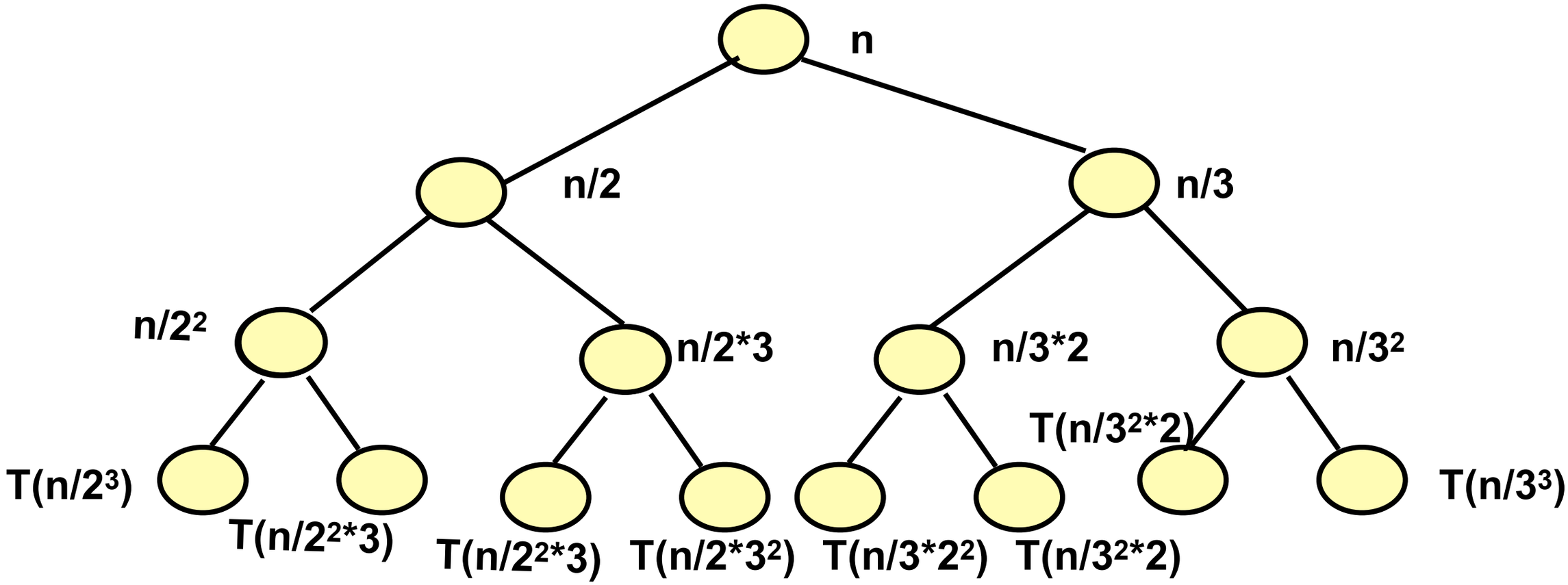
$T(n) = T(n/2) + T(n/3) + n$


n
n/2
n/3
n/2²
n/2*3
n/3*2
n/3²
T(n/3²*2)
T(n/2³)
T(n/2²*3)
T(n/2²*3)
T(n/2*3²)
T(n/3*2²)
T(n/3²*2)
T(n/3³)

Analisi algoritmi ricorsivi

Posto j-i+1 = n

```
Esercizio1(A,i,j)
 prec: A è un vettore di n elementi e i
e j sono indici compresi tra 1 e n.
1. k ← i
2. while (k ≤ j) do
3.          h ← i
4.      while (h ≤ j) do                    eseguito n volte
5.                  A[h] ← A[h] +1
6.                  h ← h +1
7.          k ← k +1
```

```
Esercizio2 (A,i, j)
if ( i<j )
    then numElem ← j - i+1
Esercizio2( A, i, i + numElem/4)
Esercizio2( A, j - numElem/4,j )
Esercizio1( A, i, j )
```

