PROG DIN + Greedy

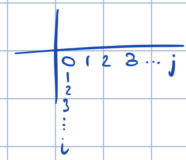
LCS

$$c[i,j] = \begin{cases} 0 & i=0 \text{ OR } j=0 \\ c[i-1,j-1]+1 & i,j>0 \text{ AND } x_i = y_j \\ \max(c[i,j-1], c[i-1,j]) & i,j>0 \text{ AND } x_i \neq y_i \end{cases}$$

RICOSTRUZIONE: seguo le frecce

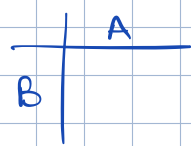
EDIT-D

$$c[i,j] = \begin{cases} i & j=0 \\ j & i=0 \\ \min\big(c[i,j-1]+1, \; c[i-1,j]+1, \; c[i-1,j-1]+p(i,j)\big) & i,j>0 \end{cases}$$



$$p(i,j) = \begin{cases} 0 & x_i = y_j \\ 1 & x_i \neq y_j \end{cases}$$

RICOSTRUZIONE:

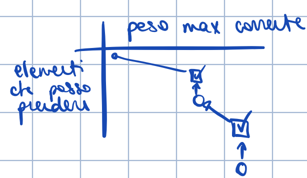


$$\begin{cases} \uparrow = \overline{\text{char}}\ B \\ \leftarrow = \text{char } \underline{A} \\ \nwarrow = \text{char char} \end{cases}$$

per ricostruire le stringhe tengo conto ad ogni scelta.

ZAINO 0-1

$$B[w,k] = \begin{cases} B[k-1, w] & \text{se } w_k > W \\ \max(B[k-1,w], B[k-1, w-w_k] + v_k) \end{cases}$$



RICOSTRUZIONE

if(B[i,k] ≠ B[i-1,k])

flag

$i = i-1$, $K = K - W_i$

else

$i = i-1$

ZAINO F

Zaino Frazionario (W, v[n], w[n])

1. While **w** > 0 e ci sono elementi da aggiungere
2. scegli elemento con il massimo v_i/w_i
3. $x_i \leftarrow \min(1, w/w_i)$
4. rimuovi elemento **i** dalla lista
5. $w \leftarrow w - x_i w_i$

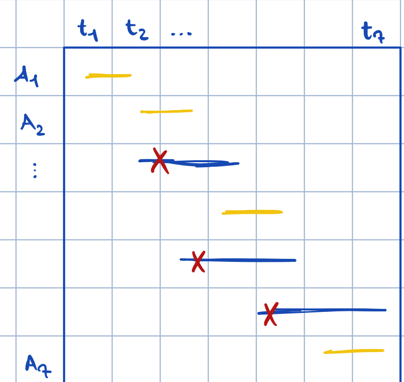
- w – spazio rimanente nello zaino (w = W)
- **Tempo**: $\Theta(n)$ se gli elementi sono già ordinati, altrimenti $\Theta(n \lg n)$

SCHEDULING

- seleziono quella che finisce prima
- tolgo le non compatibili

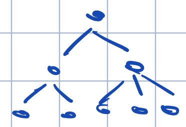
DIMOSTRARE

- ☑ sottostrutt. ottima
- ☑ greedy choice property

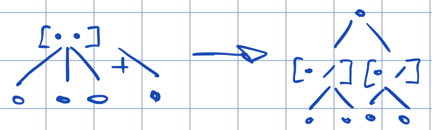


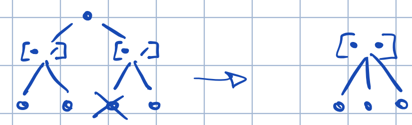
2-3 Alberi

- $2^{h+1}-1 \leq n \leq \frac{3^{h+1}-1}{2}$
- $2^h \leq f \leq 3^h$



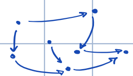
$h = 2$
$n = 8$
$f = 5$

split: 

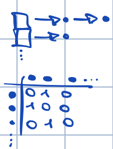
fuse: 

SRC, INS, DEL = $O(\log n)$ al caso peggiore. garantito dal bilanciamento.

GRAFI



con liste
con mat



DENSO: $|E| \simeq |V|^2$
SPARSO: $|E| \simeq |V|$

512
ORDINAMENTO TOPOLOGICO

- DFS per tempo fine
- $\forall v$ ispezionato: head di LL

QUANDO COSA

- shortest
  - $\neg$Pesato → BFS
  - Pesato $\mathbb{R}^+$ → Dijkstra
  - Pesato $\mathbb{R}$ → Bellman Ford
- visita completa → DFS
- comp. conn
  - G: BFS
  - #: DFS
- cicli
  - DFS
  - Negativi: Bellman Ford

	497 BFS	504 DFS
albero	✓	✓
tutto	✗	✓
distanze	✓	✗
camm. minimi	✓	✗
tempo	✗	✓
class. archi	✗	✓

552
$\boxed{\text{DIJKSTRA}}$: BFS con priority q. + RELAX

calcola il cammino minimo di ogni v dalla sorgente

546
$\boxed{\text{BELLMAN-FORD}}$: RELAX in $\Theta(|V|^2)$ + controllo

calcola se esiste un ciclo negativo raggiungibile dalla sorgente.

COSTI

- BFS: $O(|V| + |E|)$
- DFS: $\Theta(|V| + |E|)$
- Dijkstra: DIPENDE *
- B-F: $\Theta(|V| \cdot |E|)$

* con MIN-HEAP: $O((|V|+|E|)\log|V|)$
dalla priority q.

COMMON USAGE

- contare comp. conn → DFS
- grafo connesso → BFS
- $x \to y$ $y \to z$ → 2 DFS

Tabelle Hash

- divisione: $K \% m$
- moltiplicazione: $\lfloor m(KA \% 1) \rfloor$ $0<A<1$

	Ricerca senza successo	Ricerca con successo	Inserimento
Chaining	$1 + \lambda$	$1 + \lambda$	1
Open Addressing	$\frac{1}{1-\lambda}$	$\frac{1}{\lambda}\ln\frac{1}{1-\lambda}$	$\frac{1}{1-\lambda}$

OPEN ADDRESSING

- lineare: $h(K,i) = (h'(K) + i)\% m$
- quadratico: $h(K,i) = (h'(K) + c_1 i + c_2 i^2)\% m$
- doppio: $h(K,i) = (h_1(K) + i h_2(K))\% m$
  - $h_1(K) = K \% m$
  - $h_2(K) = 1 + K\%(m-1)$

Master Theorem

$T(n) = aT(\frac{n}{b}) + f(n)$

$f(n)$ vs. $n^{\log_b a}$

- $f(n) < n^{\log_b a}$ | $f(n) = O(n^{\log_b a - \varepsilon})$ | $f(n) \leq c \cdot n^{\log_b a - \varepsilon}$ $\Rightarrow T(n) = \Theta(n^{\log_b a})$
- $f(n) \sim n^{\log_b a}$ | $f(n) = \Theta(n^{\log_b a})$ | $c_1 \cdot n^{\log_b a}\log^k \leq f(n) \leq c_2 \cdot n^{\log_b a}\log^k$ $\Rightarrow T(n) = \Theta(n^{\log_b a} \cdot \log^{k+1} n)$
- $f(n) > n^{\log_b a}$ | $f(n) = \Omega(n^{\log_b a + \varepsilon})$ | $\begin{cases} f(n) \geq c \cdot n^{\log_b a + \varepsilon} \\ af(\frac{n}{b}) \leq c \cdot f(n) \quad c<1 \end{cases}$ $\Rightarrow T(n) = \Theta(f(n))$