OSSERVAZIONI

BUON ALGORITMO SEQUENZIALE $\longrightarrow$ BUON ALGORITMO PARALLELO

non sempre

Esempio: Merge-sort

BUON ALGORITMO PARALLELO $\longrightarrow$ BUON ALGORITMO SEQUENZIALE

non sempre

Esempio: Bit-Sort

VALUTAZIONE SEQUENZIALE DI BIT-SORT

- Bit-merge

$$t_{bm}(n) = \begin{cases} O(1) & n = 2 \\ 2\, t_{bm}(n/2) + O(n) & n > 2 \end{cases}$$

$$t_{bm}(n) = O(n \log n)$$

- Bit-sort

$$t_{bs}(n) = \begin{cases} O(1) & n = 2 \\ 2\, t_{bs}(n/2) + O(n \log n) & n > 2 \end{cases}$$

(bit-merge)

$$t_{bs}(n) = O(n \log^2 n)$$

Tecnica del ciclo Euleriano

Def. basi di Teoria dei grafi:

- Un grafo <u>diretto</u> D è una coppia (V, E) dove $E \subseteq V^2$. Notazione $(v, w) \in E$

- cammino: è una sequenza di archi $e_1 e_2 \dots e_i e_{i+1} \dots e_k$ tale che il nodo finale di e_i coincide con il nodo iniziale di e_{i+1}, $\forall i$

- ciclo: è un cammino tale che il nodo finale di e_k coincide con il nodo iniziale di e_1

Definizioni:

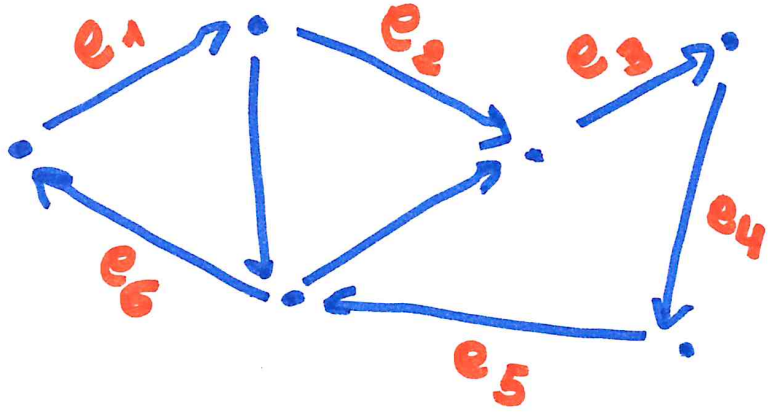
- ciclo **euleriano**: ciclo in cui ogni arco in E compare una e una sola volta

- cammino **euleriano**: cammino in cui... (come sopra)

- grafo **euleriano**: si dice Tale sse contiene un ciclo euleriano

Esempio

D



Problema naturale:

Input: D

Output: È Euleriano?

NO

Notazioni:

$\forall v \in V$ definiamo: $\rho^-(v) = |\{(w,v) \in E\}|$

grado di entrata di v

$\rho^+(v) = |\{(v,w) \in E\}|$

grado di uscita di v

Teorema (Eulero 1736)

D è Euleriano sse $\forall v \in V : \rho^-(v) = \rho^+(v)$

Digressione COMPUTAZIONALE su Tale problema e uno simile

Def:

- Dato D, un ciclo è Hamiltoniano sse è un ciclo dove ogni vertice in V compare una e una sola volta
- D è Hamiltoniano sse contiene un ciclo Hamiltoniano

Euleriano

D è Euleriano?

⇓

Efficiente: $O(n^3)$

$n = |V|$

HAMILTONIANO

D è Hamiltoniano?

