Diego Alberto Flores Leon
961206-6697
dleon@kth.se
DD2350
2018-10-07

Mästarprov 1

Täckning av en kvadrat med L

Algoritmen som används tar emot en m x m matris, lägger till ett L i mitten och delar upp matrisen i fyra delar. Med hjälp av rekursion fortsätter man lägga till L och dela upp kvadrater tills man når basfallet. Då ett L bara får ta tre platser så blir basfallet en 2x2 matris. För varje kvadrat man besöker går det bara att fylla alla element utom 1. L:en kommer kunna placeras på 4 olika sätt och bestäms med hjälp av en rotations variabel. Värdet för variabeln och den "tomma" platsen beror på hur L:et från det tidigare rekursion anropet lagts till. Slutligen kombineras resultaten och matrisen returneras.

Pseudokod

```
//Numret på det första L:et som läggs till
c ← 1;

void main(args)
      m ← 2^n

      //En 2^n x 2^n matris skapas
      matrix ← new int[m][m]

      /*Funktionen algo anropas och returnerar en nästan
      heltäckt matris*/
      matrix ← algo(matrix, m, m, 0, 0, 0)

static algo(matris, m, size, x, y, rotation)
      if rotation = 0 then
            //Skriver ut ett L
      else if rotation = 1 then
            //Skriver ut ett uppåtriktat L
      else if rotation = 2 then
            //Skriver ut ett nedåtriktat L
      else if rotation = 3 then
            //Skriver ut ett upp och ner vänt L

      //Efter att L:et har lagts till ökas numret på L
      c ← c + 1

      //Basfallet. Returnera matris efter L har lagts till.
      Om vi har en 1x1 kvadrat går det inte att lägga till fler L*/
      if size = 2 then
```

```
            return matris

        /*Första gången vi delar kvadraten i fyra delar ger vi
        rotations variabeln sina start värden beroende på vilken
        fyrkant vi väljer*/
        if size = m then
            algo(matris, m, size/2, x+size/2, y, 0)
            algo(matris, m, size/2, x+size/2, y + size/2, 1)
            algo(matris, m, size/2, x, y, 2)
        else
            /*size står för det aktuella kvadratens bredd.
            Gemensamt för dessa anrop är att de hela tiden
            väljer nya kvadrater för att sen fortsätta skapa nya.*/
            algo(matris, m, size/2, x+size/2, y, (r != 1 &&
            r != 2 && r != 3 ? 0 : 3))
            algo(matris, m, size/2, x+size/2, y + size/2,
            (r != 2 ? 1 : 2))
            algo(matris, m, size/2, x, y, (r != 1 ? 2 : 1))

        /*Den fjärde kvadraten kan antingen ha rotations värdet 0
        eller 3 om det föregående roationsvärdet också va 3*/
        algo(matris, m, size/2, x, y + size/2, (r != 3 ? 0 : 3))

        //Returnerar matrisen
        return matris
```

Komplexitet

Då man för varje besök av en 2x2 kvadrat i m x m matrisen lägger till ett L och returnerar en matris där L täcker alla platser utom en så har man besökt alla platser i matrisen utom en. Därför blir tidskomplexiteten för denna algoritm O(m*m) = O(m^2), där m är matrisens sidolängd.

Korrekthetsbevis

För att algoritmen ska anses vara korrekt bör den ta emot ett jämnt heltal m som indata och returnera en nästan täckt m x m matris. Detta sker genom att placera L-bitar med ytan 3 som bör täcka alla platser utom en i matrisen. Algoritmen bör även använda sig av dekomposition, som går ut på att dela upp problemet i mindre delar och lösa delproblemen för att sen kombinera resultaten. Tidskomplexiteten bör vara polynomisk i m.

För vilket m vi än väljer kommer algoritmen alltid att dela upp matrisen i fyra mindre kvadrater tills man når basfallet då kvadraten man precis undersöker har storleken 2 x 2. Sättet L:en läggs till varierar beroende på vilken rotationen som det föregående L:et hade när man valde den nuvarande kvadraten. Detta kommer ske på ett sätt som resulterar i att det kommer finns en plats i kvadraten som inte används. Avslutningsvis spelar det ingen roll hur stort m är, algoritmen kommer alltid att rekursivt dela upp matrisen i mindre delar, “slå ihop” delarna och lägga till L tills den är klar. Tidskomplexiteten för algoritmen är polynomisk m^2 då man besöker varje plats i matrisen utom en.