

Definizione: Sia  $A = \{s_1, s_2, \dots, s_n\}$  n attività. Ad ogni attività  $i \in A$  associamo:

- $s_i$ : tempo inizio
- $e_i$ : fine attività
- $v_i$ : valore attività

Due attività sono dette compatibili se  $[s_i, e_i] \cap [s_j, e_j] = \emptyset$  (non si intersecano)

Un insieme  $A$  di attività mutualmente compatibili sse  $\forall i, j \in A$  con  $i \neq j$   $[s_i, e_i] \cap [s_j, e_j] = \emptyset$

Definisco  $\text{COMP}: P(\{s_1, s_2, \dots, s_n\}) \rightarrow \{\text{true}, \text{false}\}$  e  $V: P(\{s_1, s_2, \dots, s_n\}) \rightarrow \mathbb{R}$ , tc  $V(A) = \begin{cases} \sum_{i \in A} v_i & \text{se } A \neq \emptyset \\ 0 & \text{se } A = \emptyset \end{cases}$

Problema: Dato un insieme  $\{s_1, s_2, \dots, s_n\}$  di attività, trovare un suo sottinsieme di attività mutualmente compatibili di valore massimo

Sottoproblema  $i$ -esimo: Dato un insieme  $\{s_1, s_2, \dots, s_i\}$  di attività, trovare un suo sottinsieme di attività mutualmente compatibili di valore massimo

Numero sottoproblemi: Dato che  $0 \leq i \leq n$  si ottengono  $(n+1)$  sottoproblemi, ad ognuno dei quali è associata una coppia di valori  $(\text{OPT}_i, S_i)$  dove  $\text{OPT}_i$  contiene il valore ottimo  $i$ -esimo

Equazioni di ricorrenza:

caso base:  $i=0$  avremo  $\text{OPT}_0$  e di conseguenza  $S_0 = \emptyset$

passo ricorsivo: definiamo  $\psi(i) = \max \{j \mid j < i \wedge j \text{ compatibile con } i\}$  allora posso trovarmi in due casi, sceglierò il massimo tra i due.

$i \notin S_i$  posso considerare  $\text{OPT}_i = \text{OPT}_{i-1}$  e  $S_i = S_{i-1}$

$i \in S_i$  posso considerare  $\text{OPT}_i = \text{OPT}_{\psi(i)} + v_i$   $S_i = S_{\psi(i)} \cup \{i\}$

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

Algoritmo bottom-up: restituisce valore ottimo e soluzione ottimale in tempo  $\Theta(n)$  e spazio  $\Theta(n \cdot n^2)$

PROCEDURE WIS( $n$ )

$$\text{OPT}[0] = 0, S = \emptyset$$

for  $i = 1$  to  $n$

$$V_1 = \text{OPT}[i-1]$$

$$V_2 = \text{OPT}[\psi(i)] + v_i$$

$$\text{OPT}[i] = \max\{V_1, V_2\}$$

if  $V_1 \geq V_2$

$$S[i] = S[i-1]$$

else

$$S[i] = S[\psi(i)] \cup \{i\}$$

return ( $\text{OPT}(n), S[n]$ )

HATEVILLE. Abbiamo un insieme di case che donano. Se la casa  $i$  dona, le case  $i-1$  e  $i+1$  non possono far parte della donazione totale. Trovare la massima donazione.

ESEMPIO:

1	2	3	4	5	6	7
3	10	1	2	10	13	7

 $S = \{2, 5, 7\}$   $D(S) = 27$ 

ISTANZA:  $n \in \mathbb{N}$   $X_n = \{1, \dots, n\}$

SOLUZIONE:  $S \subseteq X_n$  t.c.  $\text{comp}(S) = \text{true} \wedge D(S) = \max\{D(A)\} \mid A \subseteq X_n \wedge \text{comp}(A) = \text{true}$

COMP:  $\mathcal{P}(X_n) \rightarrow \{\text{true}, \text{false}\} \mid \forall A \in \mathcal{P}(X_n), \text{comp}(A) = \text{true} \iff \forall i \in A, i+1 \notin A \wedge i-1 \notin A$

$$D: \mathcal{P}(X_n) \rightarrow \mathbb{R} \mid \forall A \in \mathcal{P}(X_n), D(A) = \sum_{i \in A} d_i \quad (D(\emptyset) = 0)$$

SOTTOPROBLEMA  $i$ -mo:

$\cdot X_i = \{1, \dots, i\}$   $S_i \subseteq X_i$  se  $\text{comp}(S_i) = \text{true} \wedge D(S_i) = \max\{D(A)\} \mid A \subseteq X_i \wedge \text{comp}(A) = \text{true}$

