Algoritmers effektivitet, tidskomplexitet

Tema: Tekniker för att uppskatta effektiviteten hos algoritmer.

U 1. Vad är tidskomplexitet? Hur räknar man ut tidskomplexitet?

U 2. I de fyra deluppgifterna i denna uppgift ska följande frågor besvaras:

- hur många gånger kommer satsen `SimpleStatement` att utföras?
- vad är tidskomplexiteten (ordo-notation)?

a)
```
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= 2; j++) {
        SimpleStatement;
    }
}
```

b)
```
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n/2; j++) {
        SimpleStatement;
    }
}
```

c)
```
for (int i = 1; i <= n; i++) {
    for (int j = n; j >= i; j--) {
        SimpleStatement;
    }
}
```

d)
```
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        if (i == j) {
            SimpleStatement;
        }
    }
}
```

U 3. En s.k. *dequeue* (double-ended queue) är en följd av element där man får ta bort första eller sista elementet och sätta in nya element längst fram eller längst bak.

a) Vad blir kostnaden för de fyra operationerna om en dequeue representeras av en referens till det första elementet i enkellänkad datastruktur?

b) Vad blir kostnaden för operationerna om man använder en enkellänkad struktur och låter en dequeue representeras av en referens till första elementet och en till det sista?

c) Finns det någon representation i vilken alla operationer får kostnaden O(1)?

U 4. Varför använder man tidskomplexitet istället för att helt enkelt mäta exekveringstiden? Hur förhåller sig den verkliga tidsåtgången till tidskomplexiteten?

U 5. Man har analyserat en algoritm med avseende på tidskomplexitet och kommit fram till att den är $O(n)$. Man har också implementerat algoritmen och mätt exekveringstiden. För n = 100 tog det 1 ms. Hur lång tid kommer det att ta för n = 1000 om körningen sker på samma dator?

U 6. Man har analyserat en algoritm med avseende på tidskomplexitet och kommit fram till att den är $O(2^n)$. Man har också implementerat algoritmen och mätt exekveringstiden. För n = 10 tog det 1 ms. Hur stort problem kan man lösa med algoritmen (på samma dator) om man har en sekund på sig?

U 7. I Javas klassbibliotek finns två klasser som implementerar interfacet `List`. Den ena klassen `ArrayList` använder ett fält (vektor) för att representera listan. Den andra `LinkedList` använder en dubbellänkad lista med referens till första och sista elementet.

a) Vad blir tidskomplexiteten för metoden `get(int index)` i respektive implementering?

b) Vad blir tidskomplexiteten för metoden `indexOf(Object x)` i respektive implementering?

c) Vad blir tidskomplexiteten för metoden `add(int index, E element)` i respektive implementering?

d) Vad blir tidskomplexiteten för metoden `remove(int index)` i respektive implementering?

e) Betrakta följande kod:

```
int sum = 0;
for (int i = 0; i < list.size(); i++) {
    sum = sum + list.get(i);
}
```

Vad är tidskomplexiteten om `list` är av typ `ArrayList<Integer>` respektive `LinkedList<Integer>`?

f) Finns det något sätt att skriva om koden i föregående deluppgift så att den blir lika effektiv oavsett vilken av de två typerna `list` har?

U 8. I början av en kurs tänker kursansvarig lärare göra ett upprop. Läraren har tillgång till en osorterad lista med anmälda studenter, n st. Det finns n+k studenter i lokalen vid start (flera har nämligen glömt anmäla sig). k är litet i förhållande till n.

Läraren väljer nu mellan två tillvägagångssätt:

1. Alla studenterna presenterar sig en efter en. För varje student letar läraren på sin lista. Om studenten finns på listan antecknas närvaro, annars skrivs studenten in sist på listan.
2. Läraren läser upp namnen på sin lista. Studenterna ombeds svara om de är närvarande. Närvaro registreras på listan. När hela listan lästs ombeds de (ungefär k) vars namn inte lästs upp presentera sig och de skrivs in på listan.

Vilken algoritm är effektivast? Vad är tidskomplexiteten för algoritmerna i ordo-notation?

U 9. En student har analyserat en sorteringsalgoritm och kommit fram till att dess tidskomplexitet är $O(n^2)$. Nu vill studenten kontrollera att detta är rätt genom att implementera algoritmen och mäta exekveringstiden för olika storlek på problemet. Beskriv hur man kan göra detta så att det verifieras att analysen är korrekt.