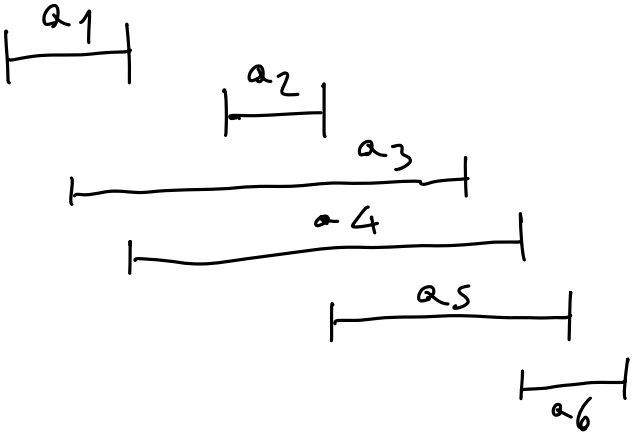
Esercizi su selezione attività

1) (Domanda d'esame)
6 attività

$s = (1, 5, 2, 3, 7, 10)$ $f = (3, 7, 9, 10, 11, 12)$

Determinare l'output di GREEDY-SEL, descrivendo brevemente l'algoritmo



$A = \{a_1\}$ $A = \{a_1, a_2\}$ $A = \{a_1, a_2\}$

last = 1 last = 2 last = 2

$A = \{a_1, a_2\}$ $A = \{a_1, a_2, a_5\}$ $\boxed{A = \{a_1, a_2, a_5\}}$

last = 2 last = 5 last = 5

2) Scelte greedy alternative per selezione di attività

- scegli l'attività di durata inferiore

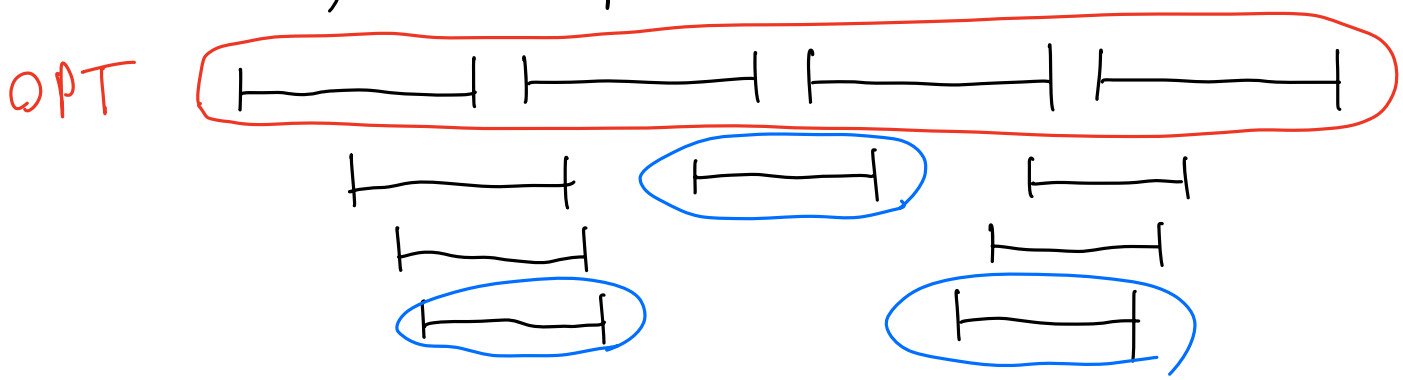
  NO; controesempio: 

- scegli l'attività che inizia prima / finisce per ultima

  NO; controesempio: 

- scegli l'attività col minor numero di sovrapposizioni con altre attività

  NO; controesempio:



- scegli l'attività che inizia per ultima

  Sì, è il "duale" di GREEDY-SEL → esercizio

→ la maggior parte degli algoritmi greedy non sono (sempre) corretti

Paradigma generale:

si basa su 2 proprietà:

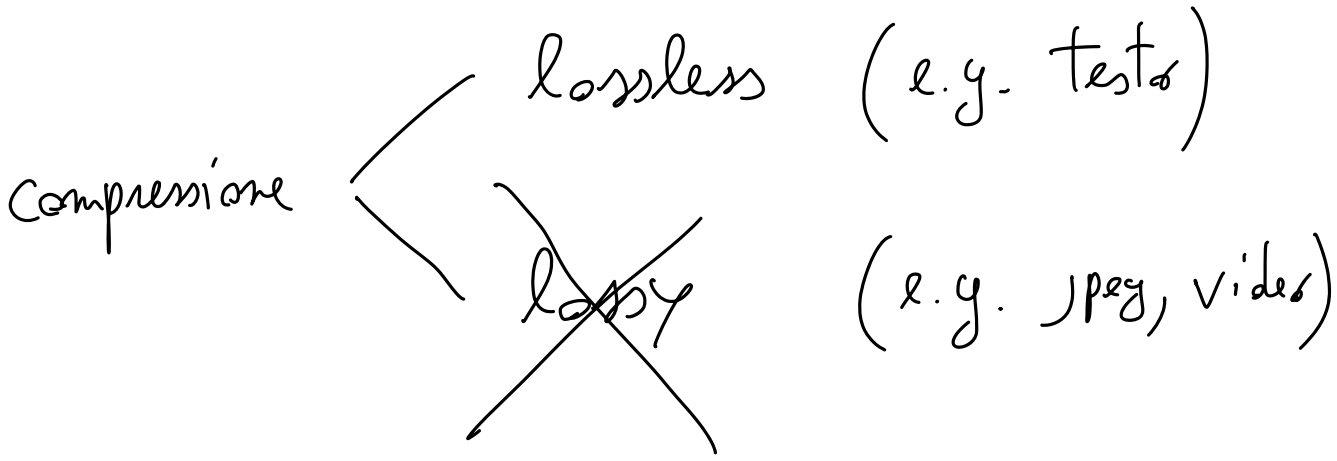
- Proprietà di scelta greedy

  la soluzione ottima può essere costruita componendo scelte "incaute". Si dimostra con la tecnica del cut & paste

- Proprietà di sottostruttura ottima con spazio dei sottoproblemi lineare

  significa dimostrare che la soluzione ottima, che contiene la scelta greedy, contiene una soluzione a l'unico sottoproblema da risolvere dopo aver fatto la scelta greedy

Compressione dei dati



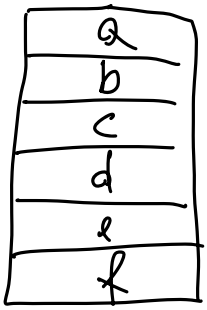
File con 100k caratteri ASCII → 800Kbit

	a	b	c	d	e	f
frequenze (%)	45	13	12	16	9	5

min n° di bit per codificare 6 caratteri è 3

a	b	c	d	e	f	
000	001	010	011	100	101	← "codeword"

serve anche una tabella di conversione:



} 6 byte = 48 bit

→ in totale servono 300kbit + 48 bit

→ risparmio > 50 %

decodifica 001 | 011 | 100 | ...

→ lookup in posizione 4

b d e

In generale:

C = alfabeto di simboli presenti nel file

possiamo la codifica su C e non su tutto l'universo dei simboli possibili

da determinare:

funzione di encoding $e : C \to \{0,1\}^*$

proprietà che e deve avere:

- invertibile: $a \neq b \Rightarrow e(a) \neq e(b)$
- ammettere algoritmi efficienti, come e^{-1}

Fixed-length encoding: associa ad ogni carattere di C una stringa di 0 e 1 della stessa lunghezza

È l'idea + semplice possibile. Nota: ignora totalmente le frequenze → posso usare questa informazione per migliorare le cose?

Idea: associare a caratteri + frequenti codeword + corte, e viceversa

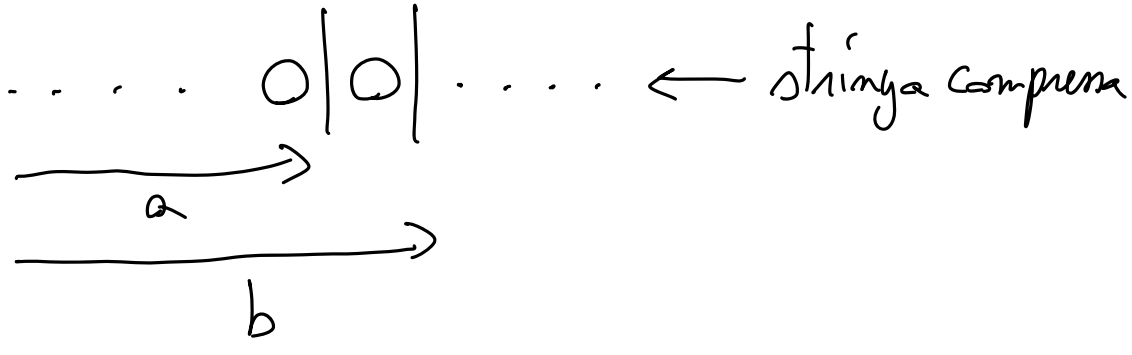
→ Variable-length encoding

Esempio:

	a	b	c	d	e	f
frequenze (%)	45	13	12	16	9	5
codeword	0	00	01	1	100	101

È un buon encoding?

NO: $e(a)\,e(a) = e(b)$



Problema: dopo aver visto il 1° bit non so se fermarmi perché ho visto una codeword completa oppure solo il prefisso di una codeword + lunga. Quindi il problema è che ∃ codeword

che sono prefissi di altre codeword!

Quindi, buone codifiche devono:

- lunghezza variabile delle codeword (più efficiente)
- il codice deve essere libero da prefissi ("prefix code") cioè $\nexists\ a, b \in C$ tali che $e(a)$ è un prefisso di $e(b)$