

# Tutoraggio Ricerca Operativa 2020/2021

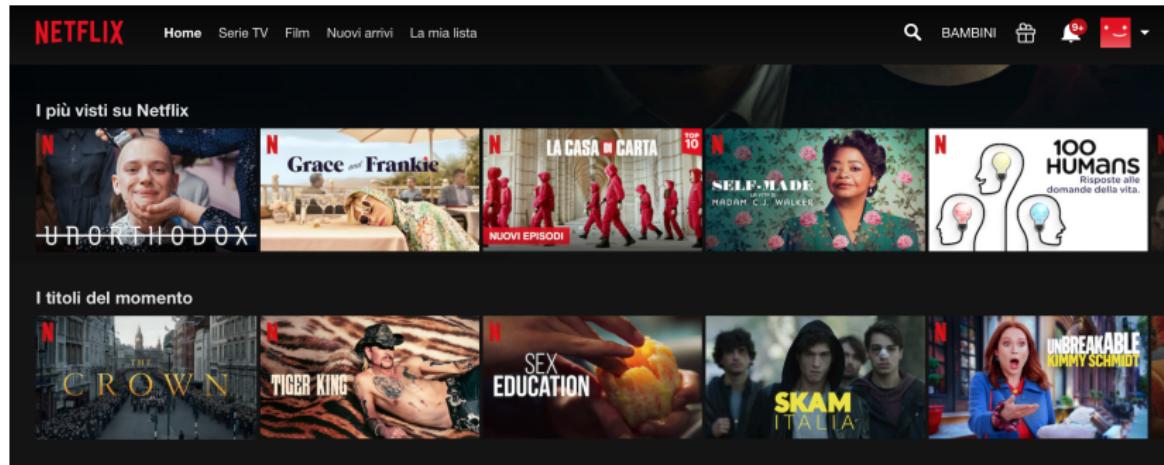
## 2. Programmazione Dinamica (II)

Aurora Rossi, Alice Raffaele, Romeo Rizzi

Università degli Studi di Verona

17 marzo 2021

# Problemi da quarantena (I)



Dopo questa esercitazione avete finito di studiare per oggi e avete a disposizione due ore di tempo prima che sia pronta la cena. Volete guardare qualche episodio delle vostre serie preferite su Netflix, ma quali scegliere?

## Problemi da quarantena (II)

Tutto dipende da qual è il vostro obiettivo, per esempio:

- ① usare il più possibile il tempo a disposizione;
- ② scegliere le puntate che vi piacciono di più in assoluto.

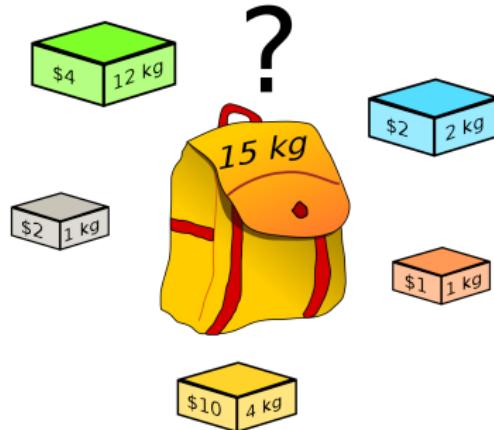
Consideriamo i seguenti episodi.

Serie TV	Durata (min)	Preferenza [1-10]
La Casa di Carta	60	10
Numb3rs	50	5
Stranger Things	45	4
The Good Place	20	7

