Soluzione Progetto 2 ASD a.a. 2016/2017

La vendetta del Re Lich

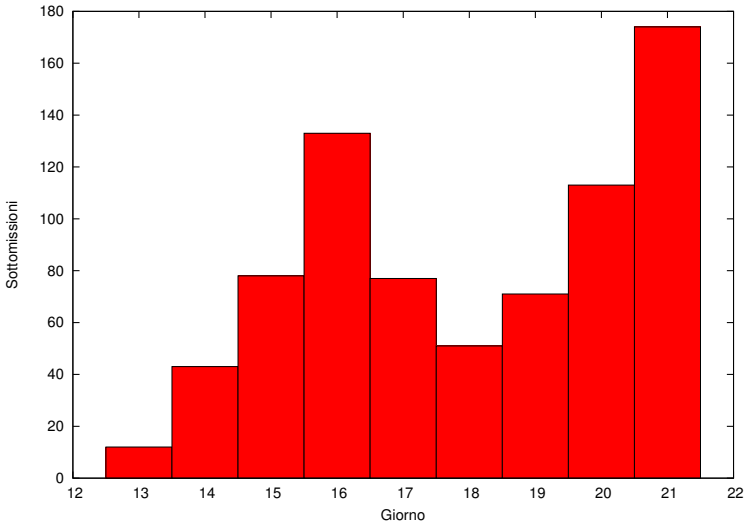
Alessio Guerrieri e Cristian Consonni

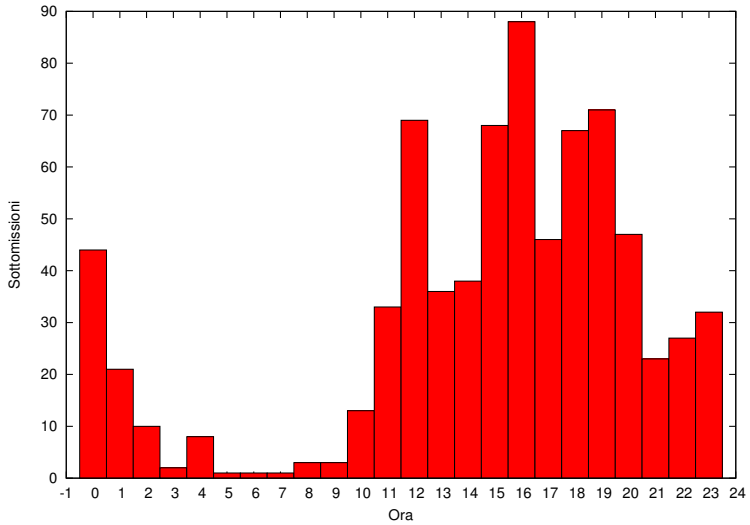
January 16, 2017

Statistiche

Statistiche

Numero sottoposizioni: 751





- 93 gruppi iscritti;
- 201 studenti;
- 16 ore di ricevimento (compresi i laboratori);
- 31 mail ricevute;

Risultati

Punteggi (classifica completa sul sito)

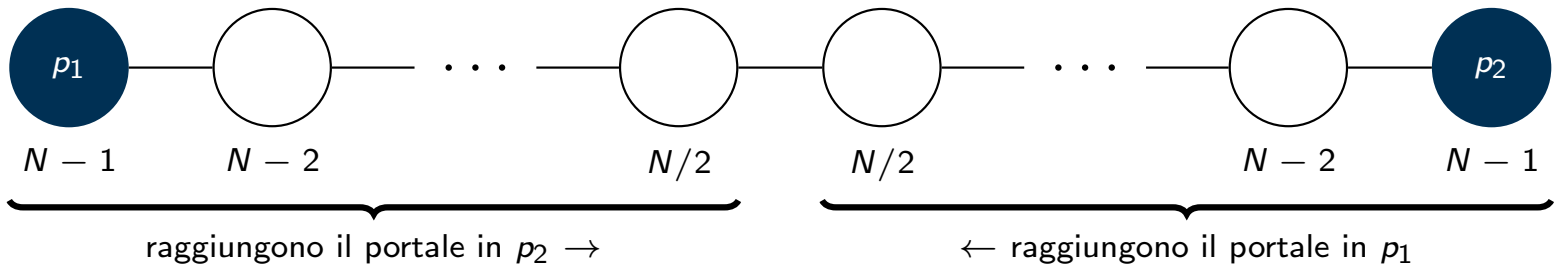
- $P < 30$ $\longrightarrow$ progetto non passato
- $30 \leq P < 65$ $\longrightarrow$ 1 punto bonus (29 gruppi)
- $65 \leq P \leq 100$ $\longrightarrow$ 2 punti bonus (39 gruppi)

Soluzione dei casi con linea non pesata - I

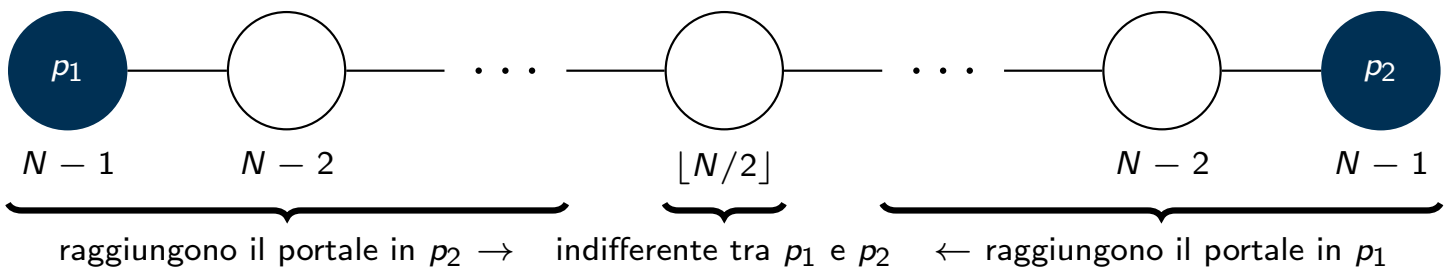
- sapendo il numero di nodi N è possibile ricavare la struttura della linea:
- ci sono 2 portali (=foglie dell'albero) alle estremità della linea;
- bisogna distinguere i casi in cui N è pari e in cui N è dispari;

Soluzione dei casi con linea non pesata - II

N pari

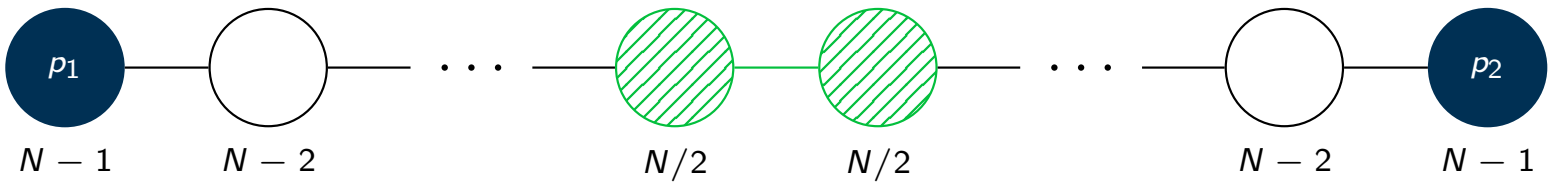


N dispari

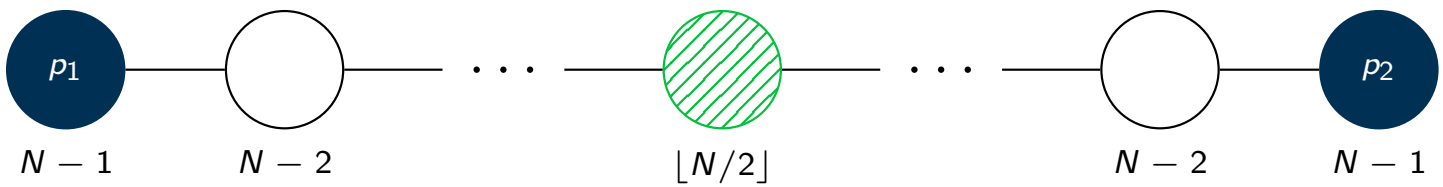


Soluzione dei casi con linea non pesata - II

N pari $(L = 0 \rightarrow C = 2)$



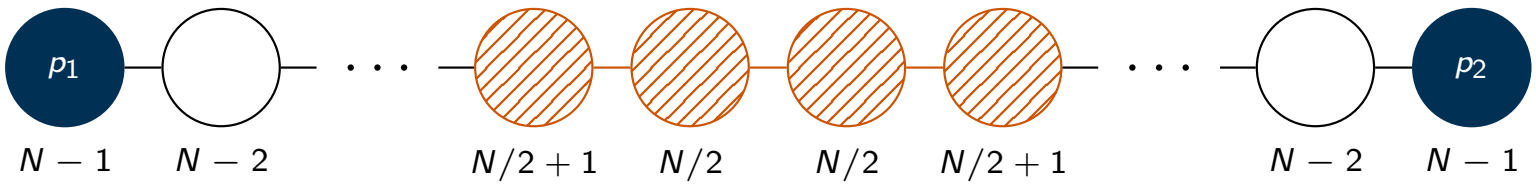
N dispari $(L = 0 \rightarrow C = 1)^{(*)}$



