

Data Structures and Algorithms

Exercise 3

Markus Innerebner, Romans Kasperovics
dsa(a)inf.unibz.it

Free University of Bozen - Bolzano, Italy

Thursday, March 12, 2009

Implement in C the tromino tiling algorithm from the lecture notes. Model the $2^n \times 2^n$ board with a two dimensional array of integers. Model the empty cells with '0' and the initial hole with '1'. Fill it in with numbers >1 so that same numbers belong to the same tile. Print the array after filling.

```
#include <stdio.h>
```

```
#define N 8
```

```
int B[N][N],c=2;
```

```
// a,b – the top-left corner of the subproblem
```

```
// n – the size of the subproblem
```

```
// x,y – the location of the occupied cell
```

```
void tile(int a, int b, int n, int x, int y);
```

Solution 03

Task 1

```
int main() {
    int i,j;
    // initialization
    for (i=0; i<N; i++) // for each row
        for (j=0; j<N; j++) // for each column
            B[i][j] = 0;
    B[1][3] = 1; // the initial hole
    // tiling
    tile(0,0,N,1,3);
    // printing
    for (i=0; i<N; i++) { // for each row
        for (j=0; j<N; j++) // for each column
            printf("%3d",B[i][j]);
        printf("\n");
    }
    return 0;
}
```

Solution 03

Task 1

```
void tile(int a, int b, int n, int x, int y) {  
    if (n  $\geq$  2) {  
        int tileno = c++;  
        if (x  $\geq$  a+n/2 || y  $\geq$  b+n/2) { // (x,y) is not in the top-left quadrant  
            B[a+n/2-1][b+n/2-1] = tileno; tile(a,b,n/2,a+n/2-1,b+n/2-1);  
        } else tile(a,b,n/2,x,y);  
        if (x < a+n/2 || y  $\geq$  b+n/2) { // (x,y) is not in the top-right quadrant  
            B[a+n/2][b+n/2-1] = tileno; tile(a+n/2,b,n/2,a+n/2,b+n/2-1);  
        } else tile(a+n/2,b,n/2,x,y);  
        if (x  $\geq$  a+n/2 || y < b+n/2) { // (x,y) is not in the bottom-left quadrant  
            B[a+n/2-1][b+n/2] = tileno; tile(a,b+n/2,n/2,a+n/2-1,b+n/2);  
        } else tile(a,b+n/2,n/2,x,y);  
        if (x < a+n/2 || y < b+n/2) { // (x,y) is not in the bottom-right quadrant  
            B[a+n/2][b+n/2] = tileno; tile(a+n/2,b+n/2,n/2,a+n/2,b+n/2);  
        } else tile(a+n/2,b+n/2,n/2,x,y);  
    }  
}
```

For the tiling algorithm from Task 1

- Show the exact running time $T(n)$ as a recurrence

Let $C > 0$ is some constant and n is the size of the array

$$T(n) = \begin{cases} 1 & \text{if } n < 2 \\ 4T\left(\frac{n}{2}\right) + C & \text{if } n \geq 2 \end{cases}$$

- Guess a tight asymptotic upper bound by expanding the recurrence by substitution and noticing the pattern

$$T(n) = 4T\left(\frac{n}{2}\right) + C = 4\left(4T\left(\frac{n}{4}\right) + C\right) + C = \dots = 4^i T\left(\frac{n}{2^i}\right) + \frac{1-4^i}{1-4} C$$

The recursion stops when $\frac{n}{2^i} < 2$, i.e., $i > \log_2 n - 1$, $i \geq \log_2 n$.

$$\text{At } i = \log_2 n, T(n) = n^2 + \frac{C}{3}n^2 - \frac{C}{3}.$$

For the tiling algorithm from Task 1

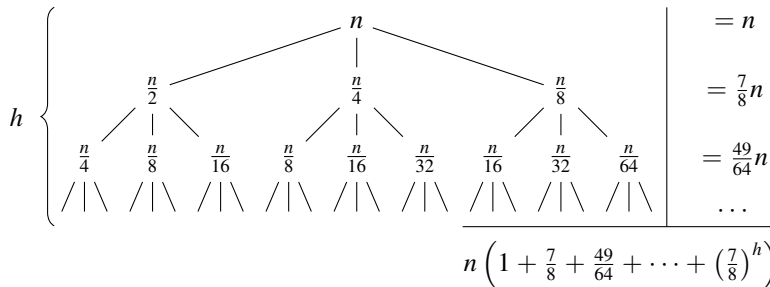
- Verify the solution by the mathematical induction

Our guess: $T(n) \leq kn^2 - d$, where k, d are some constants

Inductive step: $4T\left(\frac{n}{2}\right) + C \leq 4\left(k\frac{n^2}{4} - d\right) + C \stackrel{?}{\leq} kn^2 - d$

Proof: holds for all $C \leq 3d$

$$T(n) = \begin{cases} 1 & , \text{ if } n = 1 \\ T(n/2) + T(n/4) + T(n/8) + n & , \text{ if } n > 1 \end{cases}$$



- The tree grows until $\frac{n}{2^h} = 1$; thus, $h = \lg n$

- $n \left(1 + \frac{7}{8} + \frac{49}{64} + \dots + \left(\frac{7}{8}\right)^h \right) = n \cdot \frac{1 - \left(\frac{7}{8}\right)^{\lg n + 1}}{1 - \frac{7}{8}} = 8n \left(1 - \frac{7}{8} n^{\frac{1}{\log_7/8^2}} \right)$

- $n^{\frac{1}{\log_7/8^2}} \approx \frac{1}{\sqrt[5.19]{n}}$

$$T(n) = \begin{cases} 1 & , \text{if } n = 1 \\ T(n/2) + T(n/4) + T(n/8) + n & , \text{if } n > 1 \end{cases}$$

Our Guess: $T(n) \leq cn$

Inductive Step:

- $T(n/2) \leq c\frac{n}{2}, T(n/4) \leq c\frac{n}{4}, T(n/8) \leq c\frac{n}{8}$
- $c\frac{n}{2} + c\frac{n}{4} + c\frac{n}{8} + n \stackrel{?}{\leq} cn$

Proof:

- $\frac{7}{8}cn + n \leq cn$, dividing both parts by n we get $\frac{7}{8}c + 1 \leq c$
- $\frac{1}{8}c \geq 1, c \geq 8$

Asymptotic Complexity: for $c \geq 8, T(n) = O(n)$