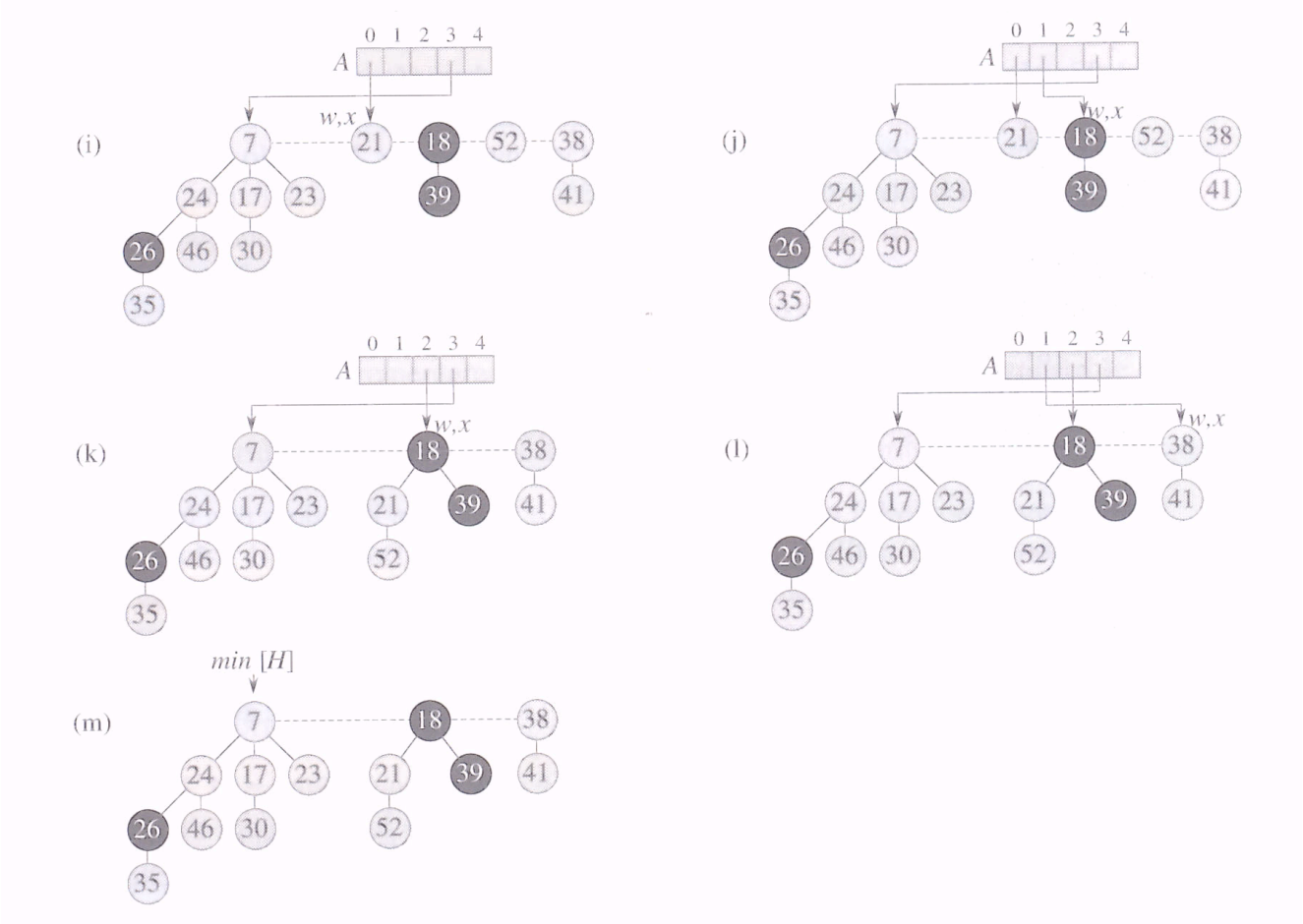
Estrazione del minimo dall'Heap di Fibonacci (4)



Unione di Heaps

Heap: ci sono due strategie: o costruire l'Heap a partire dal vettore disordinato di n+m elementi O(n+m), oppure aggiungere uno alla volta gli elementi dell'Heap di dimensione minore a quello di dimensione maggiore O(m log (n+m))

Heap di Fibonacci: si uniscono le liste delle radici dei due Heaps (si aggiorna il nuovo minimo e il numero totale dei nodi)

$$\chi_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = \Theta(1) + (t(F_i) + 2m(F_i)) - (t(F_{i-1}) + 2m(F_{i-1})) =$$
$$\Theta(1) + ((t(F^1_{i-1}) + t(F^2_{i-1})) + 2(m(F^1_{i-1})+m(F^2_{i-1}))) +$$
$$- ((t(F^1_{i-1}) + 2m(F^1_{i-1})) + (t(F^2_{i-1}) + 2m(F^2_{i-1})) = \Theta(1)$$

Decremento di un valore

Heap: si decrementa il nodo fissato x. Viene poi riaggiustato l'Heap scambiando il nuovo nodo con il padre se il valore ad esso associato è maggiore di x. Il procedimento può iterare fino a raggiungere la radice. O(logn).