Analisi algoritmi ricorsivi

Posto j-i+1 = n

```
Esercizio1(A,i,j)
 prec: A è un vettore di n elementi e i
e j sono indici compresi tra 1 e n.
1. k = i
2. while (k ≤ j) do
3.          h = i
4.      while (h ≤ j) do
5.                  A[h] = A[h] +1
6.                  h =h +1
7.          k = k +1
```

eseguito n volte, quindi in $\Theta(n^2)$

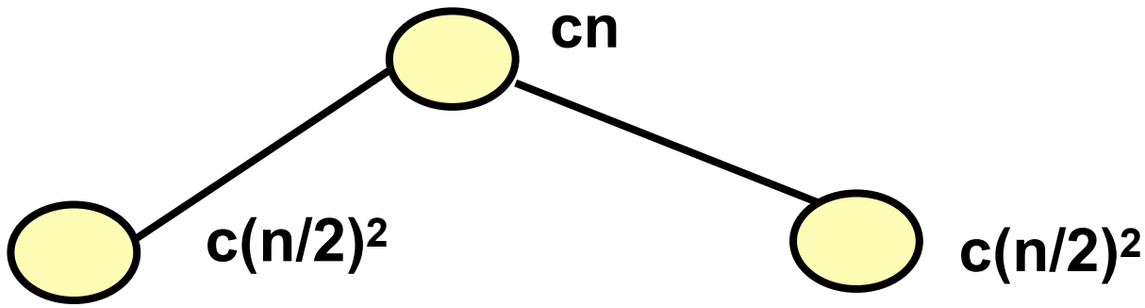
eseguito n volte

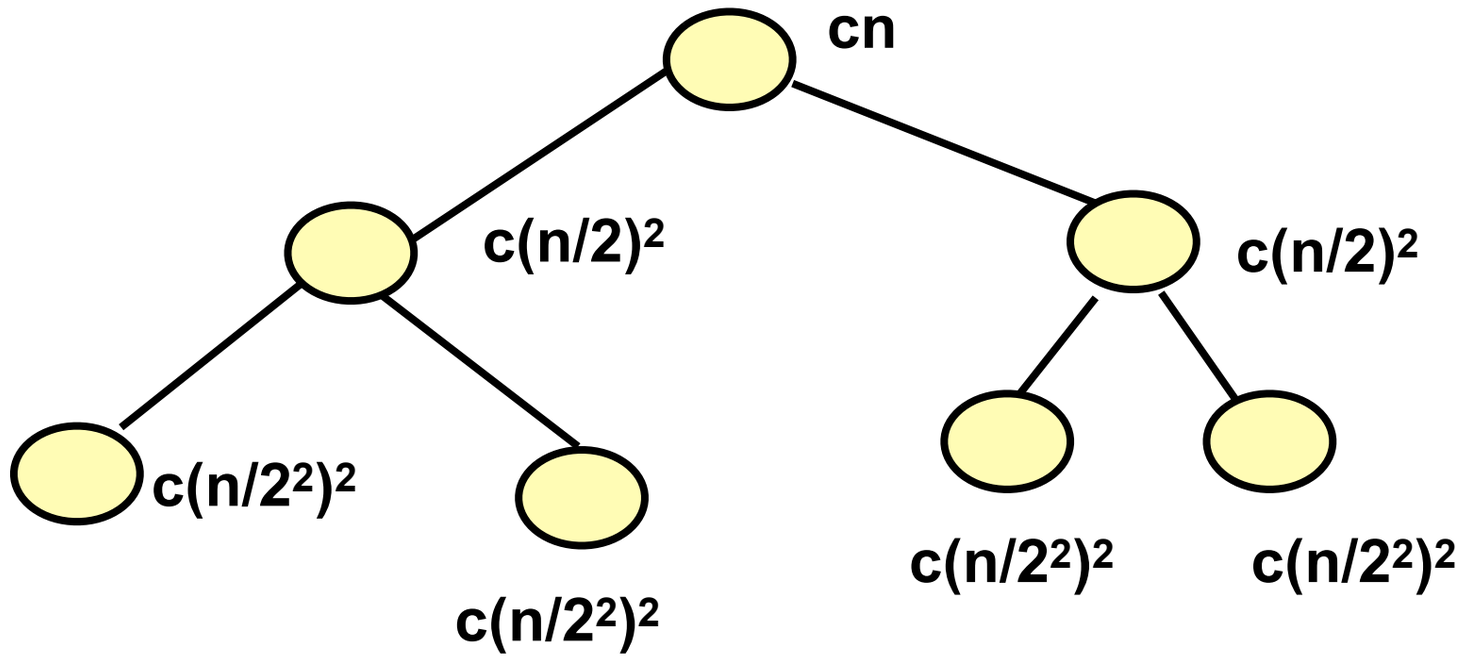
```
Esercizio2 (A,i, j)
if ( i < j) then n = j - i+1
Esercizio2( A, i, i + n/4)
Esercizio2( A, j - n/4,j )
Esercizio1( A, i, j )
```

