Binomial Queue Algorithms

Υλοποίηση και ανάλυση

Παναγιώτης Διαμαντάκης

ΑΛΜΑ

Νοέμβριος 2021

Περιεχόμενα

Περιεχόμενα

Binomial Trees

Definition

Ορίζουμε την κλάση B_k ως εξής :

- τα δέντρα-κόμβοι είναι B_0 δέντρα
- τα Y, Z ξένα B_k δέντρα για $k \geq 1$. Αν προσθέσουμε μια ακμή ώστε η ρίζα του Y να γίνει το αριστερότερο παιδί της ρίζας του Z, παίρνουμε ένα B_k δέντρο.

Lemma

Έστω Z ένα B_k δέντρο. Τότε

- *Το Z έχει 2^k κόμβους.*
- *Το Z έχει $\binom{k}{l}$ κόμβους στο επίπεδο l[a].*

[a] $\binom{k-1}{l} + \binom{k-1}{l-1} = \binom{k}{l}$

Binomial Forest

Definition (Binomial Forest μεγέθους m)

- έχει m κόμβους, και
- αν το B_k δέντρο Y βρίσκεται αριστερά του B_l δέντρου Z, τότε $k > l$

Lemma

Έστω binomial forest μεγέθους m

- *Το μεγαλύτερο δέντρο του F είναι* $B_{\lfloor \lg m \rfloor}$
- *υπάρχουν* $v(m)$ *δέντρα στο F.* ($v(m) = \#$ *των 1 στην δυαδική αναπαράσταση του m*)
- *Υπάρχουν* $m - v(m)$ *ακμές στο F.*

Binomial Queue

Definition (Binomial queue μεγέθους m)

είναι ένα binomial forest μεγέθος m, αν κάθε binomial δέντρο που περιλαμβάνει είναι **heap-ordered** : δεν υπάρχει κόμβος-παιδί με τιμή μικρότερη από αυτήν του πατέρα του.

Lemma

Δύο heap-ordered B_k δέντρα μπορούν να συγχωνευθούν σε ένα heap-ordered B_{k+1} δέντρο σε σταθερό χρόνο.

Lemma

Έστω T ένα heap-ordered B_k δέντρο. Το δάσος που σχηματίζουν τα υποδέντρα που έχουν ως ρίζες τα παιδία της ρίζας του T, είναι ένα binomial queue μεγέθους $2^k - 1$.

Περιεχόμενα

Mergable priority queues

Definition

Μία priority queue, είναι mergable, αν πέρα από τις ενέργειες *εισαγωγής* και *διαγραφής*, υλοποιεί αποδοτικά και συγχώνευση μεταξύ ξένων ουρών προτεραιότητας.

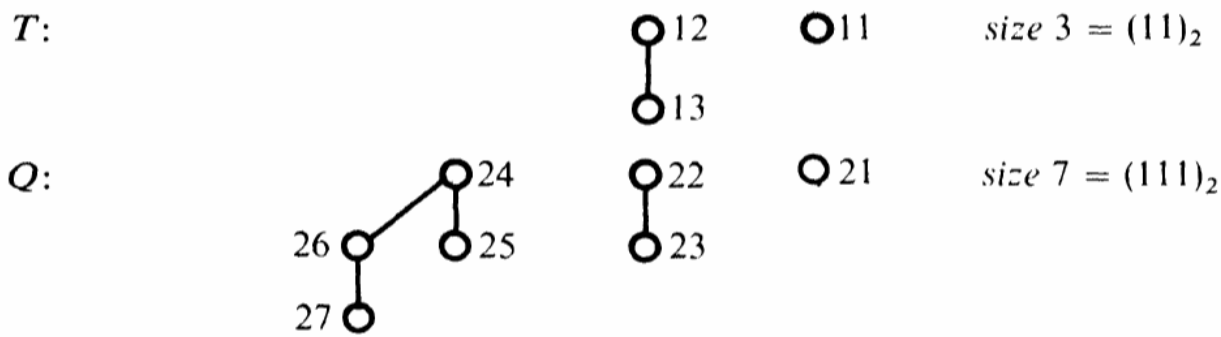
Union - Insert

Union(T,Q) : για την συνένωση δύο ουρών με μεγέθη $||T|| = t$ και $||Q|| = q$, η διαδικασία είναι ανάλογη με την πρόσθεση των t, q στο δυαδικό. Χρόνος εκτέλεσης $O\left(max(\log||T||, \log||Q||)\right)$

