GIT Department of Computer Engineering

CSE 222/505 – Spring 2020

Homework #07 Part 3 Report

Abdullah ÇELİK

171044002

Comprasions of Running Times

Regular Binary Search Tree								
Trial	Adding 10 random to 10k(ns)	10 successful deletion from 10k(ns)	Adding 10 random to 20k(ns)	10 successful deletion from 20k(ns)	Adding 10 random to 40k(ns)	10 successful deletion from 40k(ns)	Adding 10 random to 80k(ns)	10 successful deletion from 80k(ns)
1	9800	45630	3700	14000	15600	32500	11800	41900
2	4900	32300	3100	6300	6400	12300	8000	23500
3	4600	29300	6400	16400	5900	12500	4200	17100
4	3100	6100	28000	11800	3900	7000	4000	4900
5	6900	16600	12800	40900	6200	16900	5700	5200
6	3500	8500	6100	14300	6900	13800	4000	4500
7	5500	10700	6400	15900	4500	9500	7100	4500
8	5200	8200	6200	19200	7000	34000	3500	4000
9	9400	10300	3600	18600	4500	7500	4100	4000
10	2800	5300	3200	6900	7600	12400	4900	4000
Average	5570	17293	7950	16430	6850	15840	5730	11360

Red Black Tree(int the book)								
Trial	Adding 10 random to 10k(ns)	10 successful deletion from 10k(ns)	Adding 10 random to 20k(ns)	10 successful deletion from 20k(ns)	Adding 10 random to 40k(ns)	10 successful deletion from 40k(ns)	Adding 10 random to 80k(ns)	10 successful deletion from 80k(ns)
1	7200	28100	4400	8700	13500	13800	8000	13300
2	10000	29900	3800	11000	33700	15200	9100	28900
3	7400	29400	4500	8100	37200	21000	11900	30800
4	5800	11400	5000	7700	9400	9700	5200	6000
5	4600	8700	9300	11700	11100	16500	8700	24800
6	5800	9600	6100	21600	4500	7200	4700	28600
7	9000	12500	6600	11600	3900	7800	8100	6100
8	11400	19500	5300	6300	5200	8000	21900	4700
9	5000	9900	5400	14000	4300	10800	6000	5000
10	9000	12600	5100	10700	4500	7900	4700	4100
Average	7520	17160	5550	11140	12730	11790	8830	15230

Red Black Tree(int java - TreeMap)								
Trial	Adding 10 random to 10k(ns)	10 successful deletion from 10k(ns)	Adding 10 random to 20k(ns)	10 successful deletion from 20k(ns)	Adding 10 random to 40k(ns)	10 successful deletion from 40k(ns)	Adding 10 random to 80k(ns)	10 successful deletion from 80k(ns)
1	4400	17800	5000	12700	20500	8000	5600	7300
2	5300	11900	4600	10600	22000	9300	21900	29600
3	4400	10100	32600	12600	21300	8100	23000	23600

4	4700	10600	46900	38600	36700	12400	31600	16400
5	4700	9500	13100	33500	3900	7300	6700	10200
6	4500	9800	4000	11000	3800	7800	3600	4600
7	6500	30000	40200	29100	21100	8000	4500	6000
8	11100	24700	3100	11500	15100	10500	22200	3600
9	3600	8500	4500	37300	3400	6500	3700	3100
10	4100	10600	20900	24400	3500	6400	3600	4300
Average	5330	14350	17490	22130	15130	8430	12640	10870

B-Tree(in the book)								
Trial	Adding 10 random to 10k(ns)	10 successful deletion from 10k(ns)	Adding 10 random to 20k(ns)	10 successful deletion from 20k(ns)	Adding 10 random to 40k(ns)	10 successful deletion from 40k(ns)	Adding 10 random to 80k(ns)	10 successful deletion from 80k(ns)
1	25800		8700		9600		14900	
2	23300		8300		9800		9300	
3	24900		23800		8700		11200	
4	8900		23100		4000		12400	
5	8700		13700		6300		11900	
6	25400		7400		25800		34100	
7	11000		6700		8400		10600	
8	6400		5600		7300		9300	
9	6800		7500		13900		10000	
10	8900		12300		4500		30000	
Average	15010	0	11710	0	9830	0	15370	0

