Tutorial 2

Algorithms and Data Structures, 2011

Jonathan Cederberg <jonathan.cederberg@it.uu.se>

Firday, September 16th, 2011

Outline

Outline

Comments

- I have still not received any questions on e-mail. Use this possibility to influence these sessions.

Comments

- I have still not received any questions on e-mail. Use this possibility to influence these sessions.
- Answers without explicit justifications give 0 pts in an exam.

Comments

- I have still not received any questions on e-mail. Use this possibility to influence these sessions.
- Answers without explicit justifications give 0 pts in an exam.
- Numerical justification is no justification.

Comments

- I have still not received any questions on e-mail. Use this possibility to influence these sessions.
- Answers without explicit justifications give 0 pts in an exam.
- Numerical justification is no justification.
- Please write your name on the assignment, just as you would if you'd hand in a paper copy.

Important facts

- The definitions of $\Omega, \mathcal{O}.\Theta$

Important facts

- The definitions of $\Omega, \mathcal{O}.\Theta$
- A sum grows according to its fastest growing term (the left one)

Important facts

- The definitions of $\Omega, \mathcal{O}.\Theta$
- A sum grows according to its fastest growing term (the left one)
- Polynomials grow faster than polylogs

Important facts

- The definitions of $\Omega, \mathcal{O}.\Theta$
- A sum grows according to its fastest growing term (the left one)
- Polynomials grow faster than polylogs
- Exponentials grow faster than polynomials

Important facts

- The definitions of $\Omega, \mathcal{O}.\Theta$
- A sum grows according to its fastest growing term (the left one)
- Polynomials grow faster than polylogs
- Exponentials grow faster than polynomials
- Larger exponent $\implies$ faster growth

Important facts

- The definitions of $\Omega, \mathcal{O}.\Theta$
- A sum grows according to its fastest growing term (the left one)
- Polynomials grow faster than polylogs
- Exponentials grow faster than polynomials
- Larger exponent $\implies$ faster growth
- Larger base $\implies$ faster growth

Important facts

- The definitions of $\Omega, \mathcal{O}.\Theta$
- A sum grows according to its fastest growing term (the left one)
- Polynomials grow faster than polylogs
- Exponentials grow faster than polynomials
- Larger exponent $\implies$ faster growth
- Larger base $\implies$ faster growth
- Analyzing algorithms: what is the size of input?

Important facts

- The definitions of $\Omega, \mathcal{O}.\Theta$
- A sum grows according to its fastest growing term (the left one)
- Polynomials grow faster than polylogs
- Exponentials grow faster than polynomials
- Larger exponent $\implies$ faster growth
- Larger base $\implies$ faster growth
- Analyzing algorithms: what is the size of input?
- $\mathcal{O}$ is not the same as $\leq$!

Important facts

- The definitions of $\Omega, \mathcal{O}.\Theta$
- A sum grows according to its fastest growing term (the left one)
- Polynomials grow faster than polylogs
- Exponentials grow faster than polynomials
- Larger exponent $\implies$ faster growth
- Larger base $\implies$ faster growth
- Analyzing algorithms: what is the size of input?
- $\mathcal{O}$ is not the same as $\leq$!
- Exponentiation is right associative

Important facts

- The definitions of $\Omega, \mathcal{O}.\Theta$
- A sum grows according to its fastest growing term (the left one)
- Polynomials grow faster than polylogs
- Exponentials grow faster than polynomials
- Larger exponent $\implies$ faster growth
- Larger base $\implies$ faster growth
- Analyzing algorithms: what is the size of input?
- $\mathcal{O}$ is not the same as $\leq$!
- Exponentiation is right associative
- Use logarithms with care!

1. Asymptotic growth rates

Rank these functions in order of growth:

$$\frac{n}{2} \quad 4^{\lg n} \quad n! \quad \lg n \quad 2^{2^n} \quad (n+1)!$$

$$\left(\frac{100}{99}\right)^n \quad n \lg n \quad n^3 \quad 2^n \quad n \cdot 2^n$$

2. Tradeoffs

The Merge-Sort procedure described in the lectures calls Merge once for each recursive call. Each time memory space of the same size as input is allocated (the L and R arrays). Insertion-Sort, on the other hand, allocates no extra memory, but is an algorithm with quadratic running time. For each case below, justify in *at most* three sentences which algorithm is the asymptotically better one.

1. Memory allocation is done in constant time no matter how much memory is allocated.
2. Memory allocation is done in linear time: allocating something of size n takes $\mathcal{O}(n)$.
3. Memory allocation is done in quadratic time: allocating something of size n takes $\mathcal{O}(n^2)$.

MERGE(A, p, q, r)

```
n1 ← q − p + 1
n2 ← r − q
create arrays L[1 .. n1 + 1]
and R[1 .. n2 + 1]
for i ← 1 to n1
      do L[i] ← A[p + i − 1]
for j ← 1 to n2
      do R[j] ← A[q + j]
L[n1 + 1] ← ∞
R[n2 + 1] ← ∞
i ← 1
j ← 1
for k ← p to r
      do if L[i] ≤ R[j]
            then A[k] ← L[i]
                 i ← i + 1
            else A[k] ← R[j]
                 j ← j + 1
```

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n) & \text{if } n > 1 \end{cases}$$

3. Foo, Bar and Baz

Below the procedures Foo, Bar and Baz are given in pseudocode. For Foo and Bar, answer the following four questions:

1. What does the procedure do?
2. What is the best case running time?
3. What is the average case running time?
4. What is the worst case running time?

Each answer should be justified in one sentence. Answers without justifications give 0 points.

Foo(A)

```
n ← size(A)
for i ← 1 to ⌊n/2⌋
        do if not A[i] = A[n − i + 1]
                then return False
return True
```

```
Bar(A, z)
1   return Baz(A, 1, size(A), z)
```

```
Baz(A, x, y, z)
1   if x > y or (x = y and A[x] ≠ z)
2       then return −1
3   i ← x + ⌊(y−x)/2⌋
4   if A[i] = z
5       then return i
6   if A[i] > z
7       then Baz(A, x, i − 1, z)
8       else Baz(A, i + 1, y, z)
```