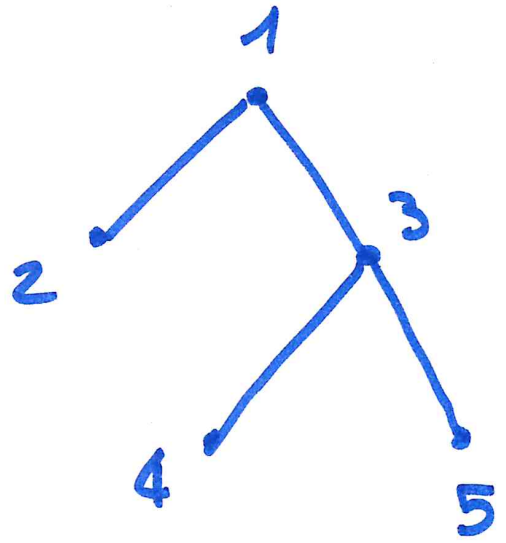
⇓

Difficile: NP-completo

Tecnica del ciclo Euleriano

Viene usata per costruire algoritmi paralleli efficienti che gestiscono strutture dinamiche come:

ALBERI BINARI



	v	$sin(v)$	$des(v)$	$pad(v)$
RADICE	1	2	3	0
→	2	0	0	1
NODO INTERNO	3	4	5	1
→	4	0	0	3
→	5	0	0	3

→ = foglia

Molti problemi ben noti usano alberi. Es:

- Ricerca
- Dizionari
- QUERY
- :

Fondamentale in Tali problemi è la navigazione dell'albero

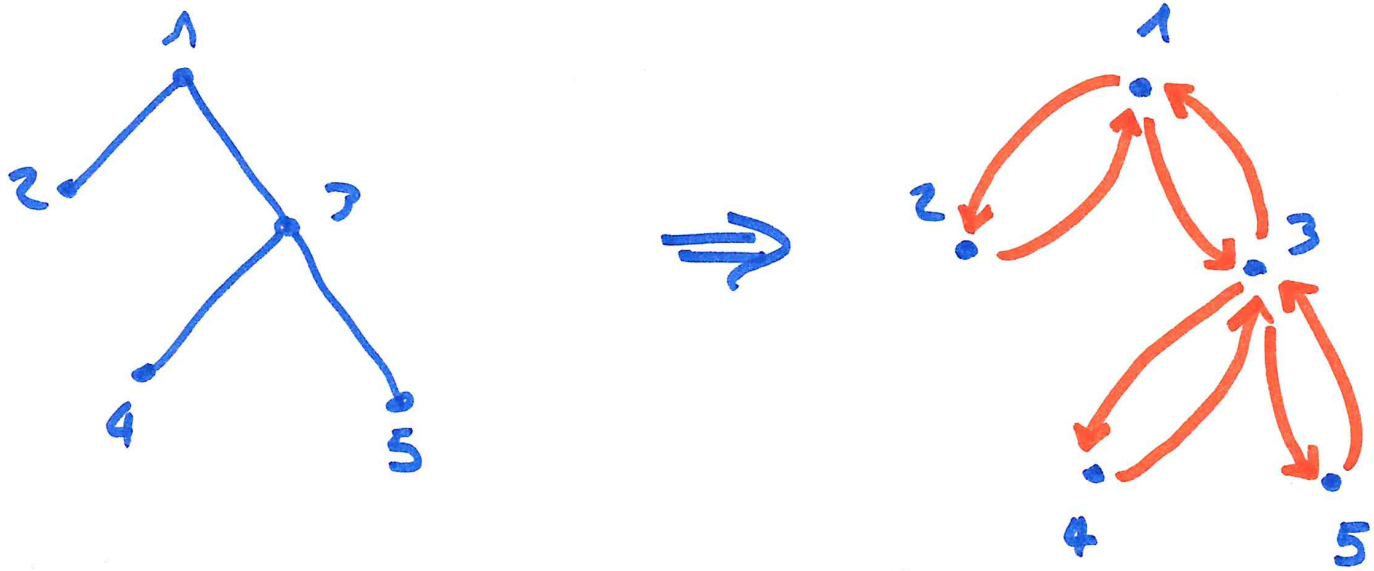
Domanda: come farlo con algoritmi paralleli efficienti?