**Soluzione Obiettivo 1:** Numb3rs, Stranger Things, The Good Place  
(tempo impiegato: 115 minuti; valore: 16);

**Soluzione Obiettivo 2:** La Casa di Carta, The Good Place (tempo impiegato: 80 minuti; valore: 17).

# Il problema dello Zaino



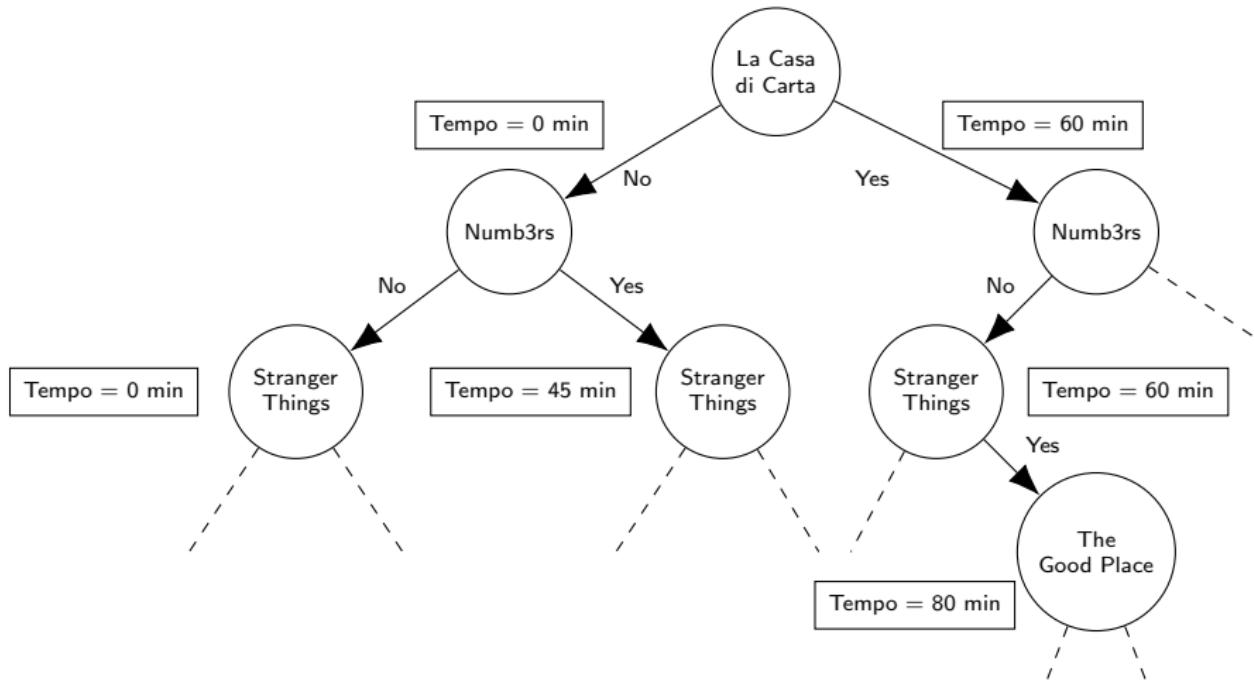
Dati:

- una collezione di oggetti  $S = \{1, \dots, n\}$ , ciascuno caratterizzato da un peso  $w_i \geq 0$  e un valore  $v_i \geq 0$ ;
- la capacità massima dello zaino  $B$ ;

Trovare un insieme  $S' \subseteq S$  tale che  $\sum_{i \in S'} w_i \leq B$  e che:

- ① minimizzi  $B - \sum_{i \in S'} w_i$ , oppure
- ② massimizzi  $\sum_{i \in S'} v_i$ .

# Approcci possibili - Ricerca esaustiva per l'Obiettivo 1



Si prosegue esplorando l'albero finché non si sono esauriti tutti gli episodi; ogni foglia rappresenta una soluzione (i.e., un sottoinsieme di episodi) feasible o infeasible (se la durata complessiva supera il tempo disponibile).

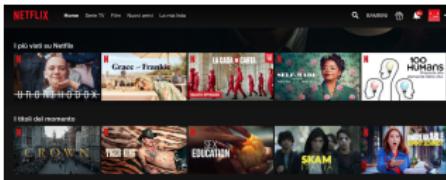
## Approcci possibili - Euristiche

Alcuni possibili **criteri di ordinamento** degli item, per poter selezionare poi i primi che rientrano nello zaino:

- Peso non decrescente
- Valore non crescente
- Rapporto valore/peso non crescente (algoritmo greedy di Martello e Toth, 1990)

**Nota:** l'ultimo è il più ragionevole e quello che funziona generalmente meglio, ma nessuno di questi metodi garantisce di trovare la soluzione ottima.