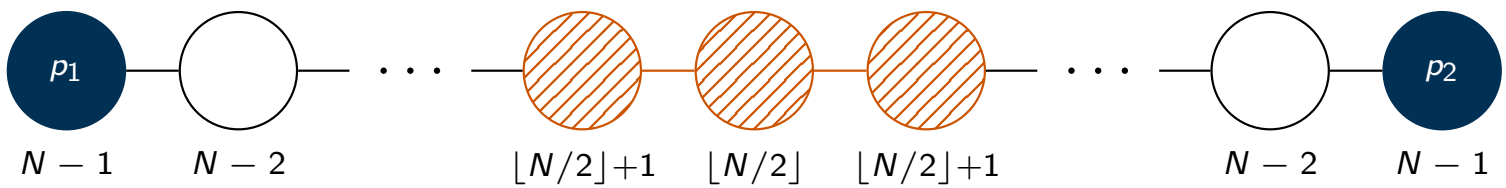
$^{(*)}$ È possibile posizionare un cavaliere su un nodo qualsiasi per ottenere una soluzione valida, il massimo numero di cavalieri posizionabili è 1.

Soluzione dei casi con linea non pesata - II

N pari $(L = 1 \rightarrow C = 4)$



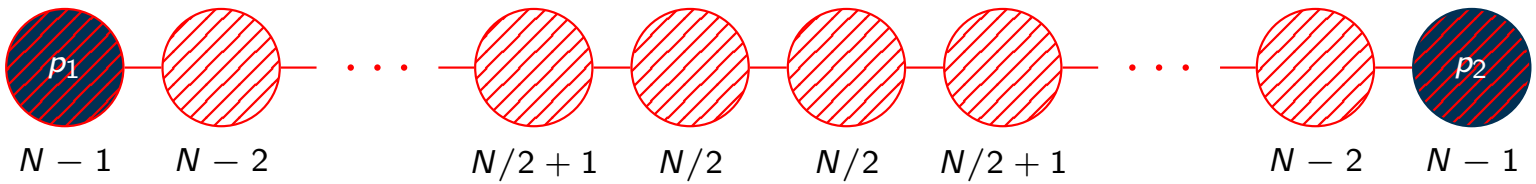
N dispari $(L = 1 \rightarrow C = 3)$



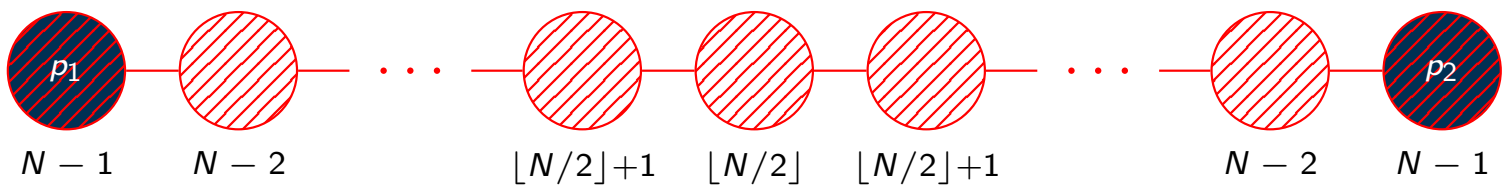
È possibile aggiungere 2 cavalieri per ogni incremento unitario di L.

Soluzione dei casi con linea non pesata - II

N pari $\left(L \geq \frac{N-1}{2} \rightarrow C = N\right)$



N dispari $\left(L \geq \frac{N-1}{2} \rightarrow C = N\right)$



Non è possibile posizionare più di N cavalieri.

Soluzione dei casi con linea non pesata - III

In sintesi

La soluzione si può esprimere tramite una formula chiusa, il numero massimo di cavalieri C_i è funzione di N e L_i:

$$C_i(N, L_i) = \begin{cases} min(N, 2(1 + L_i)) & \text{se } N \text{ è pari} \\ min(N, 1 + 2L_i) & \text{se } N \text{ è dispari} \end{cases} \quad (1)$$

- $\Rightarrow$ soluzione: `linea_nonpesata.cpp`, 29 SLOC
- $\Rightarrow$ complessità (tempo): $\Omega(1)$ (per ciascun L_i)
- $\Rightarrow$ complessità (memoria): $\Omega(1)$
- $\Rightarrow$ 30 punti

Soluzione dei casi con linea pesata

Intuizione

- ci sono 2 portali, che si trovano alle estremità della linea, sono gli unici nodi con un solo vicino;
- per ogni nodo si calcola la distanza dai 2 portali e si prende quella massima;
- si enumerano tutti i possibili segmenti della linea, C_i è la lunghezza del segmento più lungo per il quale valgono i requisiti;

$\Rightarrow$ soluzione: `linea_pesata.cpp`, 72 SLOC

$\Rightarrow$ complessità (tempo): $\Omega(N^2)$ (per ciascun L_i)

$\Rightarrow$ complessità (memoria): $\Omega(N)$

$\Rightarrow$ 50 punti

Soluzione del caso generico - intuizione

Intuizione

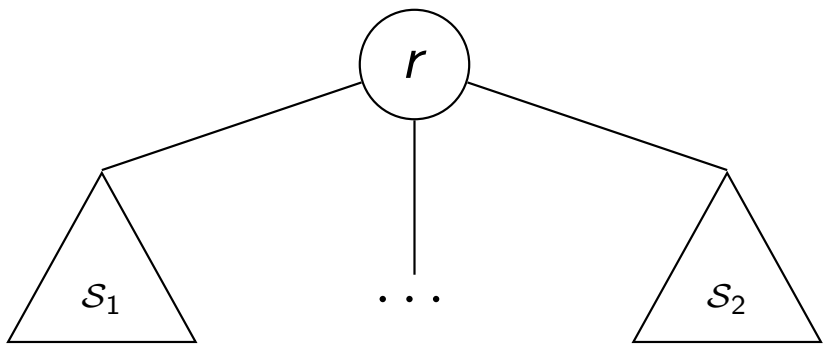
- per utilizzare la programmazione dinamica è necessario individuare i sottoproblemi/stati che permettono di risolvere il problema originale riducendolo ricorsivamente a sottoproblemi più piccoli e infine a dei casi base per i quali è possibile dare una risposta;
- gli alberi possono essere definiti ricorsivamente, questa definizione si presta bene all'applicazione della programmazione dinamica.

Soluzione del caso generico - definizione ricorsiva di albero

Albero (definizione ricorsiva)

Un albero $\mathcal{T}$ è una coppia (r, A) dove r è un valore (un *nodo*) e A è una lista di (sotto)alberi. La lista di sottoalberi può essere vuota ($A = \varnothing$). Il nodo r è la radice dell'albero.

Intuizione grafica



- $\mathcal{T} = (r, [\mathcal{S}_1, \mathcal{S}_2])$, dove $\mathcal{S}_1$ e $\mathcal{S}_2$ sono sottoalberi.

