TDDI16 Lektion 1

Ordo-notation och tidskomplexitet

Magnus Nielsen & Christoffer Holm
magnus.nielsen@liu.se & christoffer.holm@liu.se

25 Augusti 2022

Regler

Några regler från Fö1 som kan vara användbara.

- (Ordo) $f \in \mathcal{O}(g)$ omm det existerar $c > 0, n_0 > 0$ sådana att $f(n) \leq c \cdot g(n)$ för alla $n \geq n_0$
  Intuition: Bortsett från konstanta faktorer växer f inte snabbare än g

- (Omega) $f \in \Omega(g)$ omm det existerar $c > 0, n_0 > 0$ sådana att $f(n) \geq c \cdot g(n)$ för alla $n \geq n_0$
  Intuition: Bortsett från konstanta faktorer växer f minst lika fort som g

- (Theta) $f(n) \in \Theta(g(n))$ omm $f(n) \in \mathcal{O}(g(n))$ och $g(n) \in \mathcal{O}(f(n))$
  Intuition: Bortsett från konstanta faktorer växer f och g lika snabbt.

  NOTERA: Ω är motsatsen till $\mathcal{O}$, dvs $f \in \Omega(g)$ omm $g \in \mathcal{O}(f)$.

Övningsuppgifter

1. Vilken eller vilka av följande förslag är ekvivalenta med $\mathcal{O}(n^3)$?
   (a) $\mathcal{O}(n^3 + n \log n)$
   (b) $\mathcal{O}(14n^3)$
   (c) $\mathcal{O}(n^3 + 3)$
   (d) $\mathcal{O}(n^3 - n^2)$
   (e) $\mathcal{O}((n^2)(n + 4))$
   (f) $\mathcal{O}(n(n^2 - 5))$

2. Vilken tidskomplexitet har följande funktion?

```
int n_fakultet(int const n) {
    int ans{1};
    for (int i = n; i > 0; i--) {
        ans = ans * i;
    }
    return ans;
}
```

3. Vilken tidskomplexitet har följande funktion?

```
int calc(int n) {
    int ans{};
    for (int i = 0; i < n*n; i++) {
        for (int j = 0; j < n-3; j++) {
            ans += i+j;
        }
    }
    return ans;
}
```

4. Visa att funktionen $f(n) = 2n^3 + 3n + 18 \in \Theta(n^3)$.
   Håll i åtanke reglerna från Fö1.

5. Vilken tidskomplexitet har följande funktion?

```
int f3(int x) {
    int ans = 0;
    for (int i = 0; i < x; i++) {
        ans += n_fakultet(i);
    }
    return ans;
}
```

6. Vilken tidskomplexitet har följande funktion?

```
int f3(int x) {
    int ans = 0;
    for (int i = 0; i < x; i++) {
        ans += n_fakultet(i % 4);
    }
    return ans;
}
```

7. Vilken tidskomplexitet har följande funktion? (**Tips:** Tänk på hur snabbt `y` minskar)

```
int exp(int x, int y) {
    if (y == 0) {
        return 1;
    } else if (y % 2 == 0) {
        int ans = exp(x, y / 2);
        return ans * ans;
    } else {
        return x * exp(x, y - 1);
    }
}
```

8. Vilken tidskomplexitet har följande funktion?

```
int f6(vector<int> &data) {
    int ans = 0;
    for (int i = 0; i < data.size(); i += 10) {
        int part = 0;
        for (int j = 0; j < 10; j++) {
            part += data[i + j];
        }
        ans += part * part;
    }
    return ans;
}
```

9. Antag att du har två program, A och B, som båda löser samma problem men med olika tidskomplexitet. Antag att A har tidskomplexitet $\Theta(n^2)$ och B har tidskomplexitet $\Theta(2^n)$.

   (a) Förklara varför det är rimligt att A kör på 8 sekunder medan B kör på 4 sekunder för en inmatning av storlek $n = 4$ trots att B har en mycket sämre tidskomplexitet.

   (b) Antag att körtiden för A och B är ungefär samma vid $n = 32$. Kan du dra några slutsatser om programmens körtid för $n > 32$ utifrån detta?

   (c) Utifrån dina svar på (a) och (b), redogör för när A har bättre körtid än B och vice versa.

   (d) Gör om (a) och (b) men under antagandet att A och B har *samma* tidskomplexitet. Kan vi då säga något om när A har bättre körtid än B och vice versa?

   Tips: Rita skisser av graferna för körtiden av A och B vid olika n.

10. Antag att du har skrivit ett program som är såpass komplicerat att det är svårt att avgöra tidskomplexiteten genom att bara resonera kring koden. Du inser ganska snabbt att man kan uppskatta vad tidskomplexiteten är genom att mäta körtiden för olika storlekar på indatan. Du mäter upp körtiden $T = 6$ sekunder för storleken $n = 3$. Vad förväntar du dig för ungefärlig körtid för storleken $n = 6$ om tidskomplexiteten är:

    (a) $\mathcal{O}(n^2)$?

    (b) $\mathcal{O}(n^3)$?

    (c) $\mathcal{O}(2^n)$?

    Exempel: Om tidskomplexiteten är $\mathcal{O}(n)$ så betyder det att körtiden ges av funktionen $T(n) = an + b$. Det är OK att anta att $b = 0$ eftersom att an är den dominerande termen. Detta ger att $T(n) = an$. För att bestämma vad a är så använder vi sambandet: $T(3) = 6$. Detta ger oss $T(3) = a \cdot 3 = 6$ vilket betyder att $a = 2$. Om vi då evaluerar $T(6)$ så får vi förväntad körtid 12 sekunder vid $n = 6$.

11. Antag nu istället att du har mätt upp olika mätvärden för ditt program. Vad tror du programmet har för tidskomplexitet givet följande mätvärden?

    (a)

	$n=1$	$n=2$	$n=3$	$n=4$
T(n)	$5s$	$8s$	$11s$	$14s$

    (b)

	$n=1$	$n=2$	$n=3$	$n=4$
T(n)	$1s$	$2.5s$	$5s$	$8.5s$

    (c)

	$n=1$	$n=2$	$n=4$	$n=8$
T(n)	$2s$	$5s$	$17s$	$65s$

    Notera: n ökar inte med 1 varje gång.

    (d)

	$n=1$	$n=2$	$n=4$	$n=8$
T(n)	$3s$	$7s$	$11s$	$15s$

    Notera: n ökar inte med 1 varje gång.

    Tips: Rita graferna och jämför med kända tidskomplexiteter.