$\cdot X_{i-1} = \{1, \dots, i-1\}$   $S_{i-1} \subseteq X_{i-1}$  se  $\text{comp}(S_{i-1}) = \text{true} \wedge D(S_{i-1}) = \max\{D(A)\} \mid A \subseteq X_{i-1} \wedge \text{comp}(A) = \text{true}$

$\vdots$

$\cdot X_1 = \{1\}$   $S_1 = \{1\}$   $D(S_1) = d_1$

$\cdot X_0 = \emptyset$   $S_0 = \emptyset$   $D(S_0) = 0$

TEOREMA: Sia  $i \geq 2$ . Assumiamo di aver risolto i sottoproblemi  $S_0, S_1, \dots, S_{i-2}$ . Allora la soluzione del sottoproblema  $i$  è:

$$S_i = \begin{cases} S_{i-1} & \text{se } i \notin S_i \\ S_{i-2} \cup \{i\} & \text{se } i \in S_i \end{cases}$$

Dimostrazione:

$\cdot i \notin S_i$ . Per assurdo  $S_i \neq S_{i-1}$ . Allora  $D(S_i) > D(S_{i-1})$ . Siccome  $i \notin S_i$ ,  $S_i \subseteq \{1, \dots, i-1\}$  e  $\text{comp}(S_i) = \text{true}$ .

Ma dato che se  $\text{comp}(S_i)$  allora  $\text{comp}(S_{i-1})$  arriviamo all'assurdo.

$\cdot i \in S_i$ . Per assurdo  $S_i \neq S_{i-2} \cup \{i\}$ . Allora  $D(S_i) > D(S_{i-2} \cup \{i\})$ . Siccome  $i \in S_i$  e  $\text{comp}(S_i) = \text{true}$

allora  $S_i = S' \cup \{i\}$  con  $S' \subseteq \{1, \dots, i-2\}$ . Allora  $D(S' \cup \{i\}) > D(S_{i-2} \cup \{i\})$ . Assurdo.

# Interleaving

INTERLEAVING: Date 3 sequenze  $X = \langle x_1, \dots, x_m \rangle$ ,  $Y = \langle y_1, \dots, y_n \rangle$ ,  $W = \langle w_1, \dots, w_{m+n} \rangle$  si vuole stabilire se  $W$  è un interleaving di  $X$  e  $Y$ , ossia se  $X$  e  $Y$  si possono trovare come due sottosequenze disgiunte in  $W$ .

$\forall (i, j)$  con  $i \in \{1, \dots, m\}$  e  $j \in \{1, \dots, n\}$  definiamo  $S_{i,j}$  che vale true se  $w_{i+j}$  è interleaving di  $x_i$  e  $y_j$

CALCOLO DI  $S_{i,j}$ : Supponiamo di aver già risolto tutti i sottoproblemi  $(h, k)$  con  $h \in \{0, \dots, i\}$  e  $k \in \{0, \dots, j\}$  e  $(h \neq k), (i, j)$ . Si presentano i seguenti casi:

$$\cdot i=0 \wedge j=0 \rightarrow S_{i,j} = \text{true} \quad [\text{CASO BASE}]$$

$$\cdot i=0 \wedge j>0$$

$$\cdot \text{Se } w_j \neq y_j \rightarrow S_{i,j} = \text{false} \quad [\text{CASO BASE}]$$

$$\cdot \text{Se } w_j = y_j \rightarrow S_{i,j} = S_{i,j-1}$$

$$\cdot i>0 \wedge j=0$$

$$\cdot \text{Se } w_i \neq y_i \rightarrow S_{i,j} = \text{false} \quad [\text{CASO BASE}]$$

$$\cdot \text{Se } w_i = y_i \rightarrow S_{i,j} = S_{i-1,j}$$

$$\cdot i>0 \wedge j>0$$

$$\cdot \text{Se } w_{i+j} \neq x_i \wedge w_{i+j} \neq y_j \rightarrow S_{i,j} = \text{false} \quad [\text{CASO BASE}]$$

$$\cdot \text{Se } w_{i+j} = x_i \wedge w_{i+j} \neq y_j \rightarrow S_{i,j} = S_{i-1,j}$$

$$\cdot \text{Se } w_{i+j} \neq x_i \wedge w_{i+j} = y_j \rightarrow S_{i,j} = S_{i,j-1}$$

$$\cdot \text{Se } w_{i+j} = x_i \wedge w_{i+j} = y_j \rightarrow S_{i,j} = S_{i-1,j} \vee S_{i,j-1}$$

