Ugeseddel: Opvarmning

Philip Bille

Om denne uge

Litteratur Kursusguide.

Opgaver

1 Løkker Målet med denne opgave er at du kan checke at du forstår basale programmeringskonstruktioner. Du skal *ikke* kode funktionerne. Hvad returner Java/C/C++ funktionerne `loop1`, `loop2`, `loop3` og `loop4` i figur 1 når

1.1 `n` = 4?

1.2 `n` = 10?

1.3 `n` = 1000?

1.4 udtrykt som en funktion af `n`?

2 Rekursion og iteration En funktion er *rekursiv* hvis den kalder sig. F. eks. er Java/C/C++ funktionen `f(A,n)` i figur 1 rekursiv (for at få gyldigt C/C++ kode ændres `[]` til `*`). Løs følgende opgaver.

2.1 Hvad beregner `f(A,n)` hvis `A` er en tabel af heltal af længde `n`? Du skal *ikke* kode den.

2.2 Omskriv `f(A,n)` til at være *iterativ* i stedet, dvs., lav en funktion der gør det samme som `f(A,n)` uden at kalde sig selv.

3 Introduktion til CodeJudge og Piazza Løs følgende opgaver.

3.1 Log på CodeJudge og læs tutorial.

3.2 [$o\dagger$] Implementer en funktion til at lægge to tal sammen og afprøv den i CodeJudge.

3.3 Log på Piazza.

4 Lineartrose Du har netop ansat 128 programmører til din nyopstartede high-tech virksomhed. Desværre lider en af dem af den frygtede *lineartrose* sygdom, der får alle i nærheden af vedkommende til at skrive langsomme programmer. For at finde den syge programmør har du lejet et særligt kammer, du kan bruge til at afprøve om en delmængde af dine programmører har en syg iblandt sig. Det er dyrt at leje lokalet og processen med at teste gruppen er meget omstændig (lange og komplicerede programmeringstest er nødvendige). Derfor vil du gerne minimere antallet af gange du skal bruge lokalet til at finde den syge. Løs følgende opgaver.

4.1 Vis at du kan finde syge programmør med højst 7 tests.

4.2 Hvor mange test kan du klare dig med hvis du har n programmører?

4.3 [$*$] Antag du lejer $k > 1$ kamre som du kan bruge til at teste k grupper af programmører samtidig. Hvor mange *runder* af tests kan du klare dig med til at finde den syge programmør? I hver runde kan du teste k grupper parallelt.

```
int loop1(int n) {
   int x = 0;
   for(int i = 0; i < n; i++) {
      for(int j = 0; j < n; j++) x++;
   }
   return x;
}

int loop2(int n) {
   int x = 0;
   for(int i = 0; i < n; i++) x++;
   for(int j = 0; j < n; j++) x++;
   return x;
}

int loop3(int n) {
   int x = 0;
   for(int i = 0; i < n; i++) {
      if (i == n-1) for(int j = 0; j < n; j++) x++;
   }
   return x;
}

int loop4(int n) {
   int x = 0;
   for(int i = 0; i < n; i++) {
      for(int j = i; j < n; j++) x++;
   }
   return x;
}

int f(int[] A, int n) {
   if(n == 0) return 0;
   else return f(A, n - 1) + A[n-1];
}
```

Figur 1: Løkker og rekursion

5 Zombieduellering Du har en hær af n hjernetomme zombier. Du vil gerne finde den stærkeste og svageste zombie blandt gruppen. Ved at sætte to zombier sammen i et bur med en luns kød kan du hurtigt afgøre hvilken af de to er stærkest. Desværre slider det på zombier at slås, så du vil gerne minimere antallet af dueller. Løs følgende opgaver.

5.1 Vis hvordan du finder den stærkeste zombie med højst $n-1$ dueller.

5.2 [$*$] Vis hvordan du finder både den stærkeste og svageste zombie med højst $3n/2$ dueller.

5.3 [$**$] Vis hvordan du finder både den stærkeste og næststærkest zombie med højst $n + \log_2 n$ dueller.