Informatik II (D-ITET)

Tutorial 12

TA: Anwar Hithnawi, E-mail: hithnawi@inf.ethz.ch
Distributed Systems Group, ETH Zürich
L11.A1 – Sorting by Search Trees

1. Build a binary search tree out of the elements
2. Traverse the tree in-order and copy the elements back into the array

- In the best case, the values in the list are well-mixed → balanced tree
- In the worst case, the values in the list are sorted in ascending or descending order → degenerate tree

- Complexity
  - In best case: $O(n \cdot \log n)$
  - In average case: $O(n \cdot \log n)$
  - In worst case: $O(n^2)$
L11.A2 – Complexity Analysis

// Fragment 1
for (int i=0; i<n; i++)
a++;

Computation:
a++ is executed \( n \) times
\[ \Rightarrow \text{Total executions: } n \sim O(n) \]

// Fragment 2
for (int i=0; i<2n; i++)
a++;
for (int j=0; j<n; j++)
a++;

Computation:
- a++ is executed \( 2n \) times
- a++ is executed \( n \) times
\[ \Rightarrow \text{Total executions: } 2n+n=3n \sim O(n) \]
L11.A2

// Fragment 3
    for (int i=0; i<n; i++)
        for (int j=0; j<n; j++)
            a++;

Computation:
• Outer loop is executed \( n \) times
• Inner loop executes \( a++ \) \( n \) times

→ Total executions:
\( n\times n = n^2 \sim 0(n^2) \)
// Fragment 4
for (int i=0; i<n; i++)
    for (int j=0; j<i; j++)
        a++;

Computation:
• Outer loop is executed $n$ times
• Inner loop executes $a++$ $i$ times

\[
Gesamtaufwand = \sum_{i=1}^{n} i = \frac{n(n-1)}{2} \sim O(n^2)
\]
// Fragment 5
while (n >= 1)
    n = n/2;

Berechnung:  - while-Schleife ruft \( \frac{n}{2} \) Mal die Operation \( n = \frac{n}{2} \) auf

\[
\underbrace{\frac{n}{2 \cdot 2 \cdot \ldots \cdot 2}}_{x \text{ mal}} = 1 \iff n = 2^x
\]

\( \Rightarrow \) Gesamtaufwand = \( x = \log_2(n) \) \( \sim \) \( O(\log_2(n)) \)
for (int i=0; i<n; i++)
    for (int j=0; j<n*n; j++)
        for (int k=0; k<j; k++)
            a++;

Computation:
• Outer loop is executed $n$ times
• Following loop is executed $n^2$ times
• Inner loop executes $a++$ $n$ times

$$Gesamtaufwand = n \cdot j = n \cdot (1 + 2 + \cdots + n^2)$$

$$= n \cdot \sum_{j=1}^{n^2} j = n \cdot \frac{n^2(n^2 - 1)}{2} \sim O(n^5)$$
L11.A3 – Complexity (I)

\[ t_{op} = \frac{1}{3} t_{op} \]

- Time per operation
- Input size
- Total running time

\[ M' \]
\[ T'_{tot} \]
### L11.A3 – Complexity (II)

<table>
<thead>
<tr>
<th>(O(...))</th>
<th>(T_{tot})</th>
<th>(T'_{tot})</th>
<th>(T'<em>{tot} = T</em>{tot})</th>
</tr>
</thead>
</table>
| \(O(n)\)   | \(T_{tot} = t_{op} \cdot M_1\) | \(T'_{tot} = t'_{op} \cdot M_1'\) | \(t_{op} \cdot M_1' = t_{op} \cdot M_1 \Rightarrow \frac{1}{3} t_{op} \cdot M_1' = t_{op} \cdot M_1\) \\
|             |             |             | \(\Rightarrow M_1' = 3M_1\) |
| \(O(n^2)\) | \(T_{tot} = t_{op} \cdot M_2^2\) | \(T'_{tot} = t'_{op} \cdot M_2^{2'}\) | \(t_{op} \cdot M_2^{2'} = t_{op} \cdot M_2^2 \Rightarrow \frac{1}{3} t_{op} \cdot M_2^{2'} = t_{op} \cdot M_2^2\) \\
|             |             |             | \(\Rightarrow M_2^{2'} = 3M_2^2 \Rightarrow M_2' = \sqrt{3}M_2\) ~1,7 |
| \(O(2^n)\) | \(T_{tot} = t_{op} \cdot 2^{M_3}\) | \(T'_{tot} = t'_{op} \cdot 2^{M_3'}\) | \(t_{op} \cdot 2^{M_3'} = t_{op} \cdot 2^{M_3} \Rightarrow \frac{1}{3} t_{op} \cdot 2^{M_3'} = t_{op} \cdot 2^{M_3}\) \\
|             |             |             | \(\Rightarrow 2^{M_3'} = 2^{M_3} \cdot 3 \Rightarrow M_3' = M_3 + \log_2 3\) ~1,5 |
| \(O(\log_2 n)\) | \(T_{tot} = t_{op} \cdot \log_2 M_4\) | \(T'_{tot} = t'_{op} \cdot \log_2 M_4'\) | \(t_{op} \cdot \log_2 M_4' = t_{op} \cdot \log_2 M_4\) \\
|             |             |             | \(\Rightarrow \frac{1}{3} t_{op} \cdot \log_2 M_4' = t_{op} \cdot \log_2 M_4\) \\
|             |             |             | \(\Rightarrow \log_2 M_4' = 3 \cdot \log_2 M_4 \Rightarrow M_4' = 2^{3 \cdot \log_2 M_4}\) \\
|             |             |             | \(\Rightarrow M_4' = \left(2^{\log_2 M_4}\right)^3 \Rightarrow M_4' = \left(M_4\right)^3\) |
L11.A4 – A knight on a chess board
L11.A4 – Numbers...