Insert(x, Q) : φτιάχνουμε δέντρο X που περιέχει μόνο το x, και κάνουμε Union(X, Q). Χρόνος εκτέλεσης $O(\log||Q||)$

Union

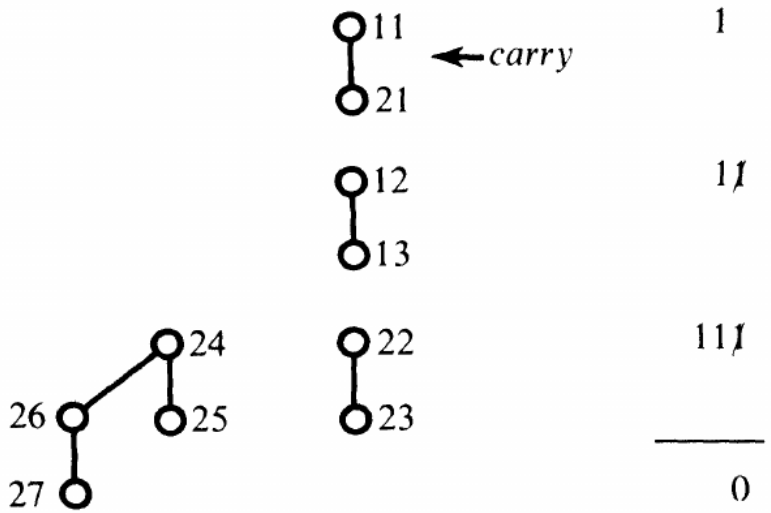
Union(T,Q) : για την συνένωση δύο ουρών με μεγέθη $||T|| = t$ και $||Q|| = q$, η διαδικασία είναι ανάλογη με την πρόσθεση των t, q στο δυαδικό. Χρόνος εκτέλεσης $O(max(\log||T||, \log||Q||))$



(a) *Binomial queues of size 3 and 7 to be merged for Union operation.*

Union

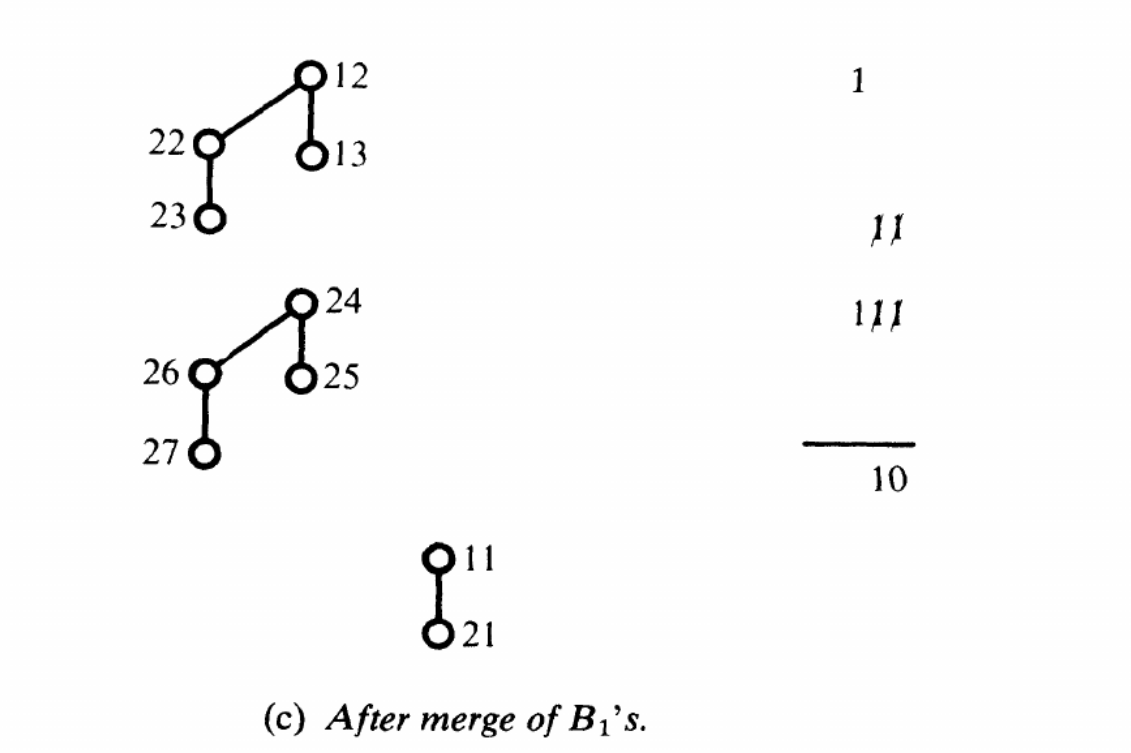
Union(T,Q) : για την συνένωση δύο ουρών με μεγέθη $||T|| = t$ και $||Q|| = q$, η διαδικασία είναι ανάλογη με την πρόσθεση των t, q στο δυαδικό. Χρόνος εκτέλεσης $O(max(\log||T||, \log||Q||))$



