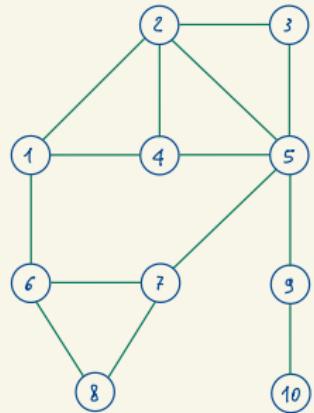
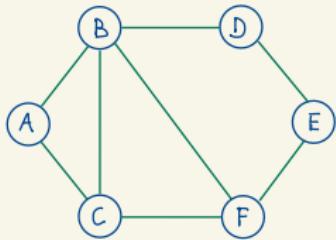


Algoritmi e Strutture Dati

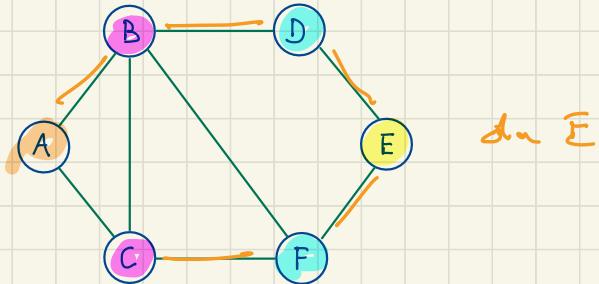
Lezione 21

10 novembre 2025

Attraversamento di grafi



VISITA IN AMPIEZZA (Breadth first search)

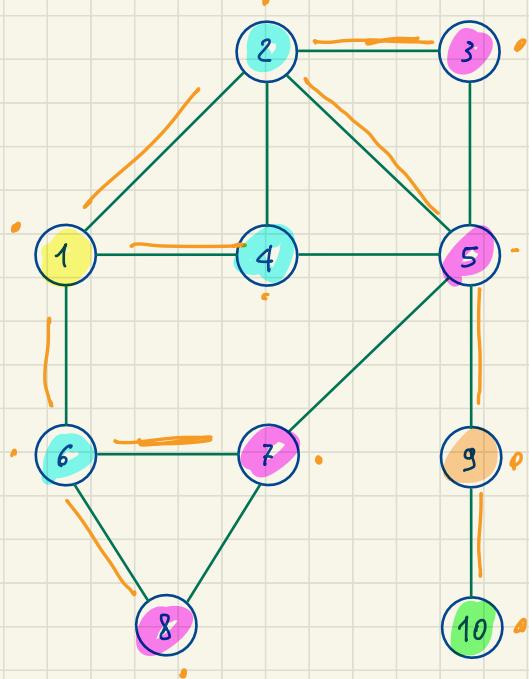


dai E

- Si inizia visitando un vertice s
- Si visitano i vertici adiacenti a s
- Si visitano i vertici adiacenti ai vertici adiacenti a s ,
non ancora visitati

...

ESEMPIO da 1



X X 4 6 X 5 X 8 X 9 X 10

ALGORITMO `visitaInAmpiezza` (grafo $G = (V, E)$, vertice s) \rightarrow Albero

$C \leftarrow$ coda vuota

$T \leftarrow$ albero formato solo da s

marca s come raggiunto

$C.enqueue(s)$

WHILE NOT $C.isEmpty()$ DO

$m \leftarrow C.dequeue()$

FOR EACH $(u, v) \in E$ DO

IF v non è marcato come raggiunto THEN

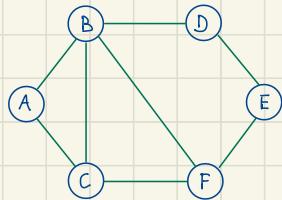
aggiungi v e (u, v) a T

marca v come raggiunto

$C.enqueue(v)$

RETURN T

VISITA IN AMPIEZZA: tempo



LISTA DI ARCHI

(A,B)