<table>
<thead>
<tr>
<th>Board</th>
<th>Number of knight’s tours</th>
<th>Number of closed knight’s tours</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 x 4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5x5</td>
<td>1728</td>
<td>0</td>
</tr>
<tr>
<td>6x6</td>
<td>9862</td>
<td>≥ 5</td>
</tr>
<tr>
<td>8x8</td>
<td>?</td>
<td>1.6 \cdot 10^{15}</td>
</tr>
</tbody>
</table>

L11.A4b – Backtracking

- Find a path...
  - which goes over all fields
  - and visits each field only once

- Early termination
  - There is no maximum depth relatively simple
  - Backtracking when the next field already in the path

- Search efficiency
  - Code static values in a static manner
  - Linear search(ArrayList) replace with your own position-set-query
HINTS ON U12

A1 – Heapsort
A2 – Parallelized Mergesort (Threads)
A3 – Recursive Problem Solving
U12.A1 – Heap

A heap is a binary tree in which:

- All levels (except possibly the last) are completely filled
- The last level is filled from the left
- For all $k$ nodes (except the root):
  - $\text{value}(\text{previous}(k)) \leq \text{value}(k)$ in a MIN-Heap
  - Or $\geq$ in a MAX-Heap

Properties (MIN-Heap):

- Root has the smallest value
- All paths from the root to a leaf are monotonically increasing
U12.A1 – Heap

Heap as Array

Heap as tree

insert
U12.A1a,b – Properties of Heaps

- How many elements are in a heap of height \( h \) containing minimum and maximum?

- Is a sorted array a heap (if it is interpreted as a binary tree)? And vice versa?
U12.A1c – HeapSort

Phase 1
Array converted to Heap

Phase 2
Read sorted Heap: remove from the root

6 5 3 1 8 7 2 4


2-phases

As in A1c

Take care of requirements for sort (copy!)

Note: all HeapSort operations are 'in-place'
U12.A2 – Parallelized Merge Sort

a) Much is up to you
- `u10a1.ISort` you still (hopefully) have
- `ISort.sort`: returns a sorted *copy* of the vector
- Your `MergeSort` class should provide a way to select the number of parallel threads

b) 1'000'000 Integers
- A main class to perform the measurements
- Here also U10.A1 offers a reference
- An important indication of your measurements is the number of available CPU cores on your system (Google helps)
- Don't forget the explanation!
The company Springli intends to bring a new chocolate on the market

Acceptance of all rectangular formats with a maximum of $n$ bits must be tested

How many formats in terms of $n$ must the company Springli test in the market?

Hint:
- For $n = 1, 2, 3, 4, 5, 6$ exists $1, 3, 5, 8, 10, 14$ formats.
U12.A3 – Springli Formats (I)

\[ n = 1 \rightarrow 1 \text{ Format} \]
U12.A3 – Springli Formats (II)

\[ n = 2 \rightarrow 3 \text{ Formats} \]
U12.A3 – Springli Formats (III)

n = 3 → 5 Formats
n = 4 → 8 Formats

U12.A3 – Springli Formats (IV)
U12.A3 – Springli Formats (V)

n = 5 → 10 Formats
U12.A3 – Springli Formats

Recursive Solution:

\[
\text{Formats}(n) = \text{Formats}(n-1) + \ldots
\]
That's it!
Enjoy the summer and good luck with the exam!