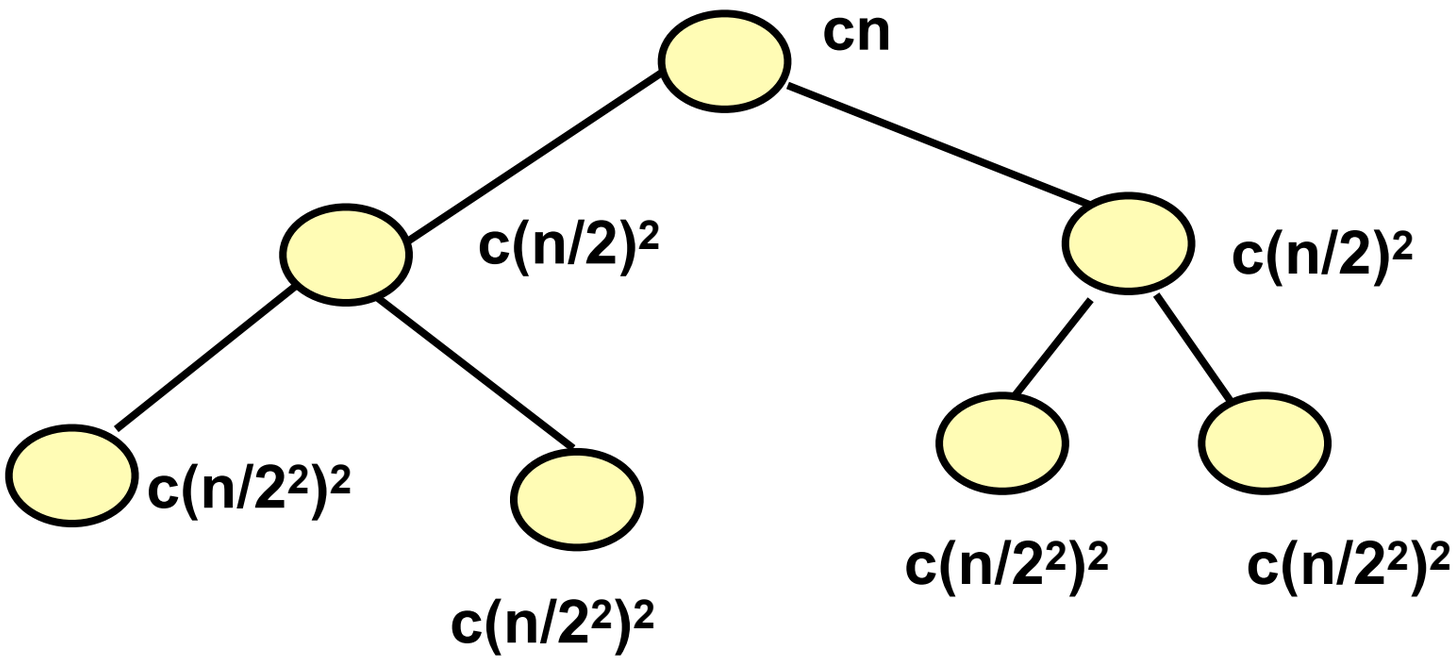
tempo di esecuzione Esercizio1 in $\Theta(n^2)$
tempo di esecuzione Esercizio2:
$T(n) = 2T(n/4) + \Theta(n^2)$
relazione da risolvere
$T(n) = d$ se $n \leq 1$
$T(n) = 2T(n/4) + cn^2$ altrimenti

Assumiamo che i + x con $0<x<1$ sia i e anche j-x = j

$T(n) = 2T(n/4) + cn^2$







In un generico livello i allora i nodi saranno etichettati con $c(n/2^i)^2 = cn^2/2^{2i}$ il penultimo livello conterrà nodi etichettati $cn^2/2^{2(h-1)}$ mentre l'ultimo livello nodi etichettati d. Ponendo per semplicità $n=2^h$:

possiamo dire che $T(n) = 2^h d + \sum_{i=0,\ldots,h-1} cn^2/2^{2i} = nd + cn^2 \sum_{i=0,\ldots,h-1} 1/2^{2i} \leq$

$nd + cn^2 \sum_{i=0,\ldots,\infty} 1/2^{2i} = O(n^2)$, visto che la serie è convergente a una costante.