ESEMPIO:  $X = \langle \text{CIAO} \rangle$ ,  $Y = \langle \text{MAMMA} \rangle$ ,  $W_1 = \langle \text{CIMAAAMMAO} \rangle$ ,  $W_2 = \langle \text{CIMAMMAE} \rangle$

$W_1$ :		M	A	M	M	A	
	T						
C	T						
I	T	T	T				
A		T	T	T	T	T	
O						T	

Sol: True

$W_2$ :		M	A	M	M	A	
	T						
C	T						
I	T	T	T				
A		T	T	T	T	T	
O							F

Sol: False

PALINDROMA: Sia  $S = \alpha_1 \dots \alpha_n$  una stringa di lunghezza  $n$  su alfabeto  $\Sigma$ . Si vuole determinare il numero minimo di caratteri da aggiungere ad  $S$  per renderla palindroma.

ISTANZA  $S \in \Sigma^*$

SOLUZIONE  $f(S)$  tc  $f: \Sigma^* \rightarrow \mathbb{N}$

CALCOLO DI  $S$

$\cdot S = \epsilon \rightarrow S$  è palindroma  $\rightarrow f(S) = 0$  [CASO BASE]

$\cdot S = \alpha \rightarrow S$  è palindroma  $\rightarrow f(S) = 0$  [CASO BASE]

$\cdot S = \alpha S' b$  dove  $\alpha$  è il carattere iniziale,  $b$  quello finale, e  $S'$  stringa con  $|S'| < |S|$ .

$\cdot \alpha = b \rightarrow S$  palindroma sse  $S'$  palindroma

$\cdot \alpha \neq b \rightarrow S = \alpha S' b \alpha \wedge f(S) = f(S') + 1 \vee S = b \alpha S' b \alpha \wedge f(S) = f(\alpha S') + 1$

SOTTOPROBLEMA: Se  $\alpha \neq b$  la ricchezza sarà  $f(S) = \min\{f(S'), f(\alpha S')\} + 1$ . Denoto le due sottostringhe con una generica  $S_{i,j} = \alpha_i \dots \alpha_j$  con  $i \in \{1, \dots, n\}$  e  $j \in \{1, \dots, n\}$ :

$\cdot i > j \rightarrow S_{i,j} = \epsilon$  [CASO BASE]

$\cdot i = j \rightarrow S_{i,j} = \alpha_i$  [CASO BASE]

ISTANZA ORIGINALE:  $S = S_{1,n}$

SOLUZIONE ORIGINALE:  $f(S) = f(S_n)$

ISTANZA SOTTOPROB:  $S = S_{i,j}$

SOLUZIONE SOTTOPROB:  $f(S) = f(S_{i,j})$

Introduciamo la variabile  $m_{i,j} = f(S_{i,j}) \forall (i,j)$ . Avremo quindi:

CASO BASE:  $\forall (i,j)$  tc  $i \geq j \rightarrow m_{i,j} = 0$

PASSO RICORSIVO:  $\forall (i,j)$  tc  $i < j$  avremo:

$\cdot m_{i,j} = m_{i+1,j-1}$  se  $\alpha_i = \alpha_j$

$\cdot m_{i,j} = \min\{m_{i+1,j}, m_{i,j-1}\} + 1$  se  $\alpha_i \neq \alpha_j$

ESEMPIO:  $S = \text{"casa"}$

	1	2	3	4
1	0	i. 1	ii. 2	iv. 3
2	0	0	iii. 1	2
3	0	0	0	1
4	0	0	0	0

- i.  $C[1][2] = \text{"ca"} \rightarrow +c \rightarrow \text{cac} \rightarrow +1$
- ii.  $C[1][3] = \text{"cas"} \rightarrow +ac \rightarrow \text{casac} \rightarrow +2$
- iii.  $C[2][3] = \text{"as"} \rightarrow +a \rightarrow \text{osa} \rightarrow +1$
- iv.  $C[1][4] = \text{"casa"} \rightarrow +c \rightarrow \text{casac} \rightarrow +1$

ALGORITMO:

$n = \text{length}(S)$

for  $j=1$  to  $n$

    for  $i=j$  to  $n$

$m_{i,j} = 0$

        for  $i=n-1$  down to  $1$

            for  $j=i+1$  to  $n$

                if  $a_i = a_j$

$m_{i,j} = m_{i+1,j-1}$

                else

$m_{i,j} = \min \{ m_{i+1,j}, m_{i,j+1} \}$

# Sottovettore valore massimo

Definizione: Abbiamo un vettore  $V = \{v_1, \dots, v_n\}$  con  $n$  elementi. Trovare il sottovettore, ovvero il vettore senza salti di posizione, di valore massimo.

