Linköpings Tekniska Högskola
Institutionen för Datavetenskap
Tommy Färnqvist

28 oktober 2014

TDDI16 Datastrukturer och algoritmer
Tentamen (TEN1)
2014-10-28, 14–18 (TER1, TER2, TERE)

Examinator: Tommy Färnqvist
Jour: Tommy Färnqvist (telefon 013-282479).
Max poäng: 21 poäng (betyg 5 = 18p, 4 = 14p, 3 = 10p)
Hjälpmedel: INGA HJÄLPMEDEL TILLÅTNA!!!

VÄNLIGEN IAKTTAG FÖLJANDE

- Lösningar till olika problem skall placeras enkelsidigt på separata blad. Skriv inte två lösningar på samma papper.
- Sortera lösningarna innan de lämnas in.
- MOTIVERA DINA SVAR ORDENTLIGT: avsaknad av, eller otillräckliga, förklaringar resulterar i poängavdrag. Även felaktiga svar kan ge poäng om de är korrekt motiverade.
- Om ett problem medger flera olika lösningar, t.ex. algoritmer med olika tidskomplexitet, ger endast optimala lösningar maximalt antal poäng.
- SE TILL ATT DINA LÖSNINGAR/SVAR ÄR LÄSBARA.
- Lämna plats för kommentarer.

Lycka till!

1. För var och en av funktionerna nedan, ange den komplexitet (A–N) som bäst matchar dess exekveringstid. (5 p)

```
1. public static int f1(int N) {
     int x = 0;
     for (int i = 0; i < N; i++)
       x++;
     return x;
   }

2. public static int f2(int N, int R) {
      int x = 0;
      for (int i = 0; i < R; i++)
         x += f1(i);
      return x;
   }

3. public static int f3(int N, int R) {
      int x = 0;
      for (int i = 0; i < R; i++)
         for (int j = 0; j < N; j++)
            x += f1(j);
      return x;
   }

4. public static int f4(int N, int R) {
      int x = 0;
      for (int i = 0; i < N; i++)
         for (int j = 1; j <= R; j += j)
            x++;
      return x;
   }

5. public static int f5(int N, int R) {
      int x = 0;
      for (int i = 0; i < N; i++)
         for (int j = 1; j <= R; j += j)
            x += f1(j);
      return x;
   }
```

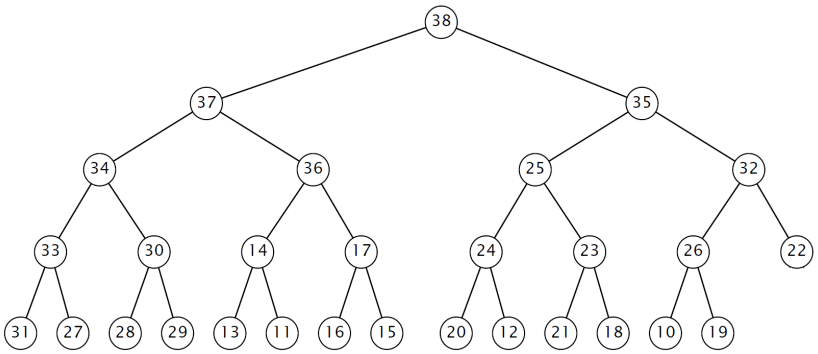
(A) R	(D) $N\log(R)$	(G) R^2	(J) $NR\log(R)$	(M) R^3
(B) N	(E) $R\log(N)$	(H) N^2	(K) NR^2	(N) N^3
(C) $N+R$	(F) NR	(I) $NR\log(N)$	(L) RN^2	

2. Binära sökträd. (3 p)

   (a) Visa resultatet av att sätta in nycklarna 4, 10, 3, 8, 5, 6, 25 i den ordningen i ett initialt tomt AVL-träd. (2)

   (b) I vilken ordning besöker en preordertraversering noderna i ditt resulterande AVL-träd ovan? (1)

3. Betrakta den binära heapen nedan. (2 p)



(a) Antag att den sista operation som utförts på den binära heapen ovan var att sätta in nyckeln x. Vilka av följande är möjliga värden på x? (1)

```
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
```

(b) Antag att du utför operationen `removeMax` på den binära heapen ovan. Vilka av följande nycklar blir då inblandade i jämförelser med andra nycklar? (1)

```
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
```

4. Vi använder en array med indexen 0 till 11 för att implementera en hashtabell där kollisioner hanteras med hjälp av kvadratisk sondering (*quadratic probing*). Hashfunktionen som används är $h(k) = k \bmod 12$. Visa en representation av tabellen efter att nycklarna 33, 10, 9, 13, 12, 45 satts in i den ordningen i den initialt tomma tabellen. (2 p)

5. Visa resultatet av de olika faserna när radix-sort med bas 10 används för att sortera följande värden. (2 p)

```
346 22 31 212 157 102 568 435 8 14 5
```

6. Givet en $N \times N$-matris av reella tal, ge en algoritm för att hitta det största talet som uppträder (minst) en gång på varje rad (eller avgör att inget sådant tal existerar). (3 p)

9	6	3	8	5
3	5	1	6	8
0	7	5	3	5
3	5	7	8	6
4	3	5	7	9

För full poäng krävs att exekveringtiden för din algoritm är proportionell mot $N^2 \log(N)$ i värsta fallet. Din algoritm får använda sig av $\mathcal{O}(N^2)$ extra minne.

7. En *läckande stack* är en generalisering av en stack som har metoder för att lägga till en sträng, ta bort den senast inlagda strängen och att ta bort en slumpvis utvald sträng, som i följande gränssnitt: (4 p)

	`LeakyStack()`	skapa en tom läckande stack
`void`	`push(string item)`	pusha strängen på den läckande stacken
`string`	`pop()`	ta bort och returnera den senast pushade strängen
`void`	`leak()`	ta bort en slumpvis utvald sträng från den läckande stacken

Beskriv, gärna med pseudokod, hur man kan implementera metoderna `push(String)`, `pop()` och `leak()` i tid proportionell mot $\log(N)$ i värsta fallet, där N är antalet strängar insatta i datastrukturen. Antag att du har tillgång till en slumptalsgenerator `Random.uniform(N)` som returnerar slumptal mellan 0 och $N-1$ med likformig sannolikhet.

Exempel:

```
LeakyStack stack;
stack.push("A");   // A           [ add A ]
stack.push("B");   // A B         [ add B ]
stack.push("C");   // A B C       [ add C ]
stack.push("D");   // A B C D     [ add D ]
stack.push("E");   // A B C D E   [ add E ]
stack.pop();       // A B C D     [ remove and return E ]
stack.push("F");   // A B C D F   [ add F ]
stack.leak();      // A B C F     [ choose D at random; delete D ]
stack.leak();      // A C F       [ choose B at random; delete B ]
stack.pop();       // A C         [ remove and return F ]
stack.pop();       // A           [ remove and return C ]
stack.pop();       //             [ remove and return A ]
```