Analys - Quadtrees

Programkonstruktion och datastrukturer

Mikael Sernheim & Anton Gildebrand, IT1, 2013-02-01

emptyQtree

Tidskomplexiteten för funktionen emptyQtree är alltid Θ(1) eftersom den bara skapar ett tomt träd, oavsett vad argumentet har för värde. Det sker alltså inga rekursiva anrop och argumenten bearbetas inte.

insert

x = Höjden på trädet
T_{match}=Tiden för mönstermatchning och jämförelser

$$T(x) = \begin{cases} T_{match} & \text{if } x = 0 \\ T(x-1) + T_{match} & \text{if } x > 0 \end{cases}$$

Sluten form

Med hjälp av Theorem 1 kommer vi fram till att den slutna formen för tidskomplexiteten av insert blir Θ(x)

Worst-case

A = Arean av extenten i rot-trädet.

Värsta fallet uppnås när rektangeln som ska föras in är tillräckligt liten och befinner sig i ett hörn. D.v.s. när man måste söka i i minsta möjliga delträd för att hitta den. Genom tester kom vi fram till att tidskomplexiteten är $\Theta(\text{Log}_4(A))$, vi ser sedan att $\text{Log}_4(A)$ är detsamma som höjden på trädet. Det betyder att kan säga något om komplexiteten utan att veta djupet på trädet, förutsatt att vi vet arean på rot-trädets extent.

Givet att A är konstant så blir då tidskomplexiteten för insert Θ(1).

query

Hjälpfunktionen queryList kommer alltid ha komplexiteten Θ(n), där n är längden på listan.

x=Höjden på trädet

Denna funktions rekursion är av samma karaktär som insert. Vi ser att metodiken för att söka i och att lägga in i trädet är likadan, det som skiljer sig är att vi i query använder oss av hjälpfunktionen queryList samt utför append flera gånger i varje rekursivt anrop av query.

I analysen av det värsta fallet är det givet att varje nod har konstant antal rectangles i sina listor (C=antalet rektanglar i unionen av Horizontal och Vertical). Vi antar då att det värsta fallet här blir att alla rektanglar i listorna har punkten (a,b) i sig. Vi använder oss av append två gånger; först då vi sätter ihop queryList (horizontal) med querList(vertical) sedan då vi appendar detta med rekursionen. I varje nod kommer vi att anropa den rekursiva funktionen queryList och appenda resultatet från denna. Detta ger oss 5C/2 rekursioner (summan av de rekursioner som append och queryList utför) i respektive nod. I vår rekursiva form på **query** försummar vi detta och skriver det som T_{rect} som representerar tiden för att hitta rektanglarna som innehåller punkten (a,b) i den aktuella noden.

Vi förutsätter att punkten (a,b) inte ligger på någon mittlinje då detta avbryter rekursionen och således inte är det värsta fallet.

$$T_{query}(x)=\{$$

$$T_{rect} \quad \text{if } x=0$$

$$T_{query}(x-1) + T_{rect} \quad \text{if } x>0$$

$$\}$$

Vi använder oss av induktionen nedan för att komma fram till en sluten form på worst-case för $T_{query}(x)$.

Rekursiv form

$$T_{query}(x) = \begin{cases} T_{rect}(c) & \text{if } x = 0 \\ T_{query}(x-1) + T_{rect}(c) & \text{if } x > 0 \end{cases}$$

"Expansion method" - för att upptäcka mönster för olika värden

$T_{query}(0) = T_{rect}(c)$

$T_{query}(1) = T_{query}(0) + T_{rect}(c) = 2 \cdot T_{rect}(c)$

$T_{query}(2) = T_{query}(1) + T_{rect}(c) = 3 \cdot T_{rect}(c)$

Gissning: $T_{query}(x) = (x+1)\, T_{rect}(c)$

Induktionsbevis: Visa att P_0 är sann

$V.L._0 = T_{rect}(c)$ $\quad H.L._0 = (0+1)\, T_{rect}(c) = T_{rect}(c) = V.L._0$

Vi antar att P_p är sann dvs $T_{query}(p) = (p+1)\, T_{rect}(c)$

Visa att P_{p+1} är sann

$T_{query}(p+1) = T_{query}(p) + T_{rect}(c) = (p+1)\, T_{rect}(c) + T_{rect}(c) =$

$= (p+2)\, T_{rect}(c)$ $\quad H.L_{p+1} = (p+2)\, T_{rect}(c) = V.L._{p+1}$

V. S. B.