(A,C)

(B,C)

(B,D)

(B,F)

(C,F)

(D,E)

(E,F)

n iterazioni \rightarrow

ALGORITMO visitaInAmpiezza (grafo $G = (V,E)$, vertice s) \rightarrow Albero

$C \leftarrow$ coda vuota

$T \leftarrow$ albero formato solo da s

marca s come raggiunto

$C.enqueue(s)$

WHILE NOT $C.isEmpty()$ DO

$m \leftarrow C.dequeue()$

FOR EACH $(u,v) \in E$ DO m iterazioni ogni volta

IF v non è marcato come raggiunto THEN

aggiungi v e (u,v) a T

marca v come raggiunto

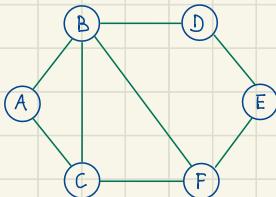
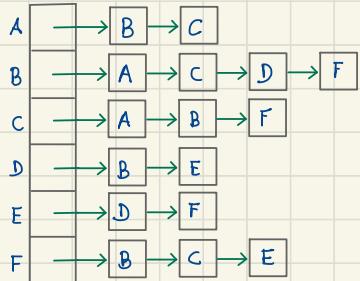
$C.enqueue(v)$

RETURN T

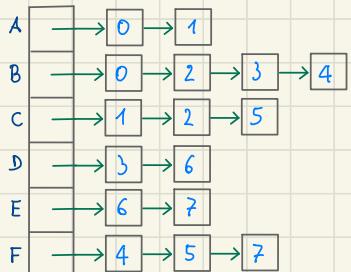
Tempo $\mathcal{O}(n \cdot m)$

VISITA IN AMPIEZZA: tempo

LISTA DI ADIACENZA



LISTA DI INCIDENZA



0	(A,B)
1	(A,C)
2	(B,C)
3	(B,D)
4	(B,F)
5	(C,F)
6	(D,E)
7	(E,F)

ALGORITMO `visitaInAmpiezza (grafo G=(V,E), vertice s) → Albero`

$C \leftarrow$ coda vuota

$T \leftarrow$ albero formato solo da s

mark s come raggiunto

$C.enqueue(s)$

WHILE NOT $C.isEmpty()$ DO

$u \leftarrow C.dequeue()$

FOR EACH $(u,v) \in E$ DO

IF v non è marcato come raggiunto THEN

aggiungi v e (u,v) a T

mark v come raggiunto

$C.enqueue(v)$

RETURN T

$d(u)$ iterazioni

n iterazioni

in totale $2m$ iterazioni

II

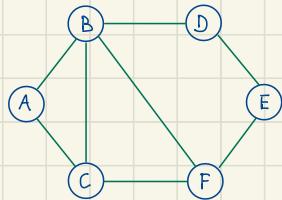
$$\sum_{u \in V} d(u)$$

Tempo $O(n+m)$

$$0 \leq m \leq n^2$$

grafo connesso
 $m \geq n-1$

VISITA IN AMPIEZZA: tempo



MATRICE DI ADIACENZA

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

ALGORITMO `visitaInAmpiezza (grafo G=(V,E), vertice s) → Albero`

$C \leftarrow$ coda vuota

$T \leftarrow$ albero formato solo da s

marca s come raggiunto

$C.\text{enqueue}(s)$

WHILE NOT $C.\text{isEmpty}()$ DO

$m \leftarrow C.\text{dequeue}()$

FOR EACH $(u,v) \in E$ DO "push" "a ogni iterazione"