Skip List(in the book)								
Trial	Adding 10 random to 10k(ns)	10 successful deletion from 10k(ns)	Adding 10 random to 20k(ns)	10 successful deletion from 20k(ns)	Adding 10 random to 40k(ns)	10 successful deletion from 40k(ns)	Adding 10 random to 80k(ns)	10 successful deletion from 80k(ns)
1	6300	12600	24300	8900	14100	10500	7800	12800
2	5000	6200	40100	10400	33200	6200	34700	29800
3	5400	5800	5100	10600	47000	13300	11100	43800
4	5400	6600	6400	12500	8600	15100	9200	5300
5	8600	16300	10200	17300	22800	12400	7700	4600
6	23500	6400	5800	11100	18800	22200	10000	6100
7	5500	7000	21900	11000	7700	11700	22900	4700
8	9800	16300	22000	12800	5600	13800	15100	5800
9	4400	7200	22500	12600	5500	8200	10000	5900
10	4500	6100	4700	10200	7000	13700	8700	5700
Average	7840	9050	16300	11740	17030	12710	13720	12450

Skip List(in java - ConcurrentSkipListSet)								
Trial	Adding 10 random to 10k(ns)	10 successful deletion from 10k(ns)	Adding 10 random to 20k(ns)	10 successful deletion from 20k(ns)	Adding 10 random to 40k(ns)	10 successful deletion from 40k(ns)	Adding 10 random to 80k(ns)	10 successful deletion from 80k(ns)
1	26800	108800	18500	24800	10000	35200	11200	33000
2	11300	44500	5000	28000	4600	22700	4700	58100
3	8700	53100	5600	58300	4600	19700	5600	58000
4	7400	33600	4700	27000	4100	19600	23100	15700
5	3800	34800	10200	38300	15900	21000	24600	19900
6	7900	40800	5000	47000	23400	41100	7000	46000
7	23600	95500	4400	31900	6000	42100	25100	41400
8	23400	35500	4700	21100	8200	35500	6200	12500
9	6200	20100	14500	69200	5900	31200	5600	11400
10	10300	38500	11500	28500	4800	28300	23600	26600
Average	12940	50520	8410	37410	8750	29640	13670	32260

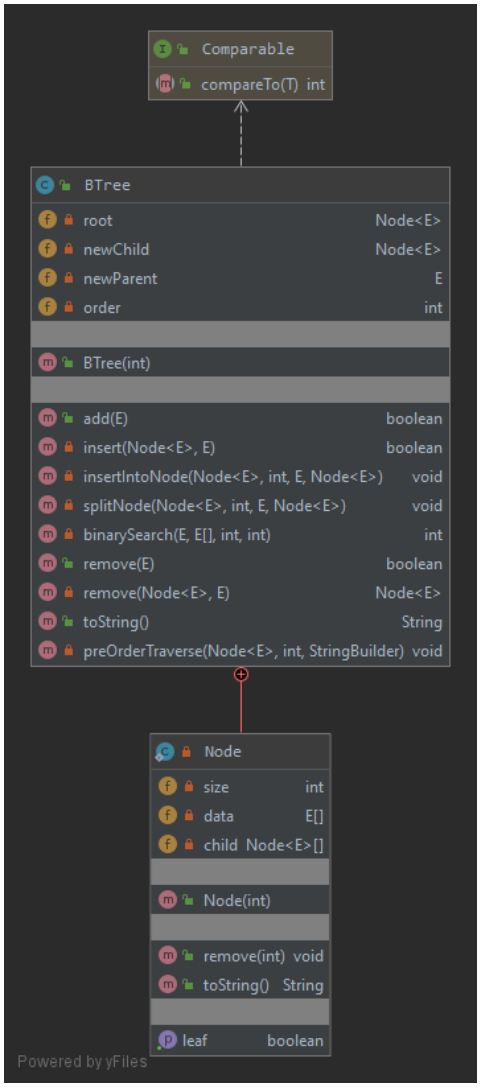
Skip List(in Q2)								
Trial	Adding 10 random to 10k(ns)	10 successful deletion from 10k(ns)	Adding 10 random to 20k(ns)	10 successful deletion from 20k(ns)	Adding 10 random to 40k(ns)	10 successful deletion from 40k(ns)	Adding 10 random to 80k(ns)	10 successful deletion from 80k(ns)
1	7700	25500	8600	15800	25600	17400	35900	32000
2	7000	11200	11300	19000	26000	16400	15600	53900
3	7200	8000	19700	21300	27800	21700	41100	41000
4	7600	11100	9200	28900	16100	30600	16200	28600
5	7500	13400	10800	17600	19900	11100	10800	12600
6	7500	12300	8000	18200	9000	15200	27700	14600
7	13500	26100	7800	18800	12000	21200	10200	12600
8	6600	13400	6800	16800	13100	17500	8800	7700
9	15600	45300	7300	22600	28100	16700	8900	13500
10	6400	13900	7000	14100	9900	24400	8100	11700
Average	8660	18020	9650	19310	18750	19220	18330	22820