Sottoproblema  $i$ -esimo: sottovettore  $V_i: (v_k, \dots, v_l)$  con  $1 \leq k \leq l \leq i$ .

$i \notin S_i: S_{i-1} - S_{i-1}$

$i \in S_i: S_i - S'$   $v_i$  con  $S'$  soluzione di uno dei sottoproblemi più piccoli. Problema ausiliario

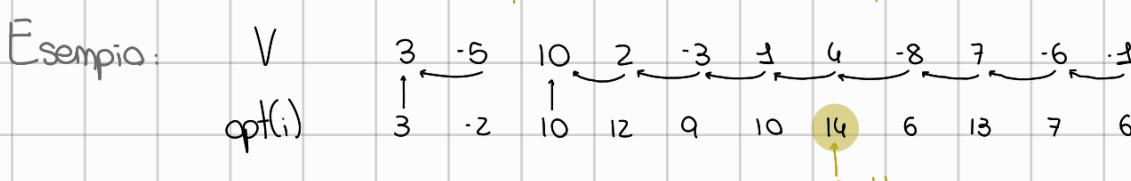
Problema ausiliario:  $S$  sottovettore di  $V$  che termina con  $v_n$  tc  $w(S) = \max\{w(A)\}$  con  $A$  sottovettore di  $V$  che termina con  $v_n$

Sottoproblema ausiliario:  $\text{opt}(i) = w(S_i)$  dove  $S_i$  sottovettore di  $(v_1, \dots, v_i)$  che termina con  $v_i$  tc  $w(S_i) = \max\{w(A)\}$  A sottovettore  $(v_1, \dots, v_i)$  che termina con  $v_i$ .

Equazioni di ricorrenza:

caso base:  $i=1$  allora  $\text{opt}(1) = v_1$

passo ricorsivo:  $i > 1$  allora  $\text{opt}(i) = \max\{\text{opt}(i-1) + v_i, v_i\}$



Soluzione ridotta: scorro  $\text{opt}(i)$  e mi salvo il massimo.

Soluzione originale: parto da max e scorro fino a quando trovo che  $\text{opt}(i) = v(i)$ .

# Costo minimo percorso e n° percorsi

Istanza: tabella  $m \times n$  con  $V(i,j)$   $i \in \{1, \dots, m\}, j \in \{1, \dots, n\}$ .  $C_{i,j} > 0 \in \mathbb{R}$

Soluzione: Partire da  $(0,0)$  e arrivare a  $(m,n)$  con percorso costo minore.

Definizione sottoproblema:  $c_{i,j}$  è il costo della mossa per spostarsi alla casella  $(i,j)$  da una delle 3 possibilità:  $\{(i,j-1), (i-1,j), (i-1,j-1)\}$

Equazioni ricorrenza:

caso base:  $S_{0,0} = 0$

passo ricorsivo:  $S_{i,j} = \min\{S_{i-1,j} + C_{i-1,j}, S_{i,j-1} + C_{i,j-1}, S_{i-1,j-1} + C_{i-1,j-1}\}$

---

Istanza: tabella  $m \times n$  con  $V(i,j)$   $i \in \{1, \dots, m\}, j \in \{1, \dots, n\}$ .  $C_{i,j} > 0 \in \mathbb{R}$

Soluzione: n° cammini da  $(0,0)$  a  $(m,n)$

Definizione sottoproblema:  $c_{i,j}$  è il numero di cammini per spostarsi alla casella  $(i,j)$  da una delle 3 possibilità:  $\{(i,j-1), (i-1,j), (i-1,j-1)\}$

Equazioni ricorrenza:

caso base:  $C_{i,j} = 1$

se  $i=1 \vee j=1$

passo ricorsivo:  $C_{i,j} = C_{i-1,j} + C_{i,j-1} + C_{i-1,j-1}$

altrimenti:

# Knapsack a coppie

8

Istanza:  $X_m = \{x_1, \dots, x_m\}$  insieme con  $m$  elementi,  $Y_n = \{y_1, \dots, y_n\}$  insieme  $n$  elementi

$$w: \Gamma \rightarrow N^+ \quad v: \Gamma \rightarrow N^+ \quad tc \quad \Gamma = X_m \cup Y_n \mid X_m \cap Y_n = \emptyset$$

$C \in N^+$  capacità zaino

Soluzione:  $S \subseteq X_m \times Y_n \in V(S) = \max_{A \subseteq X_m \times Y_n \in W(A) \leq C} \{V(A)\}$

(miglior coppia e coppie non ci sono elementi ripetuti.)