IDEA: Usiamo le liste che si gestiscono bene in parallelo

(vedi Kogge-Stone: Vettore S = LISTA)

Primo passo:

Associo all'albero binario un ciclo Euleriano

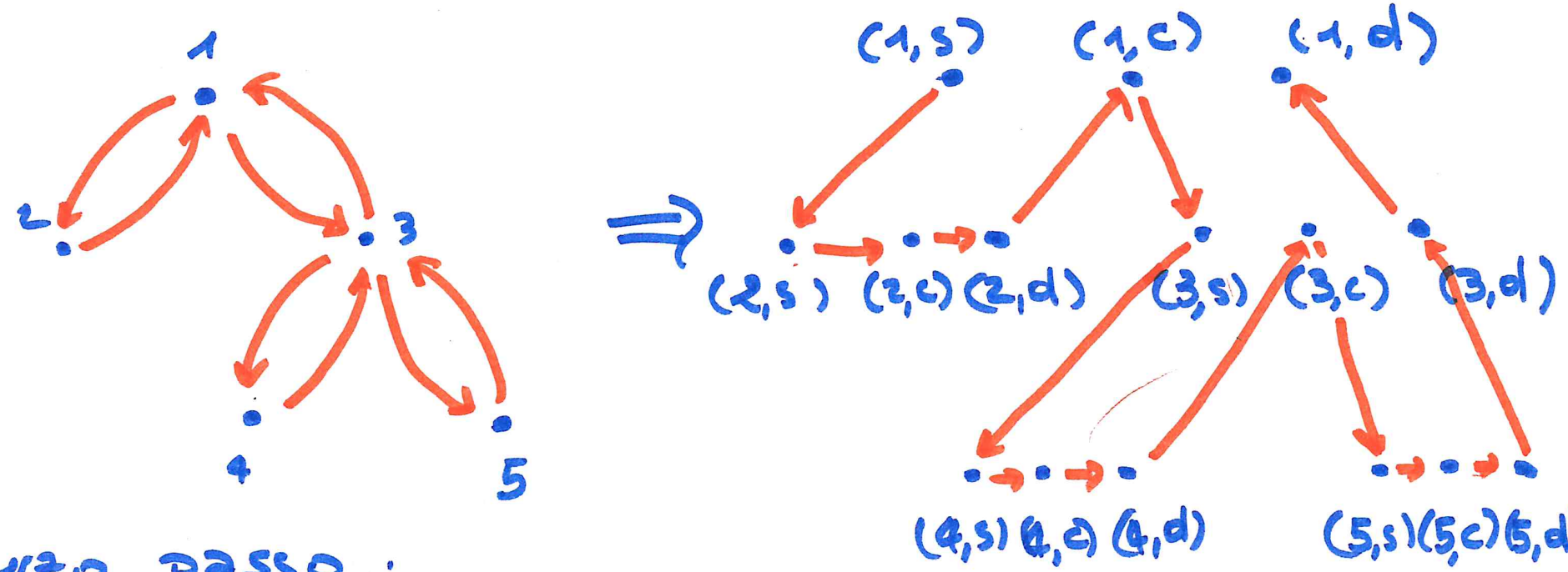


In questo modo navigo l'albero seguendo il ciclo Euleriano

Secondo passo:

Dal ciclo Euleriano al cammino Euleriano

Regola: ogni v viene espanso in (v, s), (v, c), (v, d)



Terzo passo:

Dal cammino Euleriano alla costruzione della LISTA:

$S((v, x))$ dove $1 \leq v \leq n$

$x \in \{s, c, d\}$

Costruzione dell'array S' dalla tabella

v	$sin(v)$	$des(v)$	$pad(v)$	
1	...		0	
2				
3	4	5	1	← nodo interno
4	0	0	3	} foglie
5	0	0	3	