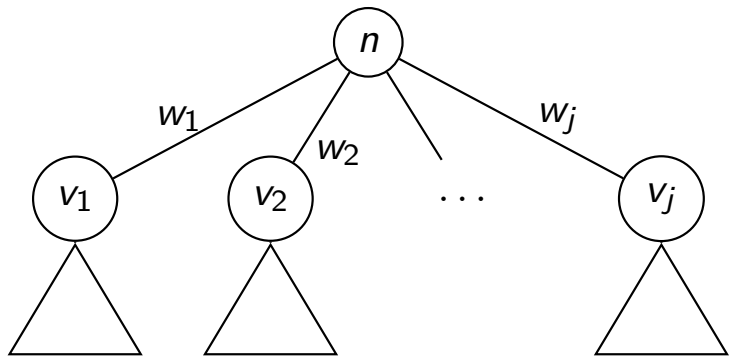
Soluzione del caso generico - calcolo distanza

Visita dell'albero

- per trovare la distanza dal portale più lontano si visita l'albero, la visita può essere definita in modo ricorsivo;

Intuizione grafica

- considero n come radice dell'albero:



- la distanza d_n è data da $d_n = \max\left(\{d_j + w_j\}_{j=1}^{|V(n)|}\right)$, dove d_j si calcola nel sottoalbero radicato in v_j;

Soluzione del caso generico - calcolo distanza

Visita dell'albero

- per trovare la distanza dal portale più lontano si visita l'albero, la visita può essere definita in modo ricorsivo;

Formalmente

- definizione del sottoproblema: lo stato è dato dal nodo corrente (n) e dal nodo padre (p):

$$S[n,p] = \begin{cases} 0 & \forall\, v \in V(n) \wedge v{=}p \\ max(\{w(v,n) + S[v,n]\}_v) & \forall\, v \in V(n) \wedge v{\neq}p \end{cases}$$

  dove $V(n)$ è l'insieme dei vicini di n;
- memoization con mappa `map<pair<int,int>, int>`;

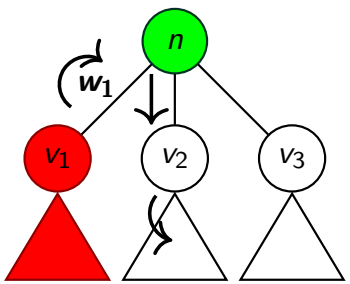
Soluzione del caso generico - selezione dei nodi e conteggio

Criterio di selezione ricorsivo

- se si scarta uno dei vicini v di n, allora viene scartato tutto il sottoalbero radicato in v, se si posiziona un cavaliere allora si analizza il sottoalbero;

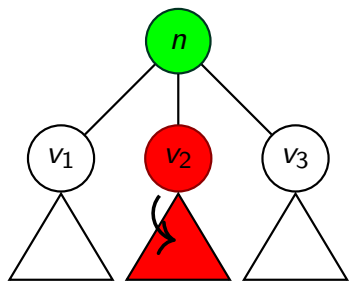
Intuizione grafica

- sul nodo n (radice) posiziono un cavaliere;
- considero che tutti i percorsi per i portali ($\rightarrow$) passino per n, quindi n è il nodo a distanza minima da ogni portale;



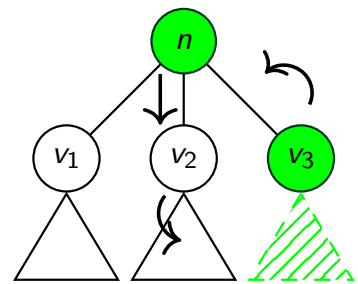
$d_{v_1} - d_n = w_1 > L_i$

scarto



$d_{v_2} < d_n$

scarto



altrimenti prendo v_3,

analizzo il sttoalbero

Soluzione del caso generico - selezione dei nodi e conteggio

Criterio di selezione ricorsivo

- se si scarta uno dei vicini v di n, allora viene scartato tutto il sottoalbero radicato in v, se si posiziona un cavaliere allora si analizza il sottoalbero;

Formalmente

- definizione del sottoproblema: lo stato è dato da nodo corrente (n), padre (p) e distanza della radice (d_r):

$$S[n,p,d_r] = \begin{cases} 0 & d_n < d_r \\ 0 & d_n - d_r > L_i \\ 1 + \sum\limits_{\substack{v \in V(n) \\ v \neq p}} S[v,n,d_r] & \text{altrimenti} \end{cases}$$

  dove $V(n)$ è l'insieme dei vicini di n;
- per ogni nodo n, si parte da $S[n,n,d_n]$

Soluzione del caso generico

Abbiamo scomposto il problema in due parti:

- calcolo della distanza dal portale più lontano per ogni nodo: complessità temporale $\Omega(N)$ con memoization;
- selezione del massimo numero di nodi che rispettano i requisiti: complessità temporale $\Omega(N^2)$;

$\Rightarrow$ C_i è il conteggio dei nodi selezionati;

$\Rightarrow$ soluzione:`albero.cpp`, 68 SLOC

$\Rightarrow$ complessità (tempo): $\Omega(N^2)$ (per ciascun L_i)

$\Rightarrow$ complessità (memoria): $\Omega(N)$

$\Rightarrow$ 85 punti

Soluzione efficiente del caso generico

Intuizione

1. calcolo della distanza dal portale più lontano:
   - nell'algoritmo precedente si visitava l'albero partendo da ogni nodo e si utilizzava la memoization per evitare di ripetere la visita su sottoalberi già visitati;

   $\Rightarrow$ possiamo rendere più efficiente l'algoritmo visitando solo i nodi vicini e visitando consecutivamente ogni sottoalbero;
2. posizionamento e conteggio dei cavalieri:
   - nell'algoritmo precedente si ciclava su tutti i nodi dell'albero, prendendo progressivamente ogni nodo come radice;
   - si assumeva che la radice fosse il nodo a distanza minima dai portali;

   $\Rightarrow$ si radica l'albero nel nodo la cui distanza dal portale più lontano è minima;

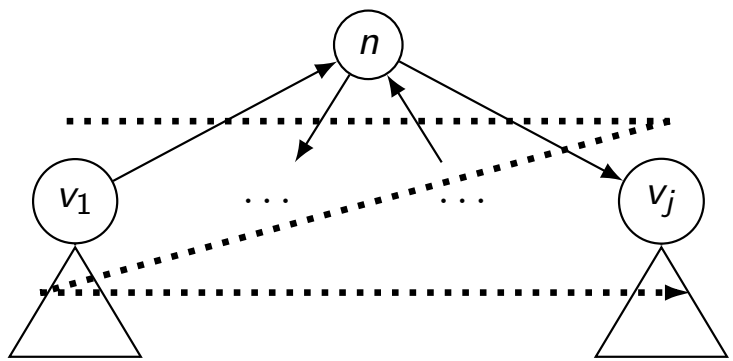
   $\Rightarrow$ si usa una struttura dati efficiente per verificare che due nodi facciano parte di uno stesso gruppo di nodi contigui con un cavaliere;

Soluzione efficiente del caso generico - calcolo distanza

Visita dell'albero

- si visita l'albero con una BFS da ogni nodo, definendo il sottoproblema sugli archi, gli archi sono orientati nella direzione indicata nell'input;
- con la memoization l'albero viene visitato una volta sola;

Intuizione grafica



Soluzione efficiente del caso generico - calcolo distanza

Visita dell'albero

- si visita l'albero con una BFS da ogni nodo, definendo il sottoproblema sugli archi, gli archi sono orientati nella direzione indicata nell'input;
- con la memoization l'albero viene visitato una volta sola;

Formalmente

- definizione del sottoproblema: lo stato è dato dall'arco e $(p \to n)$, il nodo corrente è quello di arrivo $(n := e.end)$ e il nodo padre è quello di partenza $(p := e.start)$.

$$S[e_{p \to n}] = \begin{cases} 0 & \forall\, e_{n \to v},\ v \in V(n) \wedge v = p \\ max(\{w(e_{n \to v}) + S[e_{n \to v}]\}_v) & \forall\, e_{n \to v},\ v \in V(n) \wedge v \neq p \end{cases}$$

  dove $\{e_{n \to v}\}_v$ è l'insieme degli archi uscenti da n;
- memoization con vector (`vector<int>`);

Soluzione del caso generico - selezione dei nodi e conteggio

Selezione iterativa dei cavalieri

Vogliamo costruire il più grande gruppo S di nodi contigui tali che $\max(\{d_n\}_{n \in S}) - \min(\{d_n\}_{n \in S}) \leq L_i$:

- si radica l'albero nel nodo la cui distanza dal portale più lontano è minima e si orientano gli archi dai padri ai figli;
- inizialmente:
  - ogni nodo è in un gruppo a sè stante;
  - si mette un cavaliere su ogni nodo;
- si analizza ogni nodo in ordine da quello a distanza massima dalla sorgente più lontana. Dato il nodo corrente (n):
  - si uniscono nello stesso gruppo ogni nodo con i suoi figli (escludendo il nodo padre da cui si proviene). I gruppi con nodi in comune si contano insieme;
  - si eliminano i cavalieri dai nodi qualora $d_v - d_n > L_i$;

  $\Rightarrow$ C_i è il conteggio del numero massimo di cavalieri presenti in un gruppo in un dato momento;

Soluzione efficiente del caso generico - selezione dei nodi e conteggio

L'algoritmo funziona grazie alla proprietà seguente.