Sottoproblemi: dato  $X_i$  con  $0 \leq i \leq m$  e  $Y_j$  con  $0 \leq j \leq n$  e  $c$  con  $0 \leq c \leq C$  vogliamo

$S_{i,j,c} \subseteq X_i \times Y_j$  e  $S_{i,j,c}$  s.t.  $\text{opt}(i, j, c) = V(S_{i,j,c})$

Equazioni di ricorrenza:

caso base:  $S_{i,j,c} = \emptyset$  se  $(i=0 \vee j=0) \vee c=0$

$$\text{passo ricorsivo: } S_{i,j,c} = \begin{cases} S_{i-1,j-1,c} - w(i) - w(j) \cup \{(x_i, y_j)\} & \text{se } (x_i, y_j) \in S_{i,j,c} \\ S_{i-1,j,c} & \text{se } x_i \text{ non è usato} \\ S_{i,j-1,c} & \text{se } y_j \text{ non è usato} \end{cases}$$

Istanza:  $X_m = \langle x_1, \dots, x_m \rangle$  sequenza di  $m$  elementi,  $Y_n = \langle y_1, \dots, y_n \rangle$  sequenza di  $n$  elementi,  $C \in \mathbb{N}$  capacità e  $\omega: \Sigma \rightarrow \mathbb{N}^+$

Soluzione: Una più lunga sottosequenza comune tra  $X$  e  $Y$  di ingombro complessivo  $\leq C$

Definizione sottoproblema  $i, j, c$ : Abbiamo  $X_i$  tc  $0 \leq i \leq m$  e  $Y_j$  tc  $0 \leq j \leq n$  e  $c$  tc  $0 \leq c \leq C$  vogliamo  $Z_{i,j,c}$  tc  $Z$  è LCS <sub>$i,j$</sub>  con  $c \leq C$

Equazioni di ricorrenza:

caso base:  $Z_{i,j,c} = \emptyset$ ,  $\text{opt}(i,j,c) = 0$  se  $i = 0 \vee j = 0 \vee c = 0$

Oss: Elimino dalle sottosequenze tutti gli elementi  $x_i, y_j$  tc  $\omega(x_i) > c$

passo ricorsivo:  $Z_{i,j,c} = \begin{cases} \max\{Z_{i-1,j,c}, Z_{i,j-1,c}\} & \text{se } x_i \neq y_j \\ \max\{Z_{i-1,j-1,c-\omega(x_i)} + \langle x_i \rangle, Z_{i-1,j-1,c}\} & \text{se } x_i = y_j \end{cases}$

devo confrontare se ha senso aggiungere o meno

$$X = \langle 1, 3, 8, 4, 5 \rangle \quad Y = \langle 1, 2, 3, 8, 5 \rangle \quad C = 19$$

Algoritmo DP:

procedure LCS.weighted(X, Y, C)

for i=0 to m

for c=0 to C

$S_{0,j,0} = 0$

for j=0 to n

for c=0 to C

$S_{i,0,0} = 0$

for i=0 to m

for j=0 to n

$S_{0,0,j} = 0$

for i=1 to m

for j=1 to n

for c=1 to C

if  $x_i \neq y_j$

$S_{i,j,c} = \max\{S_{i-1,j,c}, S_{i,j-1,c}\}$

else

if  $w(x_i) > c$

$S_{i,j,c} = S_{i-1,j-1,c}$

else

$S_{i,j,c} = S_{i-1,j-1,c} - w(i) + 1$

if  $S_{i,j,c} > \text{max-val}$

pos-i = i

pos-j = j

pos-c = c

serve solo per

ricostruire la sol

return  $S_{m,n,C}$

CCS pesata

Ricostruisci sol ottimale: chiamo passando ric.sol.ott.wLCS(pos\_i, pos\_j, pos\_c)

```
def ric_schott_wlcs(i,j,c):
```

if  $i > 0$  and  $j > 0$ :

if  $x_i = y_j$ :

if  $c \geq \omega(x_i)$ :

ric.sol.att.wlcs(i-1,j-1,c-w(i))+ $\langle x_i \rangle$

else

ric-sd-ott-wlcs(i-‡,j-‡,c)

else

if  $S_{i,j-s,c} > S_{i-s,j,c}$

ric-sd-ott-wlcs(i,j-s,c)

else

ric\_sd\_ott\_wics(i,j,c)

Esempio: Per semplicità riporto un caso in cui  $X_m = Y_n = \{2, 2, 3, 4\}$  e  $C = 8$

# Constrained WIS

Oss: Constrained WIS è WIS + Knapsack

Definizione: Dato un insieme  $\{1, \dots, n\}$  di attività, trovare un suo sottoinsieme di attività compatibili di valore massimo e peso complessivo  $\leq C$ .