IF v non è marcato come raggiunto THEN

aggiungi v e (u,v) a T

marca v come raggiunto

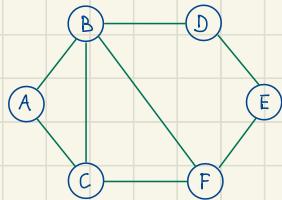
$C.\text{enqueue}(v)$

RETURN T

iterazione

Tempo $\Theta(n^2)$

VISITA IN AMPIEZZA: tempo



MATRICE DI INCIDENZA

	(A,B)	(A,C)	(B,C)	(B,D)	(B,F)	(C,F)	(D,E)	(E,F)
A	1	1	0	0	0	0	0	0
B	1	0	1	1	1	0	0	0
C	0	1	1	0	0	1	0	0
D	0	0	0	1	0	0	1	0
E	0	0	0	0	0	0	1	1
F	0	0	0	0	1	1	0	1

ALGORITMO visitaInAmpiezza (grafo $G = (V, E)$, vertice s) \rightarrow Albero

$C \leftarrow$ coda vuota

$T \leftarrow$ albero formato solo da s

marca s come raggiunto

$C.\text{enqueue}(s)$

WHILE NOT $C.\text{isEmpty}()$ DO

$m \leftarrow C.\text{dequeue}()$

FOR EACH $(u, v) \in E$ DO

IF v non è marcato come raggiunto THEN

aggiungi v e (u, v) a T

marca v come raggiunto

$C.\text{enqueue}(v)$

RETURN T

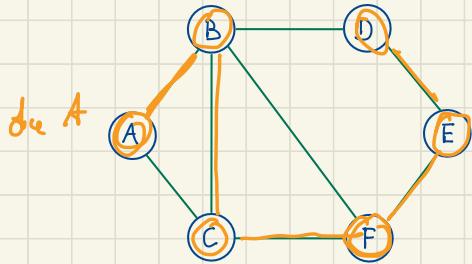
n iterazioni

n vertici
a vertici
m vertici
m.

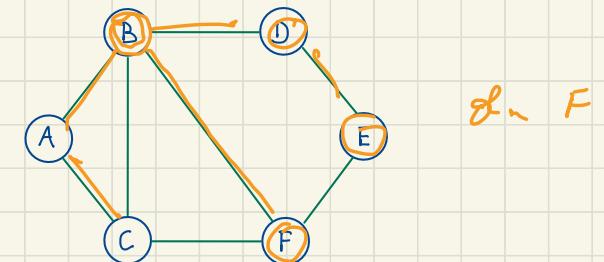
Tempo $\mathcal{O}(n \cdot m)$

VISITA IN PROFONDITÀ (Depth first search)

- Si inizia visitando un vertice s
- Partendo da s si percorre un cammino di vertici non ancora raggiunti visitandoli tutti, terminando su un vertice privo di vicini non ancora raggiunti.
- Si ripete dal passo precedente partendo dall'ultimo vertice sul cammino precedente che ha un vertice non ancora raggiunto



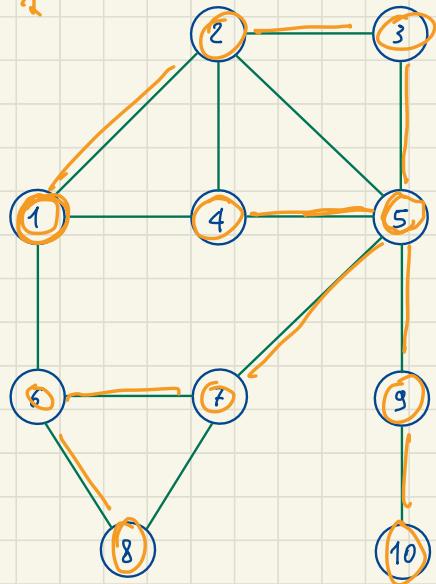
A, B, C, F, E, D



F, B, A, C, D, E

ESEMPIO

da 1



1 - 2 - 3 - 5 - 4 - 7 - 6 - 8 - 9 - 10

ALGORITMO visitaInProfondità (grafo $G = (V, E)$, vertice s) \rightarrow Albero

$T \leftarrow$ albero formato solo da s

visitaRicorsiva (G, s, T)