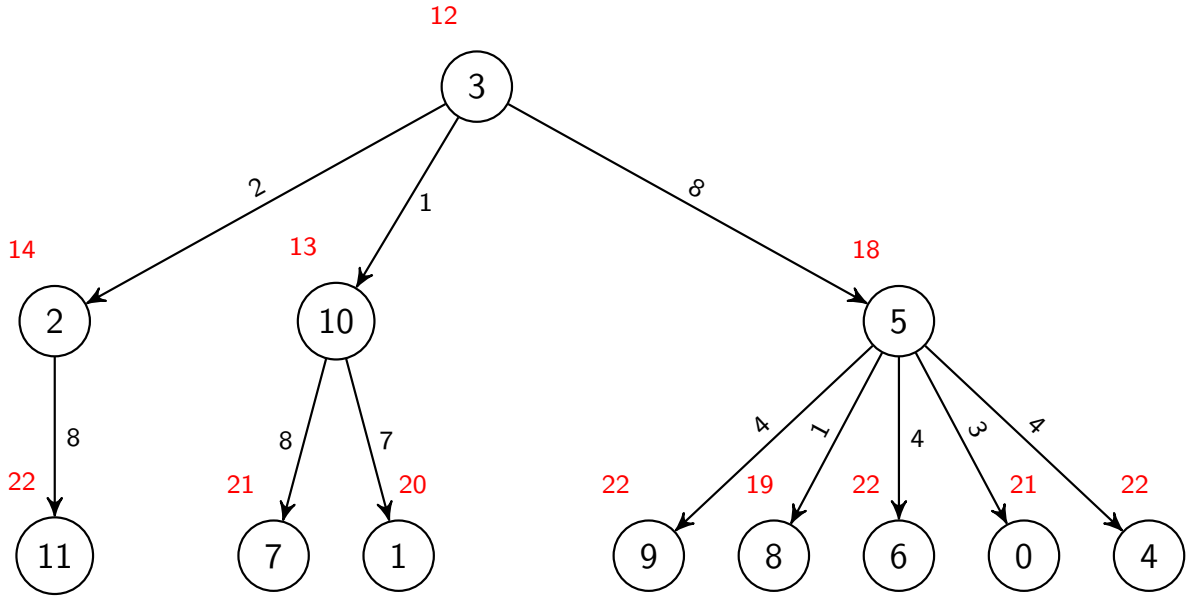
Proprietà dell'albero delle distanze dai portali

Radicando l'albero nel nodo a distanza minima e orientando i nodi dalla radice verso le foglie si ha che in ogni relazione padre-figlio il padre ha una distanza minore del figlio.

Si dimostra per induzione, a partire dalla radice che essendo il nodo a distanza minima per definizione rispetta la proprietà di cui sopra. Per ogni sottoalbero si può mostrare che il percorso dal nodo per il portale più lontano passa dal nodo in cui è radicato il sottoalbero. Quindi i figli hanno distanze maggiori dai portali rispetto ai padri.

Soluzione del caso generico - selezione dei nodi e conteggio

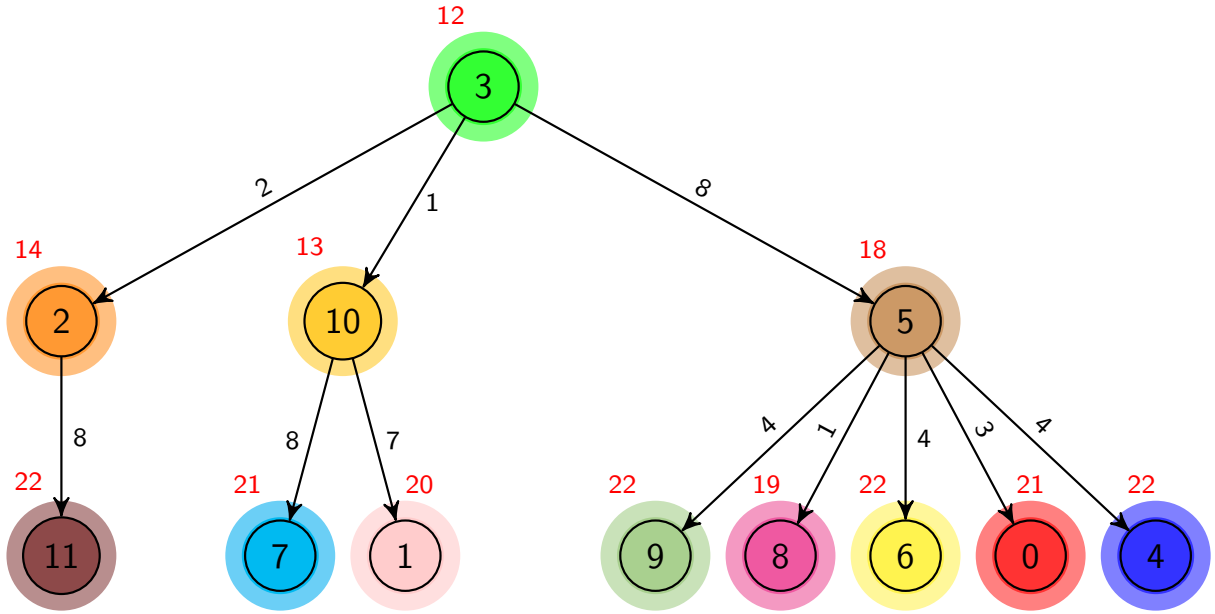
Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- si radica l'albero nel nodo a distanza minima;

Soluzione del caso generico - selezione dei nodi e conteggio

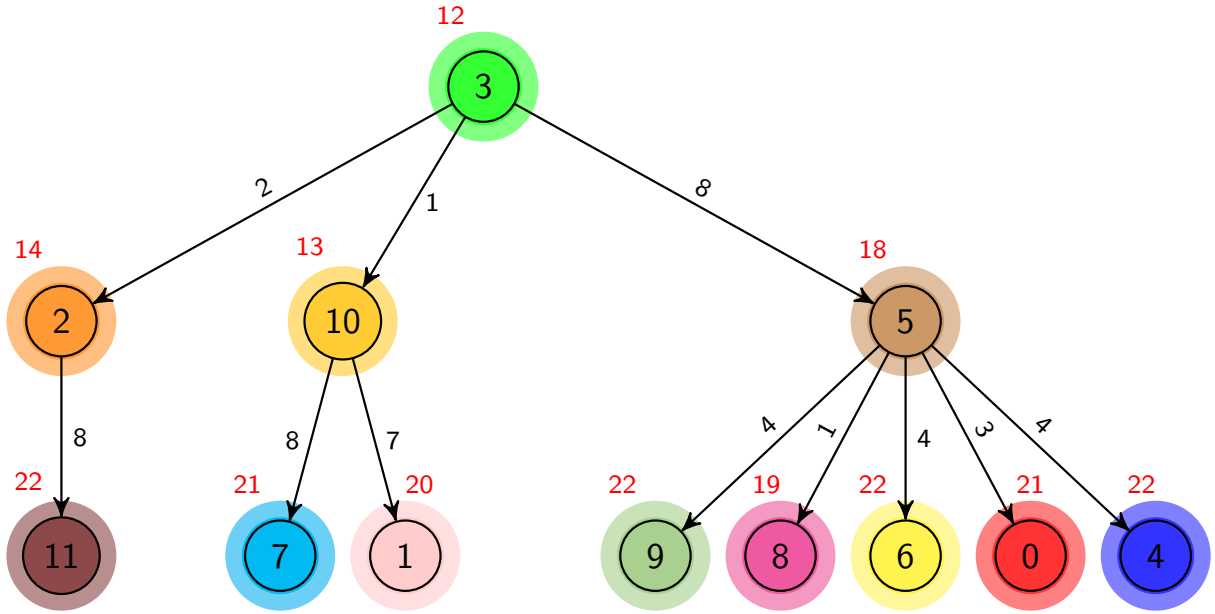
Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- si radica l'albero nel nodo a distanza minima;
- ogni colore è un gruppo diverso. I nodi pieni hanno un cavaliere posizionato. Mettiamo un cavaliere su ogni nodo.