Istanza:  $\{1, \dots, n\}$  con  $([s_i, e_i], v_i, w_i) \quad \forall i \in \{1, \dots, n\}, C \in \mathbb{R}^+$

Soluzione:  $S \subseteq \{1, \dots, n\}$  tc  $\text{comp}(S) = T \wedge w(S) \leq C \wedge V(S) = \max_{A \subseteq \{1, \dots, n\}: \text{comp}(A) = T \wedge w(A) \leq C} \{V(A)\}$

Sottoproblema  $i, c$ : Abbiamo un insieme  $\{1, \dots, i\}$  con  $0 \leq i \leq n$  e  $c$  tc  $0 \leq c \leq C$

Definizione variabile  $S_{i,c}$ : sottinsieme  $\{1, \dots, i\}$  di attività compatibili di val max e peso  $\leq c$

Definizione variabile  $\text{OPT}_{i,c}$ : valore di  $S_{i,c}$

Equazioni di ricorrenza:

caso base:  $S_{i,c} = \emptyset \quad \text{OPT}_{i,c} = 0 \quad \text{se } i=0 \vee c=0$

Dato  $\psi(i) = \max \{j \mid j < i \text{ e } j \text{ compatibile con } i\}$

passo ricorsivo:  $\begin{cases} S_{i-1,c} & \text{se } w_i > c \vee \text{OPT}_{i-1,c} \geq \text{OPT}_{\psi(i),c-w_i} + v_i \\ S_{\psi(i),c-w_i} \cup \{i\} & \text{altrimenti} \end{cases}$

# Constrained WIS

10

Algoritmo bottom-up:

procedure cwis( $n, C$ )

for  $i=0$  to  $n$

$\text{OPT}[i, 0] = 0$

$S[i, 0] = \emptyset$

for  $c=0$  to  $C$

$\text{OPT}[0, c] = 0$

$S[0, c] = \emptyset$

for  $i=1$  to  $m$

for  $c=1$  to  $C$

if  $w_i > c$

$\text{OPT}[i, c] = \text{OPT}[i-1, c]$

$S[i, c] = S[i-1, c]$

else

$V_1 = \text{OPT}[i-1, c]$

$V_2 = \text{OPT}[\psi(i), c - w_i] + v_i$

$\text{OPT}[i, c] = \max(V_1, V_2)$

if  $V_1 \geq V_2$

$S[i, c] = S[i-1, c]$

else

$S[i, c] = S[\psi(i), c - w_i] \cup \{i\}$

return  $\text{OPT}[n, C], S[n, C]$

# Strutture dati per insiemi disgiunti

11

Definizione: Abbiamo un insieme di oggetti e vogliamo raggruppargli in dei sottinsiemi disgiunti, ovvero con intersezione disgiunta.

Operazioni:

- makeSet(x): prende elemento  $x$  e crea un insieme contenente solo  $x$  e lo elegge a rappresentante (essendo l'unico elemento)
- union(x, y): prende due elementi di due insiemi diversi e restituisce l'unione di appartenenza di  $x$  e di  $y$ .
- findSet(x): prende un elemento e restituisce il rappresentante dell'insieme dell'elemento

Componenti connesse: Vogliamo scrivere un algoritmo che determini le componenti connesse di un grafo, ovvero elementi raggiungibili in una o più masse.

Connected Components (G):

```
for all  $v \in V$ 
    makeSet(v)
for  $(v_i, v_j) \in E$ 
    if findSet( $v_i$ ) ≠ findSet( $v_j$ )
        union( $v_i, v_j$ )
```

Osservazione: Abbiamo bisogno di strutture dati adeguate a memorizzare tali informazioni, ovvero flessibile e dinamiche.

Liste concatenate: Ogni insieme ha una lista concatenata, dove il 1° elemento è il rappresentante.

- makeSet(x), crea lista con 1 elemento che ha da tail e rappresentante e next=null.  $\Theta(1)$
- findSet(x), usiamo una lista doppialmente connessa con puntatore al primo elemento, oltre che al next. Avremo  $\forall e \in L: [rapp][elem][next]$ .  $\Theta(1)$ .
- union(x, y): pongo  $tail_x.next() = head_y$ ,  $head_y = null$  e  $tail_x = tail_y$ . Devo aggiornare il rappresentante di ogni elemento quindi.  $\Theta(n)$

Tempi di calcolo: Definisco  $n = n^{\circ}$  makeSet e  $m = n^{\circ}$  makeSet + union + findSet. Sapendo che le union sono max  $n-1$  e le findSet sono  $\Theta(1)$ , avremo tempo massimo  $m = \Theta(n^2)$

Euristica unione pesato: Ogni lista ha attributo length. Con union la len + coda viene attaccata a len + lunga. Riusciamo ad arrivare a  $m = \Theta(m + n \log n)$  perché per ogni elemento il numero massimo di aggiornamenti è  $\log n$ . Avendo  $n$  elementi, arrivo a  $n \log n$ .