RETURN T

PROCEDURA visitaRicorsiva (grafo $G = (V, E)$, vertice u , albero T)

marca u come visitato

FOR EACH $(u, v) \in E$ DO

IF v non è marcato come visitato THEN

aggiungi v e (u, v) a T

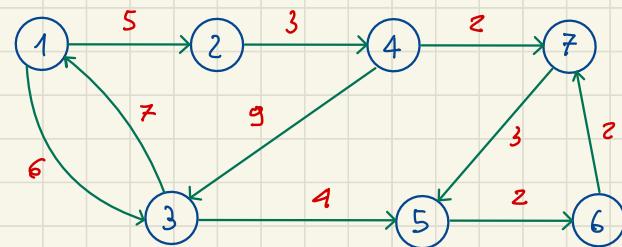
visitaRicorsiva (G, v, T)

Problemi di ottimizzazione e algoritmi *greedy*

GRAFI PESATI (sugli archi)

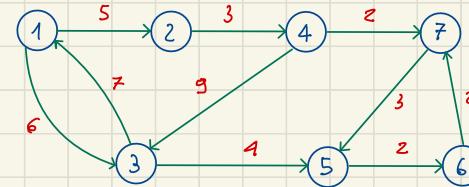
$G = (V, E)$ grafo

$w: E \rightarrow \mathbb{R}$ funzione peso



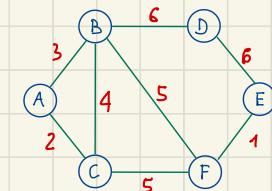
GRAFI PESATI : alcuni problemi

CAMMINI MINIMI



COMMESO VIAGGIATORE

ALBERO RICOPRENTE MINIMO (grafi non orientati)

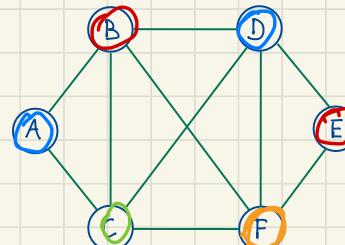
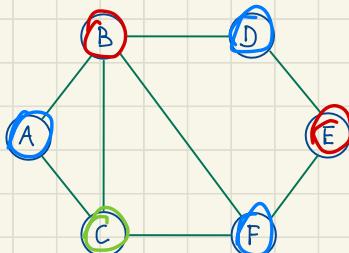


PROBLEMI DI OTTIMIZZAZIONE: esemp:

COLORAZIONE DI GRAFI

Istanza Grafo non orientato $G = (V, E)$

Problema Trovare il minimo numero di colori da attribuire ai vertici di G
in modo che vertici adiacenti abbiano colori differenti



PROBLEMI DI OTTIMIZZAZIONE: esempio

COLORAZIONE DI GRAFI

Istanza Grafo non orientato $G = (V, E)$

Problema Trovare il minimo numero di colori da attribuire ai vertici di G in modo che vertici adiacenti abbiano colori differenti

colorazione: $c: V \rightarrow \text{Colori}$ t.c. $\underbrace{(x,y) \in E \Rightarrow c(x) \neq c(y)}_{\text{vincolo}}$

obiettivo: minimizzare $\#\text{Im}(c)$

Soluzione ammissibile: vertici adiacenti hann. colori diversi

Soluzione ottima soluzione ammissibile con $\#$ colori minimo

PROBLEMI DI OTTIMIZZAZIONE:

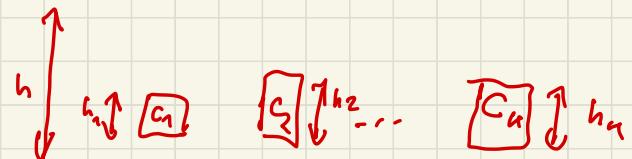
Tra tutte le soluzioni AMMISSIBILI  Soddisfano i vincoli posti dal problema

vogliamo determinarne una OTTIMA
rispetto a un dato criterio

 di solito massimizzazione o minimizzazione

PROBLEMI DI OTTIMIZZAZIONE: esempio

ZAINO MONODIMENSIONALE



Istanza Zaino di altezza h , K contenitori c_1, c_2, \dots, c_k di altezze h_1, h_2, \dots, h_k

Problema Scegliere quali contenitori collocare nello zaino, in modo da riempirlo il più possibile

Soluzione ammissibile sottoinsieme $S \subseteq \{c_1, c_2, \dots, c_k\}$ f.c.

$$\text{s.t. } \underline{F(S) \leq h} \quad \text{dove} \quad F(S) = \sum_{c_i \in S} h_i$$

vincolo

Soluzione ottima soluzione ammissibile S^*

$$\text{t.c. } F(S^*) \geq f(S) \quad \text{per ogni } S \text{ ammiss.}$$

PROBLEMI DI OTTIMIZZAZIONE: esempio

ZAINO MONODIMENSIONALE

Istanza Zaino di altezza h , K contenitori c_1, c_2, \dots, c_K di altezze h_1, h_2, \dots, h_K

Problema Scegliere quali contenitori collocare nello zaino, in modo da riempirlo il più possibile

Soluzione ammissibile sottoinsieme $S \subseteq \{c_1, c_2, \dots, c_K\}$ t.c.

$$f(S) \leq h \quad \text{con} \quad f(S) = \sum_{c_i \in S} h_i$$

Soluzione ottima sottoinsieme ammissibile $S^* \subseteq \{c_1, c_2, \dots, c_K\}$ t.c.

$$f(S^*) \geq f(S) \quad \text{per ogni } S \text{ ammissibile}$$

es.

$$h = 6 \quad K = 4$$

$$h_1 = 1.5 \quad h_2 = 2.5 \quad h_3 = 3 \quad h_4 = 5$$

$$S_1 = \{c_1, c_4\} \quad \text{NON AMM}$$

$$S_2 = \{c_4\} \quad \text{AMM} \quad f(S_2) = 5$$

$$S_3 = \{c_1, c_3\} \quad \text{AMM} \quad f(S_3) = 6.5$$

$$S_4 = \{c_2, c_3\} \quad \text{AMM} \quad f(S_4) = 5.5$$

OPTIMA

TECNICA GREEDY

Q PROBLEMA DI OTTIMIZZAZIONE

- DATO C INSIEME DI CANDIDATI
- TROVARE $S \subseteq C$ OTTIMA

Strategia

- Inizialmente $S \leftarrow \emptyset$
- Ad ogni passo:
 - prelevo da C l'elemento x "migliore"
 - se $S \cup \{x\}$ è ammmissibile aggiungo x a S
- Termino quando ho esaminato tutti i candidati

TECNICA GREEDY

PROBLEMA DI OTIMIZZAZIONE

- DATO C INSIEME DI CANDIDATI
- TROVARE $S \subseteq C$ OTTIMA

Strategia

- Inizialmente $S \leftarrow \emptyset$
- Ad ogni passo:
 - preferisco l'elemento x "migliore"
 - Se $S \cup \{x\}$ è ammmissibile aggiungo x a S
- Termino quando ho esaminato tutti i candidati

ESPANSIONE

SOLUZIONE AMMISSIBILE

La soluzione soddisfa i vincoli
del problema

SCELTA DELL'OTTIMO LOCALE

Tra i tutti i candidati disponibili
a ogni passo si sceglie quello
che al momento appare migliore