Soluzione del caso generico - selezione dei nodi e conteggio

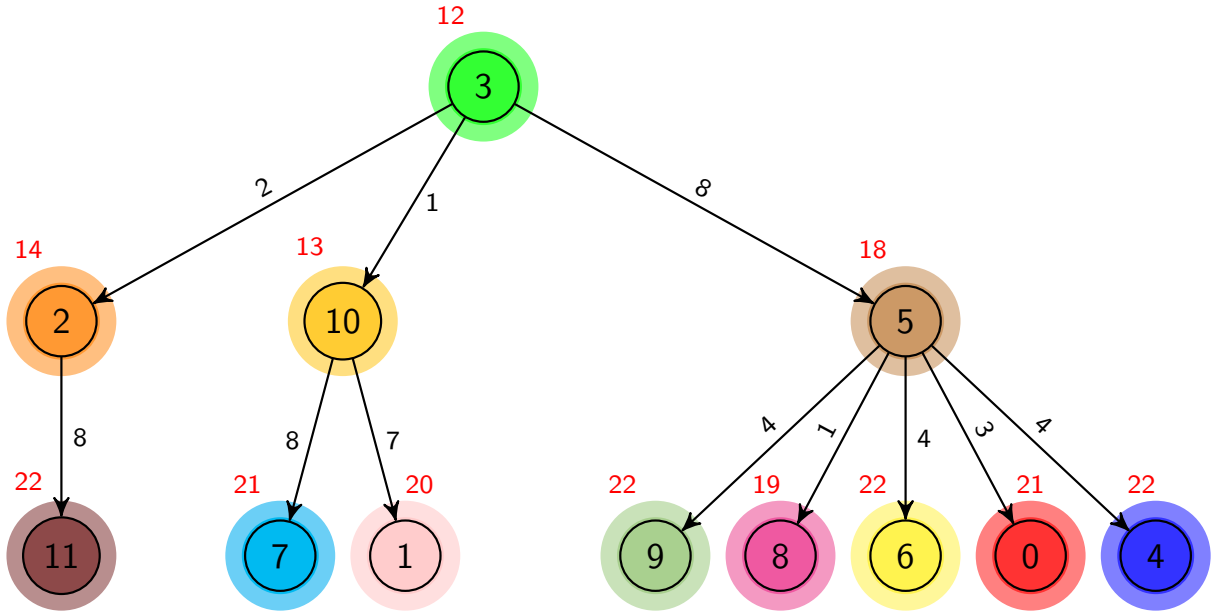
Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- si radica l'albero nel nodo a distanza minima;
- ogni colore è un gruppo diverso. I nodi pieni hanno un cavaliere posizionato. Mettiamo un cavaliere su ogni nodo.
- il massimo conteggio, ovvero C provvisorio, è 1;

Soluzione del caso generico - selezione dei nodi e conteggio

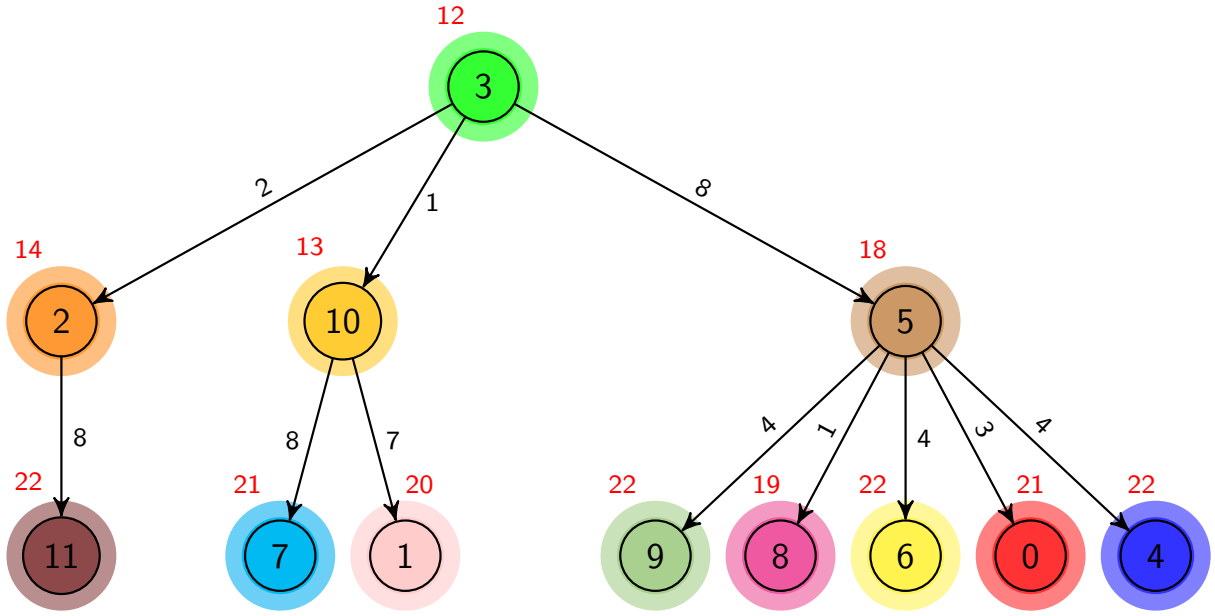
Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- si radica l'albero nel nodo a distanza minima;
- ogni colore è un gruppo diverso. I nodi pieni hanno un cavaliere posizionato. Mettiamo un cavaliere su ogni nodo.
- il massimo conteggio, ovvero C provvisorio, è 1;
- i nodi vengono analizzati nell'ordine $[11, 9, 6, 4, 7, 1, 8, 5, 2, 10, 3]$

Soluzione del caso generico - selezione dei nodi e conteggio

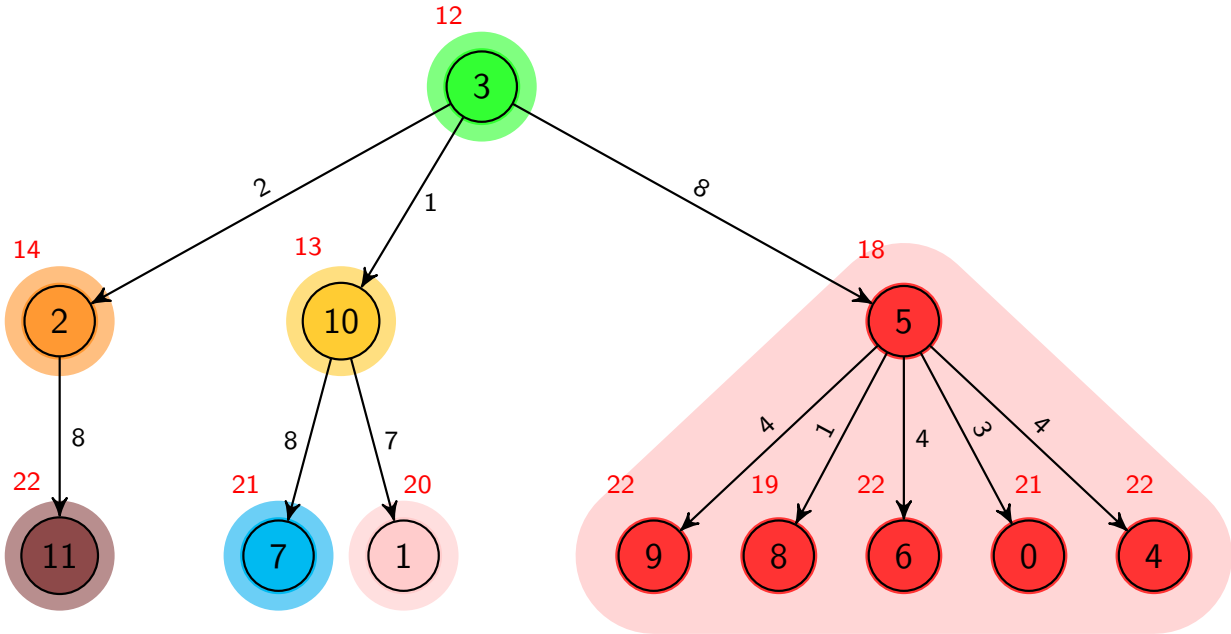
Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- si radica l'albero nel nodo a distanza minima;
- ogni colore è un gruppo diverso. I nodi pieni hanno un cavaliere posizionato. Mettiamo un cavaliere su ogni nodo.
- il massimo conteggio, ovvero C provvisorio, è 1;
- i nodi vengono analizzati nell'ordine $[11, 9, 6, 4, 7, 1, 8, 5, 2, 10, 3]$
- i nodi $11, 9, 6, 4, 7, 1, 8$ non hanno figli quindi non succede nulla;