- **v foglia**

$$S'[(v,s)] = (v,c)$$

$$S'[(v,c)] = (v,d)$$

$$S'[(v,d)] = \begin{cases} (pad(v), c) & \text{se } v = sin(pad(v)) \\ (pad(v), d) & \text{se } v = des(pad(v)) \end{cases}$$

⊙ v nodo interno

$$S[(v,s)] = (sin(v), s)$$

$$S[(v,c)] = (des(v), s)$$

$$S[(v,d)] = \begin{cases} (pad(v), c) & \text{se } v = sin(pad(v)) \\ (pad(v), d) & \text{se } v = des(pad(v)) \end{cases}$$

Diamo un algoritmo parallelo per S

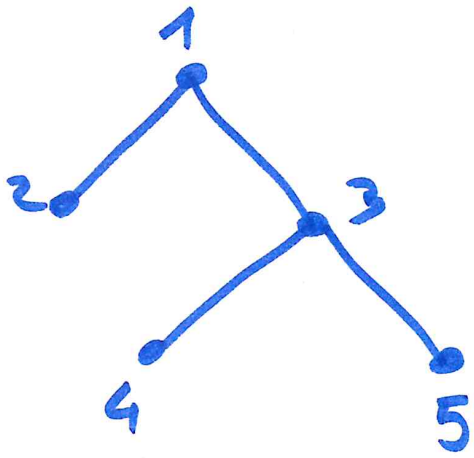
-) un processore per ogni $v \in V$ = $\forall$ riga della Tabella
-) letture non concorrenti con un piccolo accorgim.

$$\Downarrow$$

EREW usa $p(n) = n$ $\quad T(n, p(n)) = O(1)$

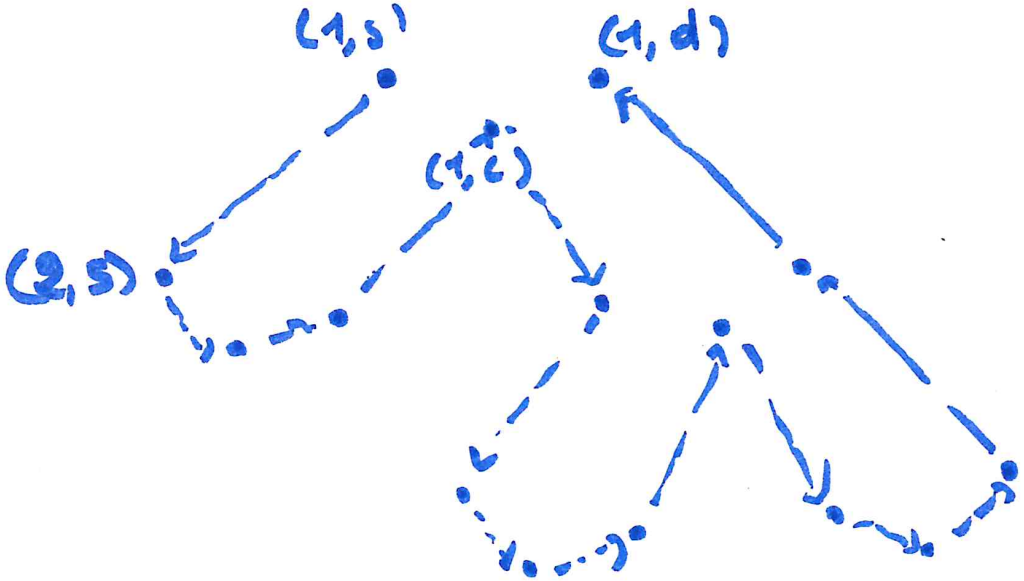
$p = n/\log n$ ←Wyllie→ $T = \log n$

Esercizio: Alberi come LISTE



⇒

v	sin(v)	oles(v)	pad(v)
1			
2			
3			
4			
5			



S'

(1,s)	(2,s)
(1,e)	
(1,d)	

1