# Applicazioni del problema dello Zaino



# TE 17/02/2016 - Es. 3: Zaino (I)

## Problema 3 (6 punti):

Sia  $B = 36$  la capacità del mio zaino. Si supponga di voler trasportare un sottoinsieme dei seguenti elementi a massima somma dei valori, soggetti al vincolo che la somma dei pesi non ecceda  $B$ .

nome	A	B	C	D	E	F	G	H	I	L	M	N
peso	2	13	14	6	13	3	16	11	4	46	41	44
valore	11	63	60	33	30	13	66	60	20	66	60	20

**3.1(1pt)** quanto vale la somma massima dei valori di elementi trasportabili (con somma dei pesi al più  $B = 36$ )? Quali elementi devo prendere?

**3.2 (1pt)** e nel caso  $B = 33$ ?

**3.3 (1pt)** e nel caso  $B = 28$ ?

**3.4 (1pt)** e nel caso  $B = 26$ ?

**3.5 (2pt)** e se l'oggetto  $H$  non fosse più disponibile, quale sarebbe allora la soluzione ottima per  $B = 26, 28, 33, 36$ ?

Con oggetto  $H$  disponibile:

B	max val	peso	quali prendere
36			
33			
28			
26			

Senza oggetto  $H$ :

- **Sottoproblema:** consideriamo un sottoinsieme di item o una capacità dello zaino più limitata;
- **Casi banali:** non ci sono item oppure la capacità dello zaino è 0;
- **Caso generale:** ?

**Knapsack**( $n, B$ ), dove  $n = \#$  item,  $B =$  capacità dello zaino:

- Per ogni elemento nella lista considerata, ci chiediamo: lo inseriamo nello zaino o lo scartiamo?
- **Preprocessing**: non consideriamo eventuali item che, persino presi singolarmente, eccedano la capacità dello zaino;
- **Suggerimento**: nella consegna dell'esercizio, fate attenzione se tra le richieste un item *non fosse più disponibile*, come l'oggetto  $H$  nel punto 3.5 → Lo considereremo per ultimo!

## Procedimento:

- Si compila una tabella di programmazione dinamica avente:
  - $n + 1$  righe, quanti gli item da considerare (più l'item nullo);
  - $B + 1$  colonne, quanta la capacità dello zaino (più la capacità nulla).
- Si ordinano gli item arbitrariamente, ma ponendo in fondo gli item che potrebbero essere *non disponibili* in alcune richieste.
- Ogni cella  $(i, j)$  rappresenta un sottoproblema analogo all'originale con  $i$  item e  $j$  capacità e conterrà il valore massimo soluzione di quel sottoproblema (NB: stiamo massimizzando il valore complessivo dello zaino - Obiettivo 2);
- Una volta trovata la soluzione alla richiesta del punto 3.1, si saprà rispondere subito anche ai successivi punti 3.2, 3.3 e 3.4.

# TE 17/02/2016 - Es. 3: Zaino (V)

Inizializziamo la tabella compilando le prime due colonne:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		
A (2, 11)	0	0																																			
B (13, 63)	0	0																																			
C (14, 60)	0	0																																			
D (6, 33)	0	0																																			
E (13, 30)	0	0																																			
F (3, 13)	0	0																																			
G (16, 66)	0	0																																			
I (4, 20)	0	0																																			
H (11, 60)	0	0																																			

# TE 17/02/2016 - Es. 3: Zaino (VI)

Partendo dal primo elemento A, valutiamo se e quando sia possibile inserirlo o meno nello zaino e come cambia il valore di quest'ultimo:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
A (2, 11)	0	0	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11				
B (13, 63)	0	0																																			
C (14, 60)	0	0																																			
D (6, 33)	0	0																																			
E (13, 30)	0	0																																			
F (3, 13)	0	0																																			
G (16, 66)	0	0																																			
I (4, 20)	0	0																																			
H (11, 60)	0	0																																			

# TE 17/02/2016 - Es. 3: Zaino (VII)

Proseguiamo con il secondo elemento, procedendo in seguito riga per riga:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		
A (2, 11)	0	0	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	
B (13, 63)	0	0	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	
C (14, 60)	0	0																																			
D (6, 33)	0	0																																			
E (13, 30)	0	0																																			
F (3, 13)	0	0																																			
G (16, 66)	0	0																																			
I (4, 20)	0	0																																			
H (11, 60)	0	0																																			

Riempiamo ogni cella applicando la seguente formula per

$$V[i, j] = \begin{cases} V[i - 1, j], & \text{se } w[i] > j \\ \max\{V[i - 1, j], V[i - 1, j - w[i]] + v[i]\}, & \text{se } w[i] \leq j \end{cases}$$