Soluzione del caso generico - selezione dei nodi e conteggio

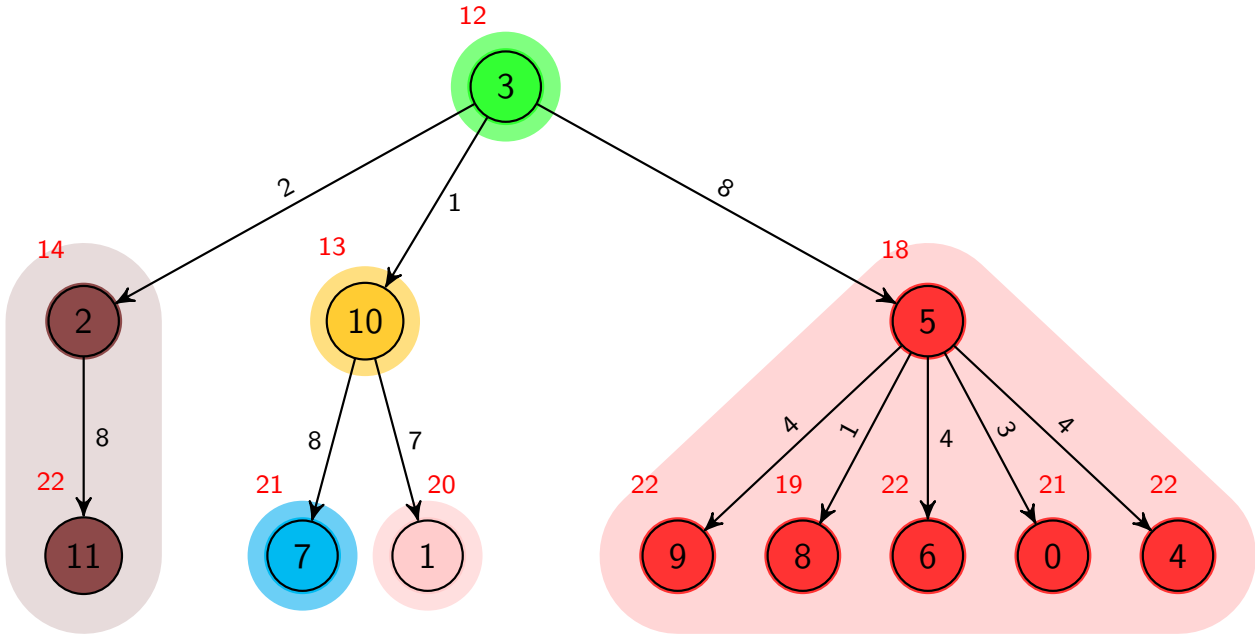
Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- quando analizziamo 5, lo si unisce con tutti i suoi figli in un gruppo, $C = 6$;

Soluzione del caso generico - selezione dei nodi e conteggio

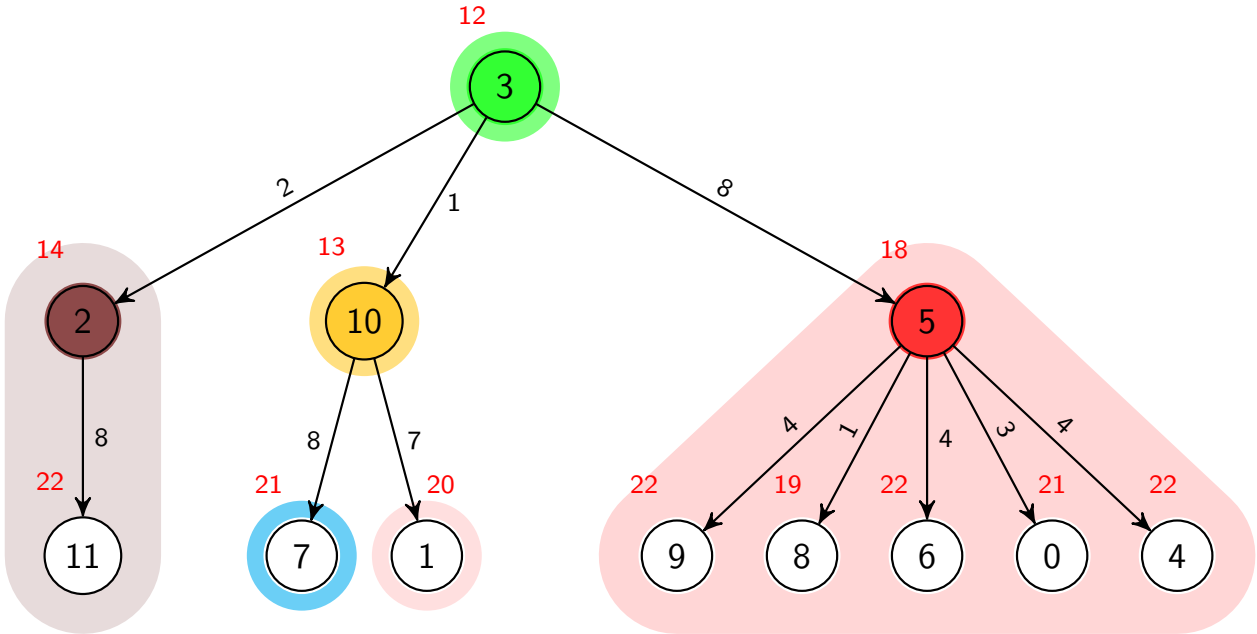
Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- quando analizziamo 5, lo si unisce con tutti i suoi figli in un gruppo, $C = 6$;
- quando analizziamo 2:
  - si crea il gruppo $\{2, 11\}$.

Soluzione del caso generico - selezione dei nodi e conteggio

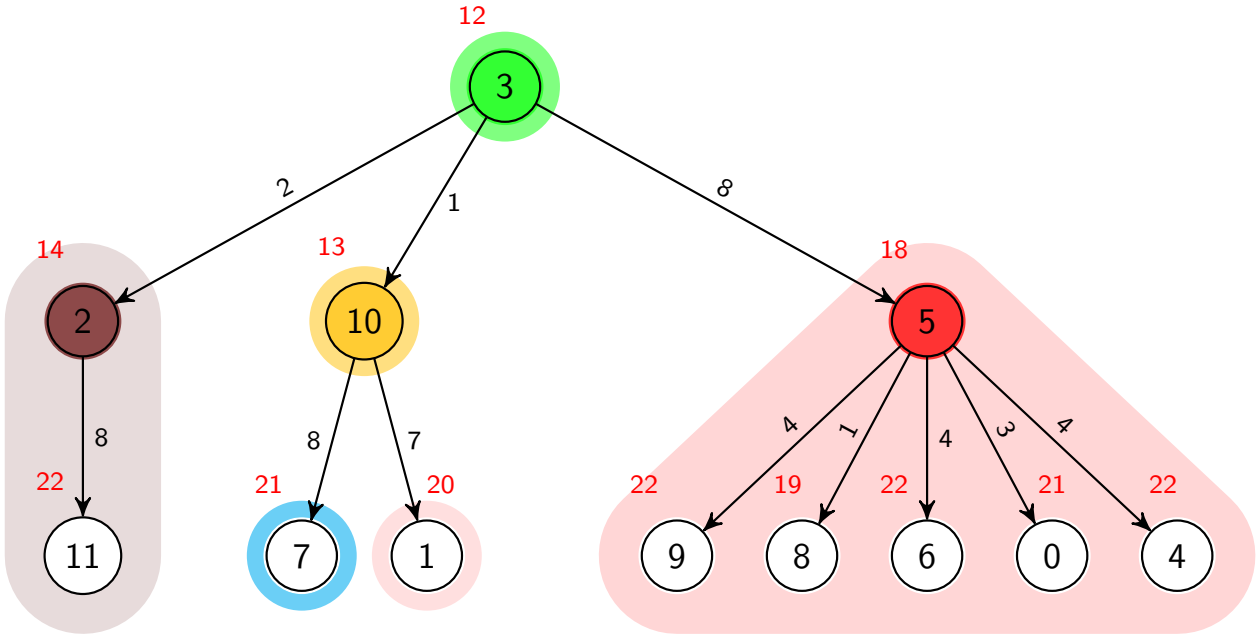
Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- quando analizziamo 5, lo si unisce con tutti i suoi figli in un gruppo, $C = 6$;
- quando analizziamo 2:
  - si crea il gruppo $\{2, 11\}$.
  - si eliminano i cavalieri dai nodi la cui distanza d è tale che $d - d_2 > 4$ ovvero $d - 14 > 4$.