Heap di Fibonacci: si decrementa il nodo fissato x. Se il valore di x è minore di quello del padre, si porta x alla radice. Se p(x) è marcato si porta anch'esso alla radice, ma smarcato. Il procedimento può iterare finchè un padre non è marcato o finchè non si arriva al livello delle radici. Sia (cost-1) il numero di volte che un padre viene portato alla radice e smarcato.

$c_i = (cost-1)\ O(1) + O(1) = O(cost)$

$\chi_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = O(cost) + (t(F_i) + 2m(F_i)) - (t(F_{i-1}) + 2m(F_{i-1})) =$

$O(cost) + ((t(F_{i-1})+cost)+2(m(F_{i-1}) - (cost-1) + 1)) - (t(F_{i-1}) + 2m(F_{i-1}))=$

$O(cost) + 4 - cost = O(1)$

Eliminazione di un nodo

Heap: scambia il contenuto del nodo x da cancellare con la foglia più a destra nell'ultimo livello. Elimina la foglia e riduci di uno la dimensione dell'Heap. Riaggiusta l'Heap scambiando il valore del nodo x con quello del padre se il valore ad esso associato è maggiore di x. Il procedimento può iterare fino a raggiungere la radice. O(logn).

Heap di Fibonacci: decrementa a 0 il nodo x. Applica prima la procedura decrementa e poi la cancella minimo

$\chi i = O(1) + O(d(n)) = O(d(n))$

Limitazione del grado massimo

TH: il grado massimo d(n) di un qualunque nodo in un Heap di Fibonacci con n nodi è O(logn)

Prova : sia k=d(x), con x nodo dell'HF di dimensione n.

Per il LemmaI. vale $n \geq size(x) \geq F_{k+2}$

Per il LemmaII. $n \geq \phi^k$ da cui

$k \leq \log_\phi n \leq \log_2 n$ $(\log_\phi n = \log_2 n / \log_2 \phi)$

LemmaI. size (x) $\geq F_{k+2}$ $(F_0 = 0; F_1 = 1; F_k = F_{k-1} + F_{k-2})$ (x è in size(x))

LemmaII. $F_{k+2} = 1 + \Sigma F_i \geq \phi^k$, con i = 0,…,k (con $\phi = (1+\sqrt{5})/2$)

LemmaIII. Siano $y_1, y_2, \ldots y_k$ i figli di un nodo x dati nell'ordine con cui sono stati collegati ad x (dal meno al più recente), si ha che: $d(y_1) \geq 0$, $d(y_i) \geq i-2$

Lemma 1.

Lemma1. size (x) $\geq F_{k+2}$

Prova per induzione :

- sia s_k la più piccola size(z) fra tutti i nodi z con d(z) = k;
  $s_0 = 1$; $s_1 = 2$; $s_2 = 3$.
- Come da LemmaIII, siano $y_1, y_2, \ldots y_k$ i figli di x, si ha:
  size (x)$\geq s_k \geq 2 + \Sigma\, s_{i-2}$ per i =2,…k

Passo base e hp. induttiva: $s_0 \geq F_2$ (1=1); $s_1 \geq F_3$ (2=2); fino a $s_{k-1} \geq F_{k+1}$

Prova: $S_k \geq 2 + \Sigma\, s_{i-2} \geq 2 + \Sigma\, F_i$ per i =2,…k

$= 1 + \Sigma\, F_i$ per i =0,…k $= F_{k+2}$

LemmaIII. Siano $y_1, y_2, \ldots y_k$ i figli di un nodo x dati nell'ordine con cui sono stati collegati ad x (dal meno al più recente), si ha che: $d(y_1) \geq 0$, $d(y_i) \geq i-2$

Lemma 11.