Dimostrazione: Considero oggetto  $x$  e calcolo il max di volte che il suo puntatore può essere aggiornato:

- la 1° volta l'insieme risultato ha almeno 2 elementi;
- la 2° volta l'insieme risultato ha almeno 2<sup>2</sup> elementi;
- la  $i^{\circ}$  volta l'insieme risultato ha almeno  $2^i$  elementi.

Foreste: Struttura dati: dove ogni sottinsieme è un albero radicato, la radice è il rappresentante. Ogni nodo punta al padre, evitando aggiornare i rappresentanti. Potevamo farlo anche sulla lista ma sono più lunghe e quindi avrebbe richiesto troppo tempo.

Operazioni:

makeSet( $x$ ): albero con solo un nodo e punta a sé stesso.  $\Theta(1)$

findSet( $x$ ): si seguono i puntatori. Caso peggiore: ogni padre ha un figlio.  $\Theta(n)$

union( $x, y$ ): il rappresentante di uno diventa figlio del rappresentante dell'altro. Non richiede molto tempo ma "paighiamo" in termini di findSet.

Euristica unione per range: Vogliamo unire l'albero più corto a quello più lungo, cercando di tenere la lunghezza dell'albero più corta possibile.

Range: Ogni nodo ha associato un range che indica il più lungo cammino da lui ad una foglia.

# Strutture dati per insiemi disgiunti

12

makeSet( $x$ ):

$$P(x) = x$$

$$\text{Rank}(x) = 0$$

findSet( $x$ ): visto che perdiamo tempo a scorrere l'albero, possiamo modificarlo per migliorare le findSet future?

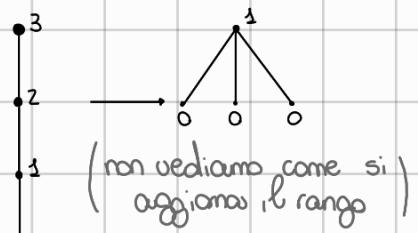
Compressione dei cammini: Ad ogni nodo viene associato come parent il rappresentante del proprio parent e permette di scorrere l'albero e modificarlo per migliorarlo.

findSet( $x$ ):

if  $x \neq P(x)$  # non è rappresentante

$P(x) = \text{findSet}(P(x))$  # assegno il rappresentante del

proprio parent in modo ricorsivo



union( $x, y$ ):

link(findSet( $x$ ), findSet( $y$ ))

link( $x, y$ )

if  $\text{rank}(x) > \text{rank}(y)$ :

$$P(y) = x$$

else

$$P(x) = y$$

if  $\text{rank}(x) = \text{rank}(y)$

$$\text{rank}(y)++$$

Tempi: unione per rango  $O(m \log n)$  + compressione cammini  $O(m \cdot \alpha(m, n))$   $\alpha \leq 4$

Istanza:  $X_m, Y_n$ , col.  $\Gamma \rightarrow \{R, B\}$

Soluzione: Calcolare una delle più lunghe sottosequenze comuni a  $X_m$  e  $Y_n$  nella quale sono presenti un numero pari di elementi rossi.

Sottoproblema  $i, j, r$ :  $X_i$  e  $Y_j$  sottosequenze:  $0 \leq i \leq m$  e  $0 \leq j \leq n$  utilizzando  $r$  pari simboli rossi

Equazioni di ricorrenza: al posto di  $r$  uso un valore  $p$  che indica se pari/dispari.

caso base:  $C_{i,j,p} = \begin{cases} 0 & \text{se } p=0 \wedge i=0 \vee j=0 \\ -\infty & \text{se } p=1 \wedge i=0 \vee j=0 \end{cases}$

$$\begin{cases} C_{i-1,j-1,p+1} & \text{se } x_i = y_j \wedge \text{col}(x_i) \neq R \\ C_{i-1,j-1,p} & \text{se } x_i = y_j \wedge \text{col}(x_i) = R \\ \max\{C_{i-1,j,p}, C_{i-1,j-1,p}\} & \text{se } x_i \neq y_j \end{cases}$$

passo ricorsivo:  $C_{i,j,p} = \begin{cases} \max\{C_{i-1,j-1,p+1}, C_{i-1,j-1,p}\}^* & \text{se } x_i = y_j \wedge \text{col}(x_i) = R \\ \max\{C_{i-1,j,p}, C_{i-1,j-1,p}\} & \text{se } x_i \neq y_j \end{cases}$

\* devo fare un confronto in quanto non sono sicuro di poter prendere l'elemento.