Soluzione del caso generico - selezione dei nodi e conteggio

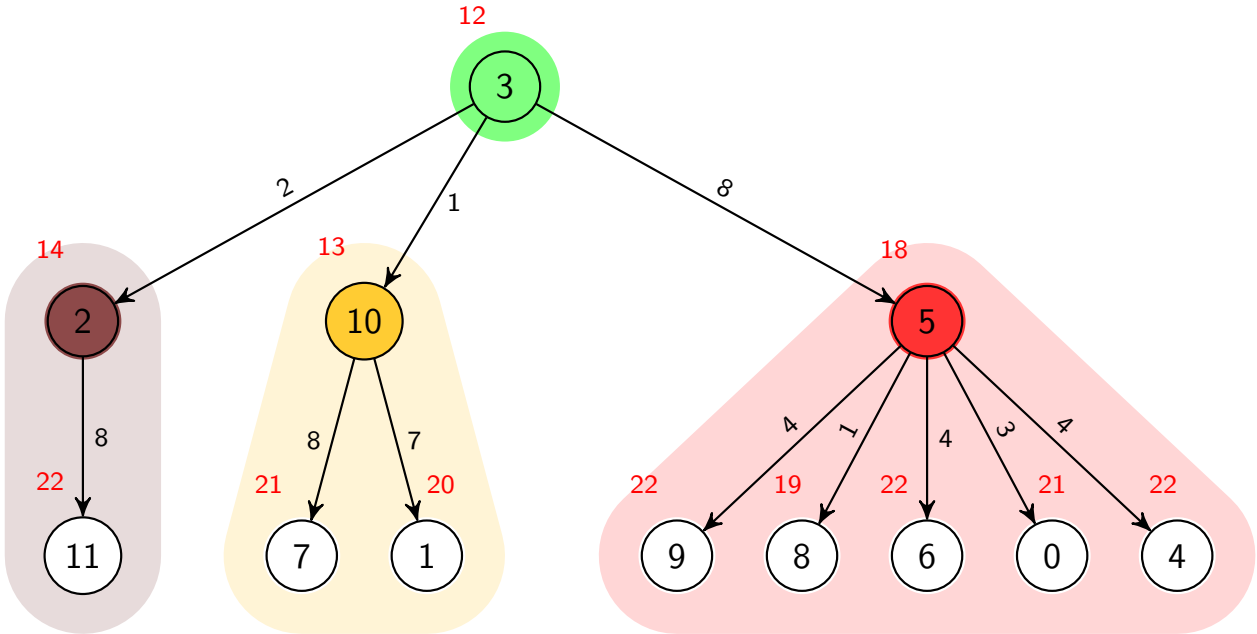
Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- quando analizziamo 5, lo si unisce con tutti i suoi figli in un gruppo, $C = 6$;
- quando analizziamo 2:
  - si crea il gruppo $\{2, 11\}$.
  - si eliminano i cavalieri dai nodi la cui distanza d è tale che $d - d_2 > 4$ ovvero $d - 14 > 4$.
  - in questo caso il gruppo $\{2, 11\}$ ha un solo cavaliere.

Soluzione del caso generico - selezione dei nodi e conteggio

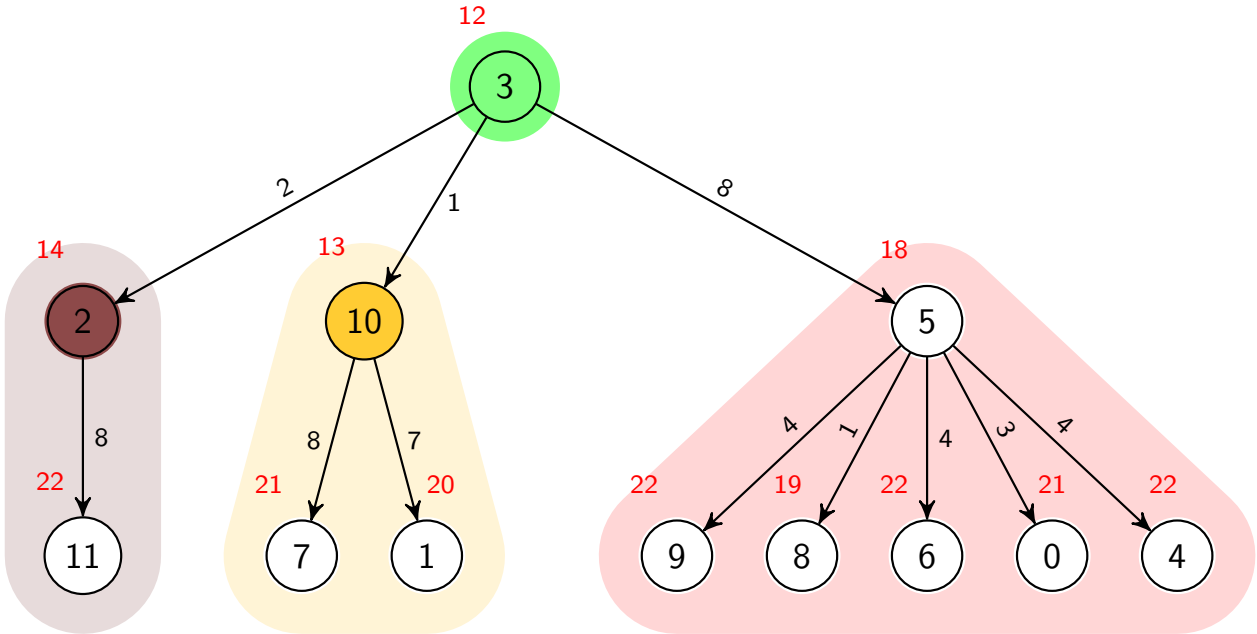
Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- Analogamente per 10:
  - si crea il gruppo $\{1, 7, 10\}$.

Soluzione del caso generico - selezione dei nodi e conteggio

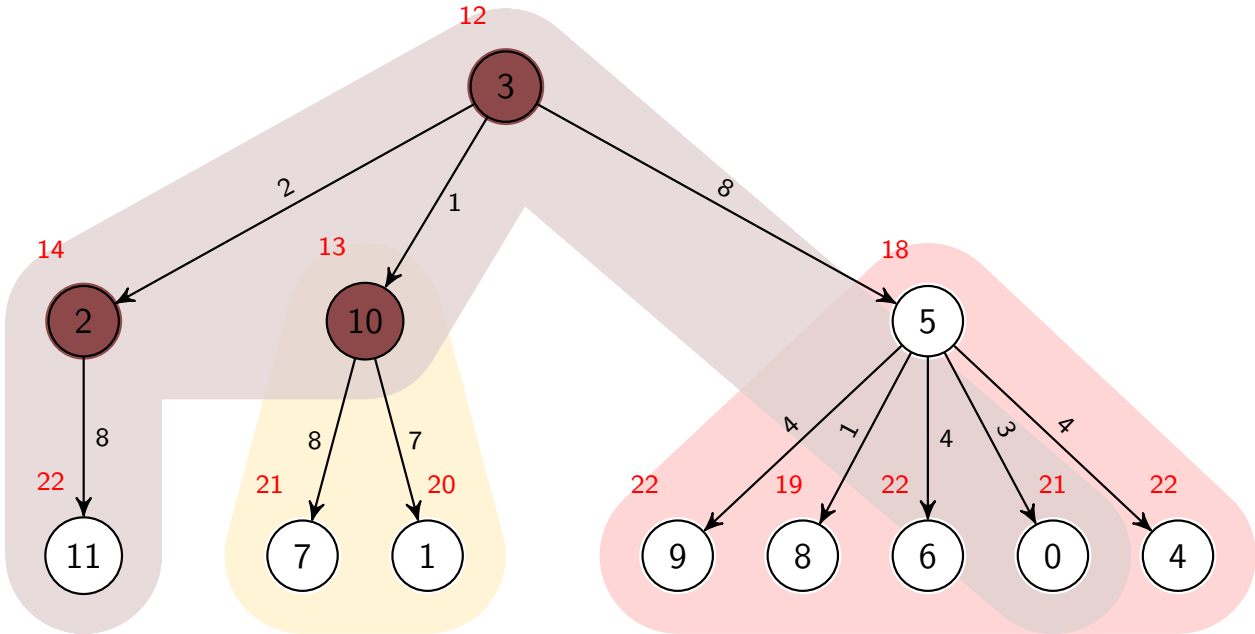
Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- Analogamente per 10:
  - si crea il gruppo $\{1, 7, 10\}$.
  - si eliminano i cavalieri da tutti i nodi la cui distanza d è tale che $d - d_{10} > 4$ ovvero $d - 13 > 4$.

