for j ← 1 to n do
            if i ≠ j and wij ≠ ∞ then
                Π⁰[i,j] ← i
    for k ← 1 to n do
        for i ← 1 to n do
            for j ← 1 to n do
                Dᵏ[i,j] ← Dᵏ⁻¹[i,j]
                Πᵏ[i,j] ← Πᵏ⁻¹[i,j]
                if i ≠ k and j ≠ k then
                    if Dᵏ[i,j] > Dᵏ⁻¹[i,k] + Dᵏ⁻¹[k,j] then
                        Dᵏ[i,j] ← Dᵏ⁻¹[i,k] + Dᵏ⁻¹[k,j]
                        Πᵏ[i,j] ← Πᵏ⁻¹[k,j]
```

Complessità Computazionale e Spazio di memoria

Sia $|V| = n$

$T(n) = \Theta(n^3)$ dato che il numero di sottoproblemi è pari a $n * n * (n+1)$.

Riguardo lo spazio in memoria si hanno $2 * (n+1)$ matrici e quindi lo spazio occupato è pari a:

$2 * (n+1) * n^2 = 2 * (n^3 + n^2) = 2n^3 + n^2 = \Theta(n^3)$

82) Descrivere la struttura di un cammino minimo di Floyd Warshall

L'algoritmo considera vertici intermedi di un cammino minimo dove un vertice intermedio di un cammino semplice $p = < v_1, \dots, v_l >$ è un qualunque vertice di p diverso da v_1 e v_l, ossia un qualunque vertice appartenente a $< v_2, \dots, v_{l-1} >$.

Sia V l'insieme dei vertici, sia $\{1, \dots, k\} \subseteq V$ per un qualche k. Lo scopo è determinare il peso del cammino minimo dal vertice i al vertice j, con vertici intermedi $\in \{1, \dots, k\}$ con $k > 0$, per ogni $(i, j) \in V^2$. L'algoritmo di Floyd Warshall effettua la seguente considerazione:

→ Se $k \notin p$, allora tutti i vertici intermedi sul cammino p sono nell'insieme $\{1, \dots, k-1\}$. Quindi un cammino minimo da i a j con tutti i vertici intermedi nell'insieme $\{1, \dots, k-1\}$ è anche un cammino minimo da i a j con tutti i vertici intermedi nell'insieme $\{1, \dots, k\}$.

→ Se $k \in p$, allora il cammino p viene scomposto in due sottocammini p_1 e p_2.

Dato il seguente lemma:

Sia un $G = (V, E, W)$ grafo (senza cappi) orientato e pesato con funzione $W: E \to R^+ : w(i,j) = w_{ij}$. Sia $p = < v_1, \dots, v_k >$ un cammino minimo dal vertice v_1 al vertice v_k. Per ogni $(i, j) \in V^2$ tale che $1 \le i \le j \le k$, sia $p_{ij} = < v_i, v_{i+1} \dots, v_j >$ il sottocammino di p dal vertice v_i al vertice v_j. Allora p_{ij} è un cammino minimo da v_i al vertice v_j.

Il cammino p_1 è minimo da i a k con tutti i vertici intermedi nell'insieme $\{1, \dots, k-1\}$ e, analogamente, il cammino p_2 è minimo da k a j con tutti i vertici intermedi nell'insieme $\{1, \dots, k-1\}$.

83) Chiusura transitiva di un grafo orientato

Dato un grafo orientato $G = (V, E)$ con $V = \{v_1, v_2, \dots, v_n\}$, si definisce chiusura transitiva di un grafo orientato $G = (V, E)$ tale che:

$E = \{(i,j): esiste\ un\ cammino\ in\ G\ dal\ vertice\ i\ al\ vertice\ j\}$

Una modalità utilizzata per calcolare la chiusura transitiva di un grafo G in tempo $\theta(n^3)$ e che permette il risparmio di spazio e tempo consiste nel sostituire gli operatori logici $\wedge$ e $\vee$ con min e $+$ nell'algoritmo di Floyd Warshall.

Vale quindi:

$$t_{ij}^{(0)} = \begin{cases} 0 & \text{se } i \neq j \text{ e } (i,j) \notin E \\ 1 & \text{se } i = j \text{ o } (i,j) \in E \end{cases}$$

Con $k > 0$ vale
$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}\right)$
Si ha il seguente pseudocodice:

```
function FW_transitive_closure(V, E)
      n := |V|

      for i := 1 to n do
            for j := 1 to n do
                  if i = j ∨ (i, j) ∈ E then
                        t^0_{i,j} = 1
                  else
                        t^0_{i,j} = 0
      for k := 1 to n do
            for i := 1 to n do
                  for j := 1 to n do
                        t^k_{i,j} = t^{k-1}_{i,j} ∨ (t^{k-1}_{i,k} ∧ d^{k-1}_{k,j})
      return T^(n)
```