$T(n) = d$ se $n \leq 1$
$T(n) = 2T(n/4) + cn^2$ altrimenti

Dobbiamo dimostrare per induzione che esistono due costanti positive k e n_0 tali che $T(n) \leq kn^2$ per ogni $n \geq n_0$.

Supponiamo la tesi vera per tutti i valori $m < n$, quindi $T(n/4) \leq k(n/4)^2$.
Considerando $T(n)$ si ha:

$T(n) = 2T(n/4) + cn^2 \leq$
$k(n/4)^2 + cn^2 =$
$kn^2/16 + cn^2$

Cerchiamo ora i valori di k e n che rendano vera la disuguaglianza:
$kn^2/16 + cn^2 \leq kn^2$ per ogni $n \geq n_0$.

$kn^2/16 + cn^2 \leq kn^2 \Leftrightarrow$

$kn^2 + 16cn^2 \leq 16kn^2 \Leftrightarrow$

$16cn^2 \leq 15kn^2 \Leftrightarrow$

$16/15c \leq k$ e $n \geq 1$ Quindi $T(n) \leq 2cn^2$ per ogni $n \geq n_0$, prendendo $k=2c$.
Ma per il caso base dovrebbe essere $T(1) \leq 2c$, mentre $T(1) = d$ e quindi prendiamo $k= 2c+d$ e possiamo concludere che $T(n) = O(n^2)$.
Poiché il termine additivo fornisce un limite inferiore in $\Omega(n^2)$, concludiamo che $T(n) = \Theta(n^2)$.

$T(n) = 2T(n/4) + n^2$

Prendiamo $n = 4^m = 2^{2m}$ e supponiamo che $T(n) = c$ per $n \leq 1$, per un certo $c > 0$.

Per iterazione:

$T(n) = 2T(n/4) + n^2 = 2[2T(n/4^2) + n^2/4^2] + n^2 = 2^2T(n/4^2) + 2n^2/4^2 + n^2 =$

$2^2[2T(n/4^3) + n^2/4^4] + 2n^2/4^2 + n^2 = 2^3T(n/4^3) + 2^2n^2/4^4 + 2n^2/4^2 + n^2$

$= 2^mT(1) + \sum_{i=0,\ldots,m-1} 2^in^2/4^{2i}$

$= 2^mc + \sum_{i=0,\ldots,m-1} n^2/2^{3i}$

$= 2^mc + n^2\sum_{i=0,\ldots,m-1} 1/2^{3i} =$

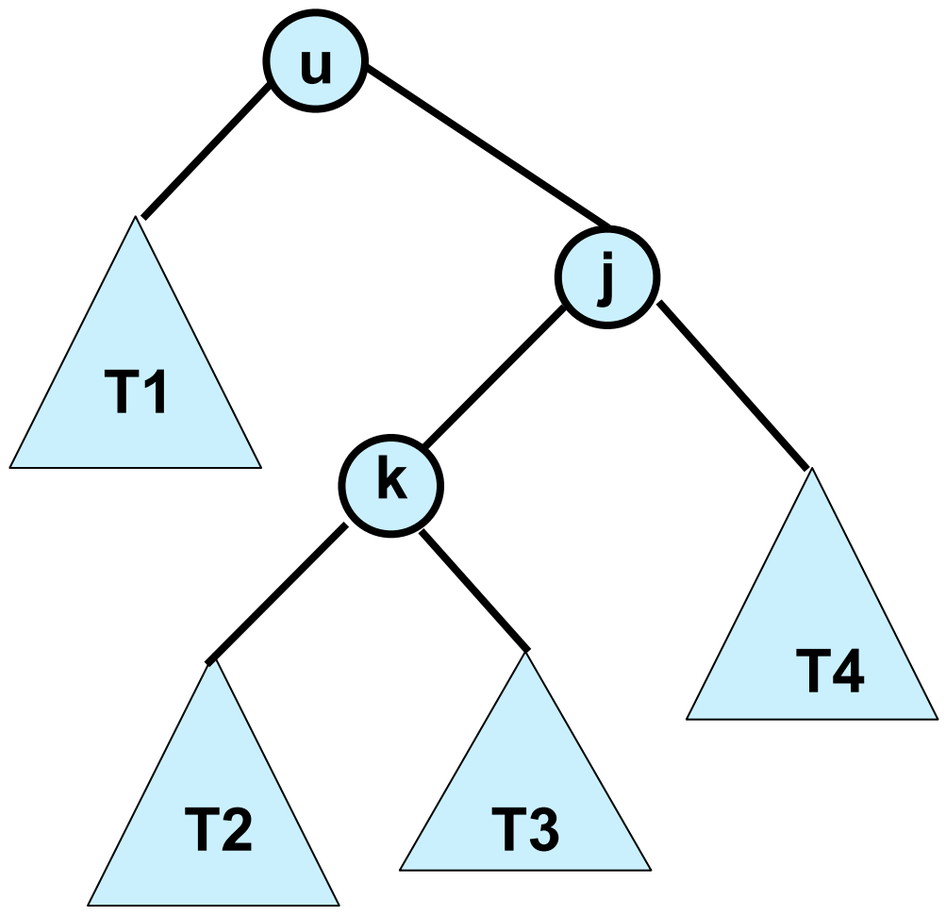
$2^mc + n^2\sum_{i=0,\ldots,m-1} 1/2^{3i} \leq$