(b) *After merge of B_0's; result is carry.*

Union

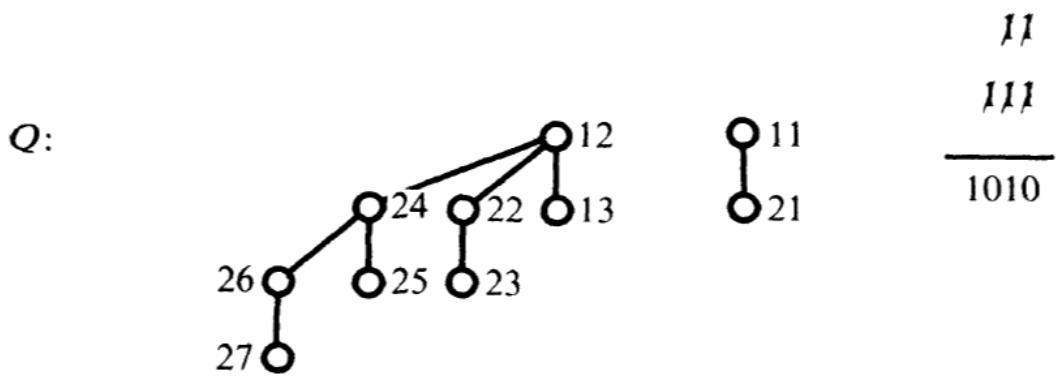
Union(T,Q) : για την συνένωση δύο ουρών με μεγέθη $||T|| = t$ και $||Q|| = q$, η διαδικασία είναι ανάλογη με την πρόσθεση των t, q στο δυαδικό. Χρόνος εκτέλεσης $O(max(\log||T||, \log||Q||))$



(c) *After merge of B_1's.*

Union

Union(T,Q) : για την συνένωση δύο ουρών με μεγέθη $||T|| = t$ και $||Q|| = q$, η διαδικασία είναι ανάλογη με την πρόσθεση των t, q στο δυαδικό. Χρόνος εκτέλεσης $O(max(\log||T||, \log||Q||))$