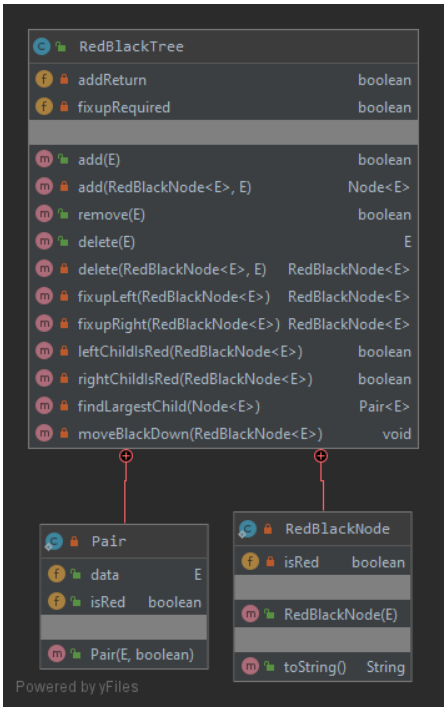
While adding process to the lists above, the time we expect should be approximately equal. The reason for this is that all add methods work in log (n) time. But there are sometimes exceptions to our results. The first reason for this is that we generate random numbers when adding. The second reason is that the computer is not always able to deliver the same performance. We expect total time to increase when the size of the lists increases. But there are exceptions above. The reason is that we reproduce random numbers. When some item is added to the best or near cases, sometimes item is added to the worst and the closest cases. This affects the total time. Since the total time is in nanoseconds, the number can grow very large in a minor glitch.