Lemma11. $F_{k+2} = 1 + \Sigma F_i$, per i = 0,…,k, per tutti gli interi $k \geq 0$

Dimostrazione per induzione:

Passo base: $1 + F_0 = 1 + 0 = 1 = F_2$

Ipotesi induttiva: $F_{k+1} = 1 + \Sigma F_i$, per i = 0,…,k-1

Prova: $F_{k+2} = F_k + F_{k+1} = F_k + (1 + \Sigma F_i$, per i = 0,…,k-1) =

$= 1 + \Sigma F_i$, per i = 0,…,k

Dalla teoria dei numeri di Fibonacci:

$F_{k+2} \geq \phi^k$ (con $\phi = (1+\sqrt{5})/2$, ovvero (a+b):a = a:b)

Lemma III.

LemmaIII. Siano $y_1, y_2, \ldots y_k$ i figli di un nodo x dati nell'ordine con cui sono stati collegati ad x (dal meno recente al più recente), si ha che: $d(y_1) \geq 0$, $d(y_i) \geq i-2$

Prova: banalmente $d(y_1) \geq 0$

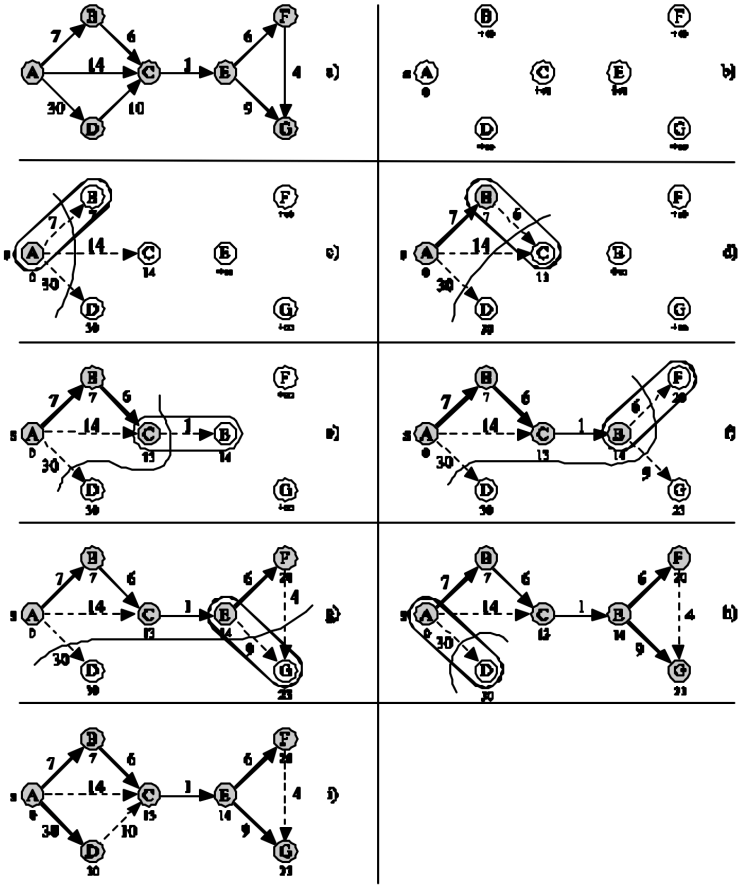
Quando y_i è stato connesso ad x, questi già aveva $y_1, y_2, \ldots y_{i-1}$ come figli. Quindi $d(x) = d(y_i) \geq i-1$.

Ma il nodo y_i per rimanere figlio di x può al più aver perso un figlio, quindi $d(y_i) \geq i-2$

Confronto dei costi

	HEAP	HEAP DI FIBONACCI
Creazione di un heap	$\Theta(1)$	$\Theta(1)$
Inserisci un nuovo elemento	$O(\log n)$	$\Theta(1)$
Trova il minimo	$\Theta(1)$	$\Theta(1)$
Estrai il minimo	$O(\log n)$	$O(d(n))=O(\log n)$
Unisci due Heaps	$O(n)$	$\Theta(1)$
Decrementa una chiave	$O(\log n)$	$O(1)$
Cancella un nodo	$O(\log n)$	$O(d(n))=O(\log n)$

Algoritmo di Dijkstra



Costo dell'algoritmo di Dijkstra

Worst case $W(n,m)$ =

O(n inserzioni + n eliminaminimo + m decrementa chiave)

Lista	$O(n\ 1 + n\ n + m\ 1) = O(n^2)$
Heap	$O(n \log n + n \log n + m \log n) = O(m \log n)$
Heap Fib	$O(n\ 1 + n \log n + m\ 1) = O(n \log n + m)$

E' possibile dimostrare che $O(n \log n + m)$ è la migliore complessità asintotica raggiungibile