$X = \langle A \rangle$ ,  $Y = \langle A \rangle$  con  $A$  rosso e  $p$  dispari.

Allora  $C_{1,1,0}$  mi manda a vedere  $C_{0,0,-\infty}$  e non va bene. Allora metto il controllo con max: andrà a prendere  $C_{i-1,j-1,p}$

Istanza:  $(V, E, W, f)$  tc  $f: E \rightarrow N$

Soluzione:  $\forall (i, j) \in V^2$  peso di un cammino minimo da  $i$  a  $j$  nel quale la sequenza dei valori associati agli archi è strettamente crescente

Problema ausiliario:  $\forall (i, j) \in V^2$  e  $\forall (a, b) \in (f(E))^2$  peso di un cammino minimo da  $i$  a  $j$  nel quale la sequenza dei valori associati agli archi è strettamente crescente e con valore associato al 1° arco =  $a$  e all'ultimo arco =  $b$

Sottoproblema  $K$ -esimo:  $\forall (i, j) \in V^2$  e  $\forall (a, b) \in (f(E))^2$  peso di un cammino minimo da  $i$  a  $j$  nel quale la sequenza dei valori associati agli archi è strettamente crescente e con valore associato al 1° arco =  $a$  e all'ultimo arco =  $b$  con vertici intermedi  $\in \{1, \dots, K\}$

Caso base:  $K = 0$   $\forall (i, j) \in V^2 \quad \forall (a, b) \in (f(E))^2 \quad d_{i,j}^{(0,a,b)} = \begin{cases} w_{i,j} & \text{se } i \neq j \wedge (i, j) \in E \wedge f(i, j) = a = b \\ \infty & \text{altrimenti} \end{cases}$

Passo ricorsivo:  $K > 0$

$$K \notin \text{cammino} \quad d_{i,j}^{(K,a,b)} = d_{i,j}^{(K-1,a,b)}$$

$$K \in \text{cammino} \quad d_{i,j}^{(K,a,b)} = \min_{(c,d) \in (f(E))^2 | c < d} \left\{ d_{i,K}^{(K-1,a,c)} + d_{K,j}^{(K-1,d,b)} \right\} = e_2$$

Equazioni di ricorrenza:  $d_{i,j}^{(K,a,b)} = \min \{ e_1, e_2 \}$

Sol problema originario =  $d_{i,j}^K = \begin{cases} 0 & \text{se } i = j \\ \min_{(a,b) \in (f(E))^2} \{ d_{i,j}^{(K,a,b)} \} & \text{altrimenti} \end{cases}$

Istanza:  $(V, E, W, f)$   $f: V \rightarrow N^+$   $H \in N^+$

Soluzione:  $\forall (i, j) \in V^2$  il peso di un cammino minimo da  $i$  a  $j$  con somma dei valori associati ai vertici  $\leq H$ .

Sottoproblema:  $\forall (i, j) \in V^2, K \in \{1, \dots, n\}$  il peso di un cammino minimo da  $i$  a  $j$  con somma dei valori associati ai vertici  $\leq h$  e vertici intermedi  $\{1, \dots, k\}$

Variabile associata:  $d^{(K, h)}_{i,j}$  peso del cammino minimo da  $i$  a  $j$  con vertici intermedi  $\{1, \dots, k\}$  e peso associato ai vertici  $\leq h$

$$\text{Caso base: } \forall (i, j) \in V^2, h \in N, K=0 \quad d^{(0, h)}_{i,j} = \begin{cases} 0 & \text{se } i=j \\ w_{ij} & \text{se } i \neq j \wedge (i, j) \in E \wedge f(i) + f(j) \leq h \\ \infty & \text{altrimenti} \end{cases}$$

Passo ricorsivo:  $K > 0$

$$\begin{aligned} K \notin \text{cammino} \quad d^{(K, h)}_{i,j} &= d^{(K-1, h)}_{i,j} = e_3 && \text{se no si conta 2 volte} \\ K \in \text{cammino} \quad d^{(K, h)}_{i,j} &= \min \left\{ d^{(K-1, h_1)}_{i,k} + d^{(K-1, h_2)}_{k,j} \right\} \quad h_1 + h_2 - f(K) \leq h = e_2 \end{aligned}$$

$$\text{Equazioni di ricorrenza: } d^{(K, h)}_{i,j} = \min \{e_3, e_2\}$$