Soluzione del caso generico - selezione dei nodi e conteggio

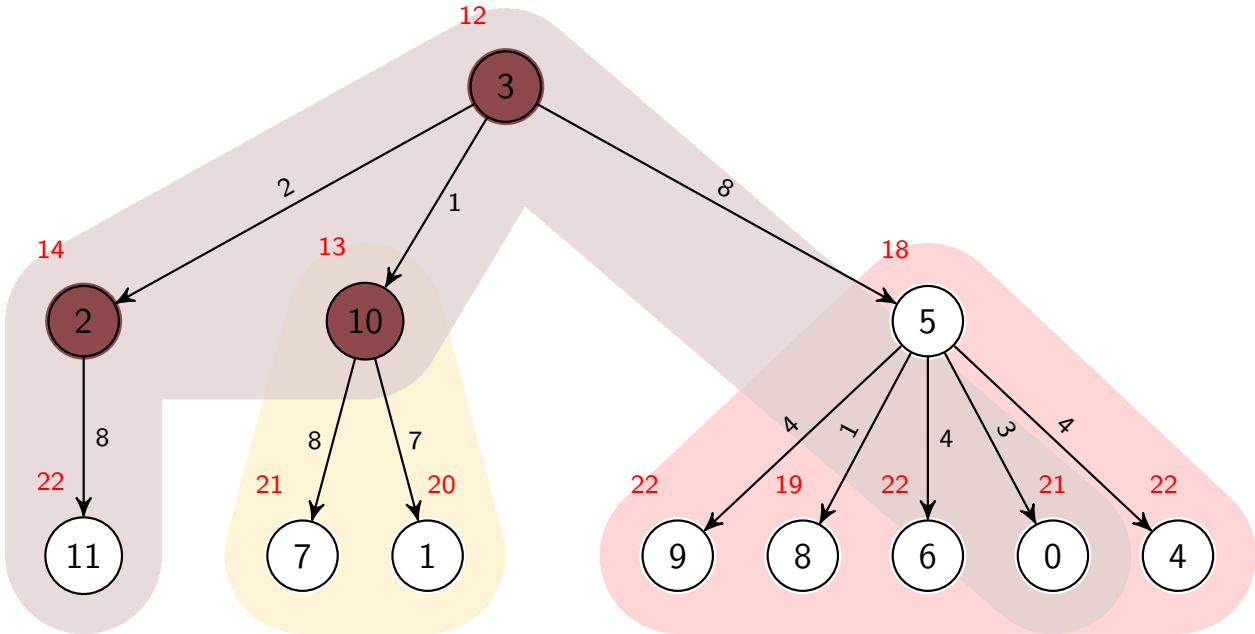
Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- Quando si analizza 3:
  - tutti i gruppi vengono uniti tra loro.

Soluzione del caso generico - selezione dei nodi e conteggio

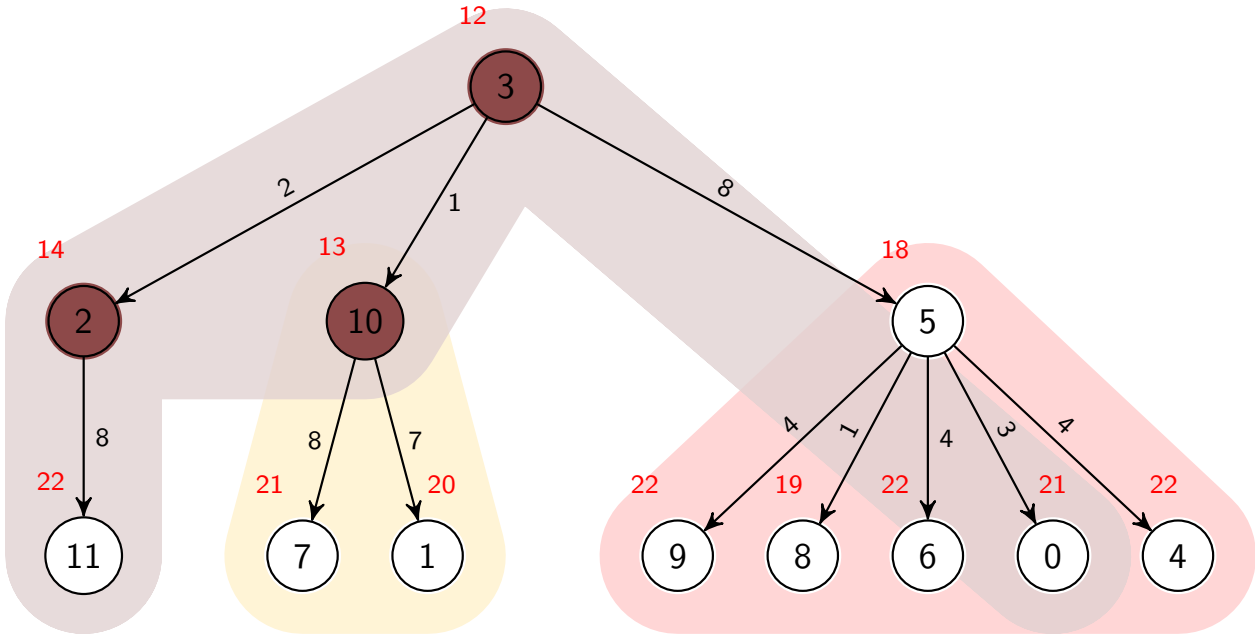
Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- Quando si analizza 3:
  - tutti i gruppi vengono uniti tra loro.
  - ci sono solo tre cavalieri.

Soluzione del caso generico - selezione dei nodi e conteggio

Eesecuzione sull'input del primo esempio dato nel testo, $L = 4$



- Quando si analizza 3:
  - tutti i gruppi vengono uniti tra loro.
  - ci sono solo tre cavalieri.

$\Rightarrow$ non ci sono più nodi da analizzare. L'algoritmo termina con $C = 6$.

Soluzione del caso generico - selezione dei nodi e conteggio

Formalmente

- si utilizza una struttura dati **union-find** per creare gruppi disgiunti di nodi;
  - operazione `union`: fondo due gruppi. Nell'esempio si colorano tutti nodi che appartegono a quel gruppo;
  - operazione `find`: trovo qual è il rappresentante del gruppo a cui appartiene il nodo. Nell'esempio restituisce il colore identificativo del gruppo;
- svariate applicazioni: reti, image processing, ecc.

operazione	`union`	`find`
complessità	$\Omega\left(\log^*(N)\right)$	$\Omega\left(\log^*(N)\right)$

La funzione $\log^*$ è il *"logaritmo iterato"* (o *"log star"*), ovvero $\log^*(N)$ è il numero di volte che la funzione logaritmo deve essere applicata iterativamente a partire da N prima che il risultato sia minore o uguale a 1. Cresce **molto** lentamente, con $N = 10^{80}$ (numero stimato di atomi nell'universo) $\rightarrow \log^*(10^{80}) = 5$.

Soluzione efficiente del caso generico

Analisi della complessità

- calcolo della distanza dal portale più lontano per ogni nodo: complessità temporale $\Omega(N)$ con memoization;
- selezione e conteggio del massimo numero di nodi che rispettano i requisiti:
  - ordinamento dei nodi rispetto alla distanza dal portale più lontano: $\Omega(N \log(N))$
  - conteggio dei cavalieri per ciascun L_i: $\Omega\left(N \log^*(N)\right)$

$\Rightarrow$ soluzione:`albero_lineare.cpp`, 140 SLOC

$\Rightarrow$ complessità (tempo): $\Omega(N \log(N))$

$\Rightarrow$ complessità (memoria): $\Omega(N)$

$\Rightarrow$ 100 punti

Note finali

- Classifiche e sorgenti sul sito (controllate i numeri di matricola):
  - `http://judge.science.unitn.it/slides/asd16/classifica_prog2.pdf`
  - Assumiamo gli stessi gruppi, in caso di cambiamenti scrivetemi (`cristian.consonni@unitn.it`)
- Per accedere allo scritto è necessario aver passato almeno un progetto.

Extra

Link utili

Slides di approfondimento su union-find:

- slides prof. Montresor:

  `http://disi.unitn.it/~montreso/asd/lucidi/10-heapmfset.pdf`

- appunti dettagliati (italiano):

  `http://www.sci.unich.it/~acciaro/ProblemaUnionFind.pdf`

- implementazione in Java e applicazioni (inglese):

  `https://www.cs.princeton.edu/~rs/AlgsDS07/01UnionFind.pdf`

Credits

- i dati sulle SLOC sono stati generati usando il programma `SLOCCount` di David A. Wheeler.