SCELTA IRREVOCABILE

Le scelte fatte non vengono
più messe in discussione

TECNICA GREEDY : schema

ALGORITMO greedy (insieme C) \rightarrow soluzione

$S \leftarrow \emptyset$

WHILE $C \neq \emptyset$ DO

$x \leftarrow \text{selezione}(C)$

elemento
considerato
nugare al numero

$C \leftarrow C \setminus \{x\}$

IF $S \cup \{x\}$ è ammssibile THEN

$S \leftarrow S \cup \{x\}$

RETURN S

TECNICA GREEDY: esempio

ZAINO MONODIMENSIONALE

Istanza Zaino di altezza h , K contenitori c_1, c_2, \dots, c_K di altezze h_1, h_2, \dots, h_K

Problema Scegliere quali contenitori collocare nello zaino, in modo da riempirlo il più possibile

Soluzione ammissibile sottoinsieme $S \subseteq \{c_1, c_2, \dots, c_K\}$ t.c.

$$f(S) \leq h \quad \text{con} \quad F(S) = \sum_{c_i \in S} h_i$$

Soluzione ottima sottoinsieme ammissibile $S^* \subseteq \{c_1, c_2, \dots, c_K\}$ t.c.

$$f(S^*) \geq f(S) \quad \text{per ogni } S \text{ ammissibile}$$

$$C = \{c_1, c_2, \dots, c_K\}$$

CANDIDATI = CONTENITORI

STRATEGIA GREEDY:

ispezioniamo i contenitori in ordine di altezza,
dal più alto al più basso

"selezione": scegli il contenitore più alto

ZAINO MONODIMENSIONALE

FS 1 $h = 20$

c_1 $h_1 = 8$

c_2 $h_2 = 7$

c_3 $h_3 = 6$

c_4 $h_4 = 3$

c_5 $h_5 = 2$

c_6 $h_6 = 1$

TOT 20

OPTIMA

$\{c_2, c_3, c_4, c_5, c_6\}$
OPTIMA 20

FS 2 $h = 20$

c_1 $h_1 = 8$

c_2 $h_2 = 7$

c_3 $h_3 = 6$

c_4 $h_4 = 3$

c_5 $h_5 = 3$

c_6 $h_6 = 1$
TOT 19

FS 3 $h = 20$

c_1 $h_1 = 10$

c_2 $h_2 = 8$

c_3 $h_3 = 8$

c_4 $h_4 = 3$

c_5 $h_5 = 3$

TOT 18

$\{c_2, c_3, c_4\}$ 19
OPTIMA

Esempio: ZAINO CON VALORI (o Sacco del Ladro)

c_1

c_2

c_K

Istanza Sacco che può portare al massimo peso P p_1, v_1 p_2, v_2 p_K, v_K

K oggetti c_1, c_2, \dots, c_K di peso p_1, p_2, \dots, p_K e valore v_1, v_2, \dots, v_K , rispettivamente

Problema Scegliere quali oggetti collocare nel sacco, in modo da massimizzare il valore totale, evitando che il sacco si rompa

Obiettivo guadagnare il più possibile

Vincolo non superare la capacità del sacco

Esempio: ZAINO CON VALORI (o Sacco del Ladro)

Istanza Sacco che può portare al massimo peso P

K oggetti: c_1, c_2, \dots, c_K di peso p_1, p_2, \dots, p_K e valore v_1, v_2, \dots, v_K , rispettivamente
Problema Scegliere quali oggetti collocare nel sacco, in modo da massimizzare
il valore totale, evitando che il sacco si rompa