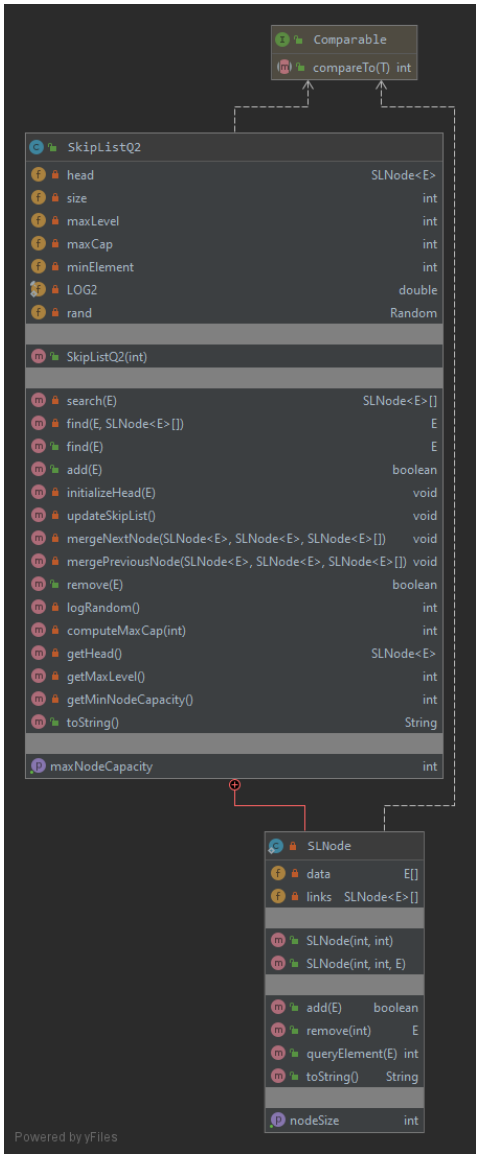
The time we wait while removing elements from the above lists should be approximately equal, as in adding. Because deletion also works at log (n) time. But there may be exceptions in the results. The

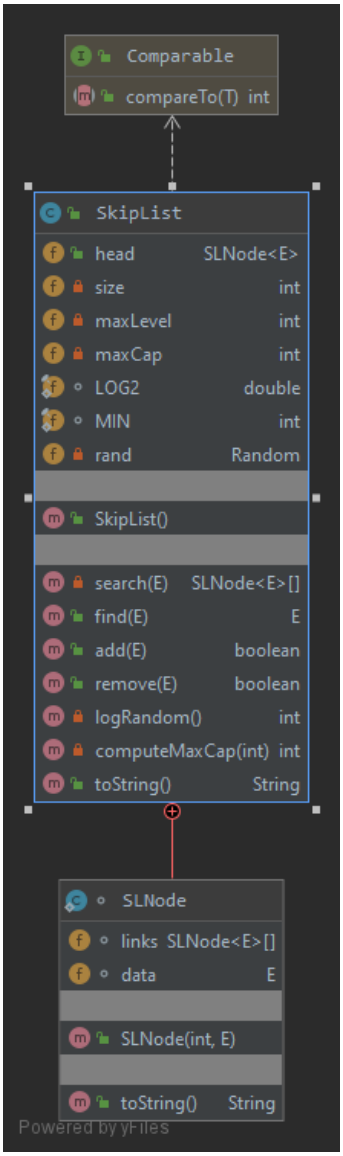
reasons are the reasons we mentioned above. The current processing capacity of the computer, such as removing is the best or worst case.

Class Diagrams






Comparable
compareTo(T) int
SkipListQ2
head SLNode<E>
size int
maxLevel int
maxCap int
minElement int
LOG2 double
rand Random
SkipListQ2(int)
search(E) SLNode<E>[]
find(E, SLNode<E>[]) E
find(E) E
add(E) boolean
initializeHead(E) void
updateSkipList() void
mergeNextNode(SLNode<E>, SLNode<E>, SLNode<E>[]) void
mergePreviousNode(SLNode<E>, SLNode<E>, SLNode<E>[]) void
remove(E) boolean
logRandom() int
computeMaxCap(int) int
getHead() SLNode<E>
getMaxLevel() int
getMinNodeCapacity() int
toString() String
maxNodeCapacity int
SLNode
data E[]
links SLNode<E>[]
SLNode(int, int)
SLNode(int, int, E)
add(E) boolean
remove(int) E
queryElement(E) int
toString() String
nodeSize int
Powered by yFiles


Comparable
compareTo(T) int
SkipList
head SLNode<E>
size int
maxLevel int
maxCap int
LOG2 double
MIN int
rand Random
SkipList()
search(E) SLNode<E>[]
find(E) E
add(E) boolean
remove(E) boolean
logRandom() int
computeMaxCap(int) int
toString() String
SLNode
links SLNode<E>[]
data E
SLNode(int, E)
toString() String
Powered by yFiles