Completiamo tutta la tabella (occhio ai conti perché se no compromettete anche le future celle):

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36			
-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-							
A (2, 11)	0	0	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11						
B (13, 63)	0	0	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11						
C (14, 60)	0	0	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11							
D (6, 33)	0	0	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11							
E (13, 30)	0	0	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11							
F (3, 13)	0	0	11	13	13	24	33	33	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44							
G (16, 66)	0	0	11	13	13	24	33	33	44	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46						
I (4, 20)	0	0	11	13	20	24	33	33	44	46	46	53	57	64	66	66	77	77	83	87	96	96	96	107	109	109	116	120	127	129	129	140	140	140	143	147	156	157	167	169
H (11, 60)	0	0	11	13	20	24	33	33	44	46	53	60	64	71	73	80	84	93	93	104	106	113	117	124	126	127	137	137	143	147	156	156	167	169	176	180	187			

**Complessità:**  $T(n) = O(n \cdot B)$  → Algoritmo pseudo-polinomiale: in realtà servono  $k = \log B$  bit per rappresentare  $B$ , quindi ( $T(n) = O(n \cdot 2^k)$ ).

# TE 17/02/2016 - Es. 3: Zaino (X)

Ricostruiamo la soluzione a partire dall'ultima cella e risalendo a ritroso:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-			
A (2, 11)	0	0	0	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	
B (13, 63)	0	0	11	11	11	11	11	11	11	11	11	11	11	11	11	11	63	63	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74		
C (14, 60)	0	0	11	11	11	11	11	11	11	11	11	11	11	11	11	11	63	63	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74		
D (6, 33)	0	0	11	11	11	11	11	33	33	44	44	44	44	44	44	44	63	63	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74		
E (13, 30)	0	0	11	11	11	11	11	33	33	44	44	44	44	44	44	44	63	63	74	74	74	74	96	96	107	107	107	107	107	107	107	107	107	107			
F (3, 13)	0	0	11	13	13	24	33	33	44	46	46	57	57	63	63	74	76	76	87	96	96	107	109	109	120	120	120	123	123	134	134	134	134	134	136		
G (16, 66)	0	0	11	13	13	24	33	33	44	46	46	57	57	63	63	74	76	76	87	96	96	107	109	109	120	120	123	123	134	136	136	147	156	156	167	169	
I (4, 20)	0	0	11	13	20	24	33	33	44	46	53	57	64	66	66	77	77	83	87	96	96	107	109	116	120	127	129	129	140	140	140	143	147	156	157	167	169
H (11, 60)	0	0	11	13	20	24	33	33	44	46	53	60	64	71	73	80	84	93	93	104	106	113	117	124	126	127	137	137	143	147	156	156	167	169	176	180	187

---

## Algorithm 1 Stampa soluzione Zaino

---

```
1:  $j = B;$ 
2:  $i = n;$ 
3: while  $i > 0$  do
4:   if  $V[i, j] \neq V[i - 1, j]$  then
5:     print  $i$ ;
6:      $j = j - w[i];$ 
7:   end if
8:    $i = i - 1;$ 
9: end while
```

---

# TE 17/02/2016 - Es. 3: Zaino (XII)

Ora possiamo rispondere anche a tutte le altre domande del TE:

Con oggetto  $H$  disponibile:

B	max val	peso	quali prendere
36	$187 = 60+20+33+11+63$	$36 = 11+4+6+2+13$	H,I,D,A,B
33	$169 = 60+33+13+63$	$33 = 11+6+3+13$	H,D,F,B
28	$143 = 60+20+63$	$28 = 11+4+13$	H,I,B
26	$137 = 60+20+33+11+13$	$26 = 11+4+6+2+3$	H,I,D,A,F

Senza oggetto  $H$ :

B	max val	peso	quali prendere
36	$169 = 13+33+63+60$	$36 = 3+6+13+14$	F,D,B,C
33	$156 = 33+63+60$	$33 = 6+13+14$	D,B,C
28	$140 = 11+13+20+33+63$	$28 = 2+3+4+6+13$	A,F,I,D,B
26	$129 = 13+20+33+63$	$26 = 3+4+6+13$	F,I,D,B