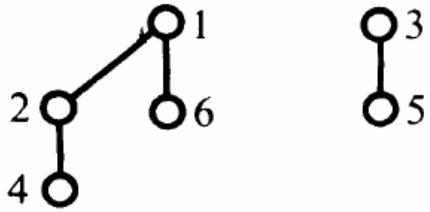
(d) *Merge completed.*

Delete Smallest

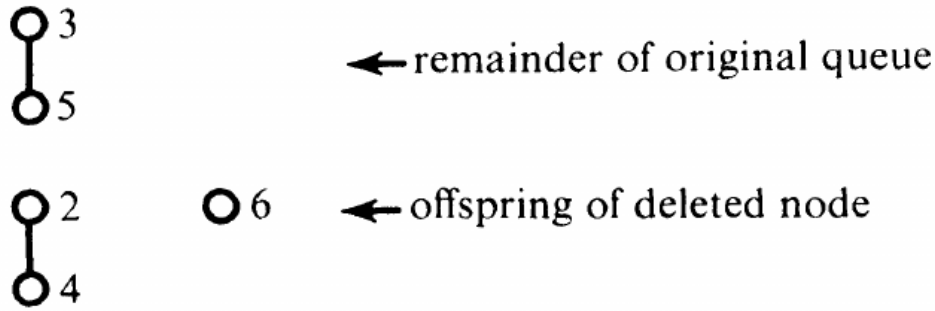
DeleteSmallest(Q) :

- Το μικρότερο στοιχείο x θα είναι ρίζα κάποιου B_k δέντρου της Q. Τα εξετάζουμε όλα διαδοχικά σε $O(\log ||Q||)$.
- Αφαιρούμε το T που περιέχει το x από την Q. Το είναι B_k, οπότε διαγράφοντας όλες τις ακμές της ρίζας x παίρνουμε μία binomial queue μεγέθος $2^k - 1$.
- Union(T', Q') σε χρόνο $O(\log ||Q||)$

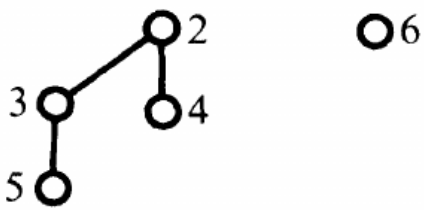
Delete Smallest - παράδειγμα



(a) *Binomial queue of size* 6. *Node* 1 *is to be deleted.*



(b) *Two queues which result from removing node* 1.



(c) *Resulting queue of size* 5 *after merging.*

Διαγραφή τυχαίου στοιχείου

Lemma

Ο παραπάνω αλγόριθμος γενικεύεται και για διαγραφή τυχαίου στοιχείου.

Αρκεί να γενικεύσουμε το δεύτερο στάδιο της DeleteSmallest.
Αν το ζητούμενο στοιχείο βρίσκεται σε ένα B_k δέντρο T, χωρίζουμε το T σε δύο B_{k-1} δέντρα Y, Z. Συνεχίζουμε αναδρομικά στο δέντρο που περιέχει το x και αποθηκεύουμε το άλλο, μέχρι το x να βρεθεί στην ρίζα κάποιου υποδέντρου.
Όταν η διαδικασία ολοκληρωθεί, έχουμε αποθηκεύσει k υποδέντρα, που σχηματίζουμε μία (πλήρη) binomial queue μεγέθους $2^k - 1$

Amortized complexity on Insertion

Lemma

Η πολυπλοκότητα της εισαγωγής μπορεί να βελτιωθεί σημαντικά, αν εξετάσουμε το κόστος σε μια σειρά από k διαδοχικές εισαγωγές.

Έστω οι εισαγωγές:

$$Insert(x_1, Q); Insert(x_2, Q); \ldots Insert(x_k, Q)$$

κάθε εισαγωή έχει κόστος
$O(1) + O(\#$ ακμών που προστέθηκαν κατά την εισαγωγή$)$
Αν αρχικά $||Q|| = m$, οι επιπλέον ακμές που προσθέτει η ακολουθία εντολών είναι $(m + k - v(m + k)) - (m - v(m)) = k + v(m) - v(m + k)$. [1]