sottoinsieme $S \subseteq C$

$$val(S) = \sum_{c_i \in S} v_i \quad peso(S) = \sum_{c_i \in S} p_i$$

Soluzione ammissibile

$S \subseteq C$ è ammissibile se $peso(S) \leq P$

Soluzione ottima $S^* \subseteq C$ è ottima se (1) S^* è ammissiblg

(2) $\forall S$ ammissibile $val(S) \leq val(S^*)$

$$C = \{c_1, c_2, \dots, c_K\}$$

insieme oggetti

ZAINO CON VALORI: strategia greedy

ispeziona gli oggetti in ordine di valore

"seleziona": scegli l'oggetto di maggior valore

Ese p=20

	P	V
C ₁	12	100
C ₂	3	80
C ₃	6	75
C ₄	6	70
C ₅	3	30

Pes 18

value 210

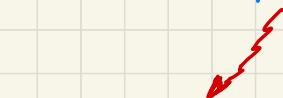
C₂, C₃, C₄, C₅ 255
ottim

Esempio: DISTRIBUTORE AUTOMATICO DI RESTO

Istanza Quantità di monete disponibili per ciascun taglio

Resto da erogare

Problema E' possibile dare il resto esatto?



$$\exists j_5, j_{10}, j_{20}, j_{50} \geq 0 \text{ t.c.}$$

$$j_5 \leq K_5, j_{10} \leq K_{10}, j_{20} \leq K_{20}, j_{50} \leq K_{50}$$

$$\text{e } 5j_5 + 10j_{10} + 20j_{20} + 50j_{50} = R ?$$



$$K_5 = \# \text{ monete da 5 centesimi}$$

$$K_{10} = \# \text{ monete da 10 centesimi}$$

$$K_{20} = \# \text{ monete da 20 centesimi}$$

$$K_{50} = \# \text{ monete da 50 centesimi}$$

$$R = \text{resto}$$

$$K_5, K_{10}, K_{20}, K_{50} \geq 0 \quad R > 0$$

MONETA	QUANTITA'
5	K_5
10	K_{10}
20	K_{20}
50	K_{50}

$$\text{Resto} = R$$

Esempio: DISTRIBUTORE AUTOMATICO DI RESTO

Istanza Quantità di monete disponibili per ciascun taglio

Resto da erogare

Problema E' possibile dare il resto esatto?

$$j_5, j_{10}, j_{20}, j_{50} \geq 0 \text{ t.c.}$$

$$j_5 \leq K_5, j_{10} \leq K_{10}, j_{20} \leq K_{20}, j_{50} \leq K_{50}$$

$$\text{e } 5j_5 + 10j_{10} + 20j_{20} + 50j_{50} = R ?$$

$$K_5 = \# \text{ monete da 5 centesimi}$$

$$K_{10} = \# \text{ monete da 10 centesimi}$$

$$K_{20} = \# \text{ monete da 20 centesimi}$$

$$K_{50} = \# \text{ monete da 50 centesimi}$$

$$R = \text{resto}$$

$$K_5, K_{10}, K_{20}, K_{50} \geq 0 \quad R > 0$$

<u>Esempio</u>	MONETA	QUANTITA'
5	10	10
10	10	10
20	10	10
50	10	10

$$\text{Resto} = 85$$

$$K_5 = K_{10} = K_{20} = K_{50} = 10$$

$$R = 85$$

$$(1 \ 0 \ 4 \ 0 \ /$$

$$(1 \ 1 \ 1 \ 1)$$

Soluzione parziale ammessa

vettore di interi: $(j_5, j_{10}, j_{20}, j_{50})$ t.c.

$$j_5, j_{10}, j_{20}, j_{50} \geq 0,$$

$$j_5 \leq K_5, j_{10} \leq K_{10}, j_{20} \leq K_{20}, j_{50} \leq K_{50} \text{ e}$$

$$5j_5 + 10j_{10} + 20j_{20} + 50j_{50} \leq R$$

DISTRIBUTORE AUTOMATICO DI RESTO: strategia greedy

"seleziona": scegli la moneta disponibile di maggior valore

Esempio 1

MONETA	QUANTITA'	
50	10	1
20	10	1
10	0	
5	5	3

Resto = 85

85

Esempio 2

MONETA	QUANTITA'	
50	10	1
20	10	1
10	0	
5	2	2

Resto = 85

85
NO

Sup. greedy
Sol. corret.

85
SI