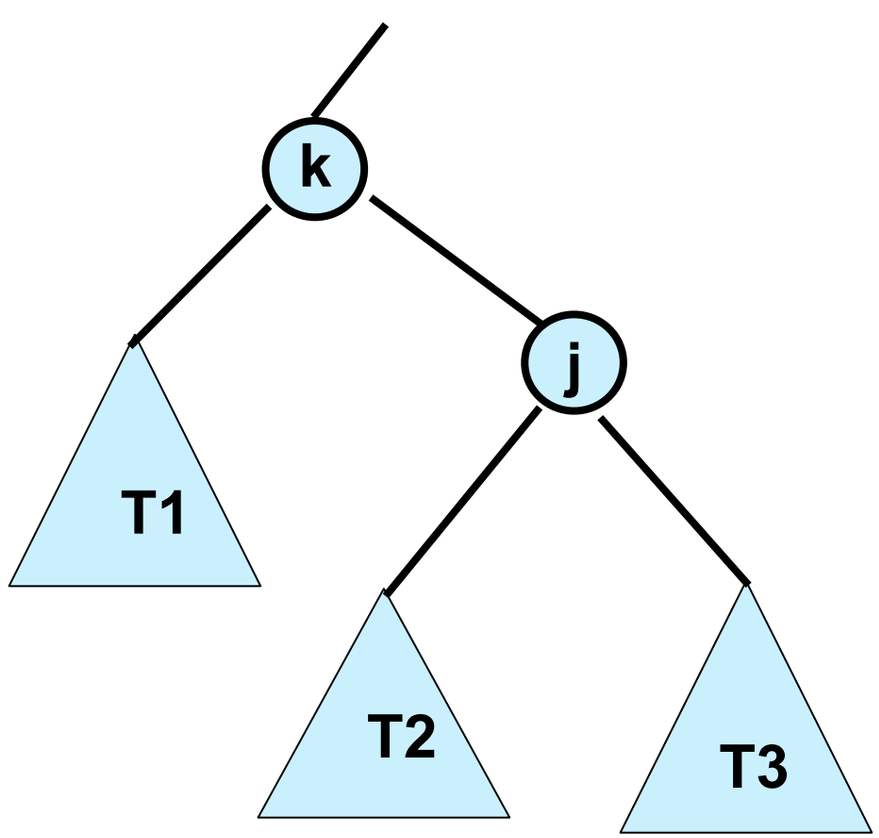
$nd + cn^2 \sum_{i=0,\ldots,\infty} 1/2^{3i} = O(n^2)$, visto che la serie è convergente a una costante.

ABR

Preso un ABR T (con chiavi tutte distinte) e una chiave k, lo split di T intorno a k è una partizione di T in due ABR T1 e T2 il primo contenente gli elementi più piccoli di k e l'altro i più grandi.
Come realizzeresti uno split e quanto costa in termini di tempo la soluzione proposta?