Άρα ο χρόνος για k διαδοχικές εισαγωγές σε ουρά μεγέθους m είναι $O(k + \log m)$

[1] $v(n) = (\#$ των 1 στην δυαδική αναπ. του $n)$

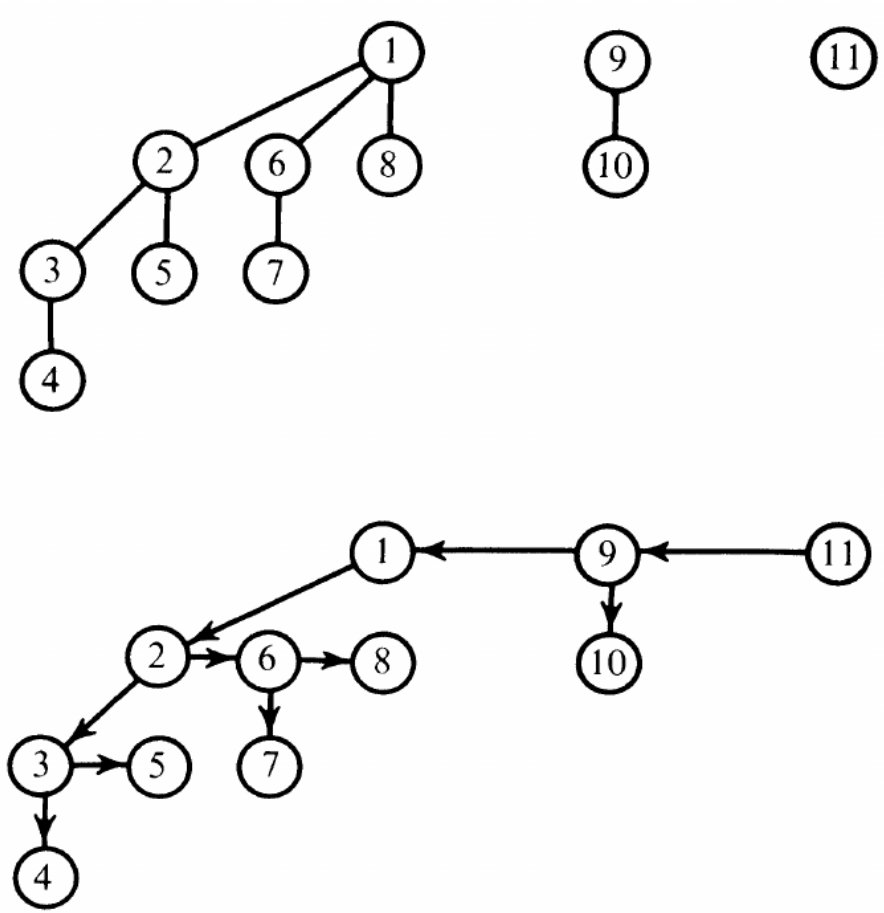
Περιεχόμενα

Structure V

Κάθε δάσος που αποτελεί στιγμιοτυπο μιας binomial queue μπορεί να απαρασταθεί ως εξής:

- τα επιμέρους δέντρα συνδέονται μεταξύ τους με ακμές από το μικρότερο στο μεγαλύτερο - επιτρέπει τη μεταφορά του "κρατούμενου" κατα την πράξη *Union*.
- καθένα από τα μπορεί να απαρασταθεί ως δυαδικό, ακολουθώντας την "left-child, right-sibling" αναπαράσταση

Structure V - παράδειγμα



(a) *A binomial queue and its representation using structure V.*

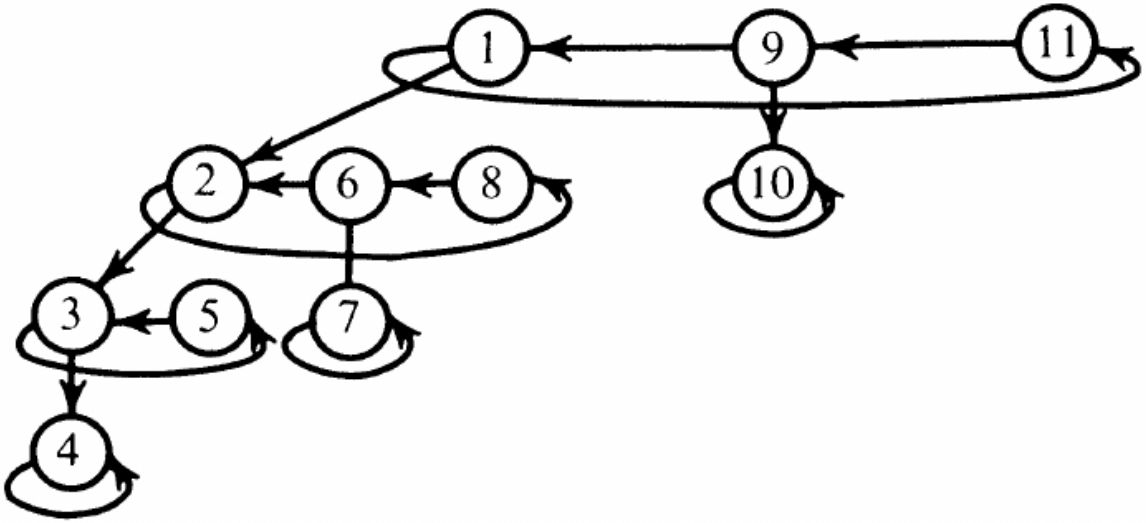
Παρατηρήσεις:

- Δύο heap-ordered binomial δεντρα ενώνονται σε σταθερό χρόνο
- Υποθέτουμε κάποιο queue header, με το μέγεθος της ουράς

Structure R

Στην προηγούμενη δομή οι ακμές του πρώτου επιπέδου έχουν αντίθετη φορά σε σχέση με τα άλλα επίπεδα $\Longrightarrow$ το δάσος που παίρνουμε αφαιρώντας ρίζα ενός binomial tree δεν αποτελεί binomial queue

Στην δομή R όλες οι οριζόντιες ακμές έχουν αριστερή φορά (ο αριστερότερος κόμβος έχει ακμή προς τον δεξιότερο)



(b) *A representation for the same queue using structure R.*

Structure R - Παρατηρήσεις

- Η δομή R πιο αργή από την V στις εισαγωγές, αλλά είναι πολύ καλύτερη επιλογή για διαγραφές.
- Καμία από τις δύο δομές δεν μπορεί να υπολογίσει αποδοτικά την διαγραφή τυχαίου στοιχείου. Μπορούν όμως να επεκταθούν και οι δύο με επιπλέον ακμές, από κάθε κόμβο προς τον γονέα του.

Περιεχόμενα

Analysis

Για την ανάλυση της ταχύτητας, περιοριζόμαστε στις πράξεις Insert και DeleteSmallest.
Η υλοποίηση βασίζεται στη δομή R και είναι γραμμένη σε assembly. Ο χρόνος εκτέλεσης έχει υπολογιστεί σε σχέση με τις αναγνώσεις και εγγραφές στη μνήμη (memory references) :

Insert	$16 + 19M + 2E + 6A$
DeleteSmallest	$38 + 11B + 6T + 4N - 2L + 4S + 14U + 2X$

Memory references

Insert	$16 + 19M + 2E + 6A$
DeleteSmallest	$38 + 11B + 6T + 4N - 2L + 4S + 14U + 2X$

M # merges που απαιτούνται για την εισαγωγή
E # αλλαγές στη ρίζα του δέντρου κατά τη διάρκεια των merge
A 1 αν $M = 0$, 0 αλλιώς

B 1 αν η ουρά δεν περιέχει B_0 δέντρο, 0 αλλιώς
T # binomial trees στην ουρά
N # φορές που άλλαξε η ελάχιστη τιμή καθώς ψάχνουμε την ελάχιστη ρίζα
L 1 αν το ελάχιστο στοιχείο βρίσκεται στο αριστερότερο υποδέντρο, 0 αλλιώς
S # παιδιών που έχει η ρίζα με την ελ. τιμή
U # merges που απαιτούνται για τη διαγραφή
X # αλλαγές στη ρίζα του δέντρου κατά τη διάρκεια των merge