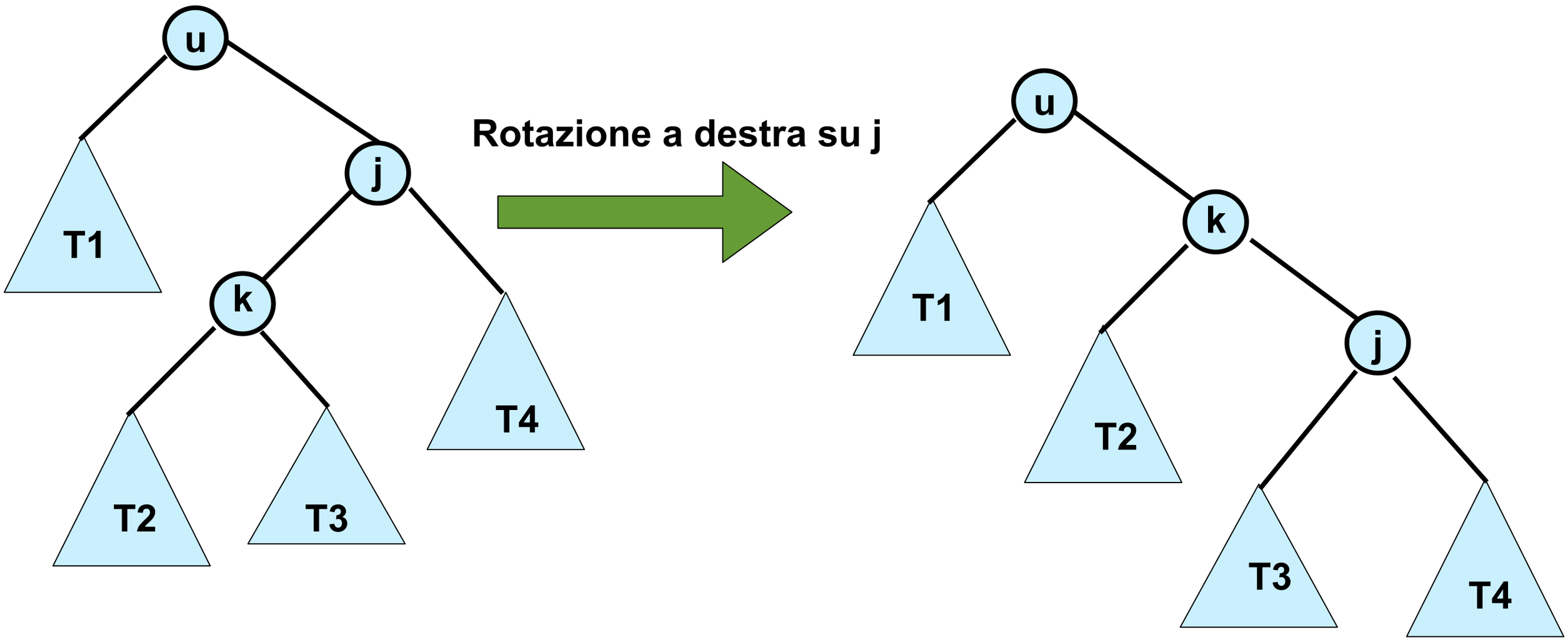
Innanzi tutto troviamo la chiave k in T.
Poi si risale verso la radice facendo delle rotazioni per portare k alla radice. A questo punto il sottoalbero sinistro e quello destro costituiscono lo split di T intorno a k.
Ogni rotazione diminuisce di uno la distanza di k dalla radice e costa O(1), quindi al più il costo è O(h)





ABR

Innanzi tutto troviamo la chiave k in T.
Poi si risale verso la radice facendo delle rotazioni per portare k alla radice. A questo punto il sottoalbero sinistro e quello destro costituiscono lo split di T intorno a k.
Ogni rotazione diminuisce di uno la distanza di k dalla radice e costa O(1), quindi al più il costo è O(h)



ABR

Innanzi tutto troviamo la chiave k in T.
Poi si risale verso la radice facendo delle rotazioni per portare k alla radice. A questo punto il sottoalbero sinistro e quello destro costituiscono lo split di T intorno a k.
Ogni rotazione diminuisce di uno la distanza di k dalla radice e costa O(1), quindi al più il costo è O(h)