Ανάλυση παραμέτρων εισαγωγής

Insert : $16 + 19M + 2E + 6A$

Έστω Q η binomial queue με μέγεθος $||Q|| = m$

- $A = 1$ αν δεν γίνει κανένα merge. Άρα $A = 1 \iff m$ άρτιος
- Ο αριθμός των merges είναι ευαίσθητος σε μικρές αλλαγές
  ($m = 2^n - 1 \implies M = n$ ενώ $m = 2^n \implies M = 0$)
  Γι αυτό παίρνουμε μέση τιμή στο διάστημα $[m/2, 2m]$ δίνει αναμενόμενη τιμή $1 + O((\log m)/m)$
  Η αντίστοιχη μέση τιμή για το A είναι $1/2 + O(1/m)$
- Ο συντελεστής Ε επηρεάζεται από την κατανομή των στοιχείων. Μπορούμε να υποθέσουμε ότι η Q είναι μία random binomial queue μεγέθους m, και ότι το στοιχείο που εισάγουμε επιλέγεται επίσης τυχαία.
  Τότε το E έχει διωνυμική κατανομή με μέση τιμή $M/2$ και διακύμανση $M/4$

Κόστος εισαγωγής

Για μεγάλες τιμές του $||Q|| = m$:

- η μέση τιμή του $A = 1/2 + O(1/m) \to 1/2$
- η μέση τιμή [2] του $M = 1 + O((\log m)/m) \to 1$
- η μέση τιμή του $E = M/2 \to 1/2$

άρα ο
μέσος χρόνος εκτέλεσης μιας εισαγωγής είναι $16+19\cdot 1+2\cdot\frac{1}{2}+6\cdot\frac{1}{2} = 39$

Για την χειρότερη περίπτωση, θα έχουμε $M = \log m$ merges, $A = 0$,
αφού $M \neq 0$ και $E = M = \log m$ άρα
$16 + 19 \cdot \log m + 2 \cdot \log m + 6 \cdot 0 = 21 \log m + 16$

[2] σε όλες τις n binary λέξεις μήκος m :
$\frac{1}{n}(\frac{n}{2}0 + \frac{n}{4}1 + \frac{n}{8}2 + \dots) = \frac{1}{2}\sum_{i=0}^{m} i/2^i = 1$

Κόστος Διαγραφής

Χειρότερη περίπτωση όταν $||Q|| = m = 2^n - 1$:

- B : η Q περιλαμβάνει B_0 αφού είναι πλήρης $\implies B = 0$
- T : ο αριθμός των δέντρων $\implies T = \log m + 1$
- N : οι φορές που αλλάζει η ελ. τιμή $(root_{B_0} > root_{B_1} > root_{B_2} > \dots) \implies N = \log m + 1$.
- L : λόγω του N θέλουμε η μικρότερη τιμή να είναι στην αριστερότερη ρίζα $\implies L = 1$
- S : η ρίζα με την μικρότερη τιμή θα είναι η αριστερότερη $\implies S = \log m$ παιδιά
- U : τόσα merges όσα και τα παιδία S
- X : όσα και τα merge

DeleteSmallest :
$38 + 11B + 6T + 4N - 2L + 4S + 14U + 2X = 46 + 30 \log m$

Σύγκριση με άλλες δομές

Average case running times when $\|Q\| = m$.

Queue	Insert (x, Q)	DeleteSmallest (Q)	Insert (x, Q); *DeleteSmallest* (Q)
binomial queue	39	$22 \lg m + 19$	$22 \lg m + 58$
leftist tree	$17 \lg m + 35$	$35 \lg m - 27$	$52 \lg m + 8$
linear list	19	$6m + 2 \lg m + 20$	$6m + 2 \lg m + 39$
heap	32	$18 \lg m + 1$	$18 \lg m + 33$
sorted list	$3m + 17$	15	$3m + 32$

Worst case running times when $\|Q\| = m$.

Queue	Insert (x, Q)	DeleteSmallest (Q)	Insert (x, Q); DeleteSmallest (Q)
binomial queue	$21 \lg m + 16$	$30 \lg m + 46$	$51 \lg m + 62$
leftist tree	$32 \lg m + 23$	$64 \lg m - 7$	$96 \lg m + 16$
linear list	19	$9m + 15$	$9m + 34$
heap	$12 \lg m + 14$	$18 \lg m + 16$	$30 \lg m + 30$
sorted list	$6m + 20$	15	$6m + 35$