Universität Innsbruck - Institut für Informatik
Prof. Clemens Ballarin, Robert Binna, Friedrich Neurauter, Francois Scharffe and Sarah Winkler

29. April 2008

Proseminar Algorithmen und Datenstrukturen

Exercise sheet 6

Exercise 1 (O Notation)

Compare the following pairs of functions in terms of order of magnitude. In each case, say whether $f(n) \in O(g(n))$, $f(n) \in \Omega(g(n))$ and/or $f(n) \in \Theta(g(n))$.

	$f(n)$	$g(n)$
a)	$n^3 + 3n + 1$	n^4
b)	$\log(n)$	n
c)	$n \cdot 2^n$	3^n
d)	n	$(\log(n))^5$
e)	$\log(n)$	$\log(n^2)$
f)	$100n + \log(n)$	$n + (\log(n))^2$

Exercise 2 (O Notation)

The addition theorem for O states that $f(n) \in O(s(n))$ and $g(n) \in O(r(n))$ imply that $f(n) + g(n) \in O(s(n) + r(n))$. Now formulate the corresponding theorem for Ω and provide a proof for it.

Exercise 3 (O Notation)

a) Find a counterexample to the following claim: $f(n) \in O(s(n))$ and $g(n) \in O(r(n))$ imply that $f(n) - g(n) \in O(s(n) - r(n))$.

b) Find a counterexample to the following claim: $f(n) \in O(s(n))$ and $g(n) \in O(r(n))$ imply that $f(n)/g(n) \in O(s(n)/r(n))$.

Exercise 4 (O Notation)

Prove or disprove (by means of a counterexample) the following statements:

a) $O(f(n) + g(n)) = O(\max(f(n), g(n)))$

b) $O(f(n) + g(n)) = O(\min(f(n), g(n)))$

c) $\Omega(f(n) + g(n)) = \Omega(\max(f(n), g(n)))$

d) $\Omega(f(n) + g(n)) = \Omega(\min(f(n), g(n)))$

e) $\Theta(f(n) + g(n)) = \Theta(\max(f(n), g(n)))$

f) $\Theta(f(n) + g(n)) = \Theta(\min(f(n), g(n)))$

Exercise 5 (O Notation)

Assuming that the bodies of the for-loops only contain simple statements (no loops, no function calls etc.), and hence have constant time complexity, what is the time complexity of each of the following code fragments?

a)
```
for (i = 0; i < N; i++) {
  ...
}
for (j = 0; j < M; j++) {
  ...
}
```

b)
```
for (i = 0; i < N; i++) {
  for (j = 0; j < N; j++) {
    ...
  }
}
for (k = 0; k < N; k++) {
  ...
}
```

c)
```
for (i = 0; i < N; i++) {
  for (j = i; j < N; j++) {
    ...
  }
}
```

d) Consider this sample implementation for the longest upsequence algorithm (exercise 4 on exercise sheet 3). What is its complexity expressed in O notation? Consider both cases, binary and naive linear search.

```
#include <stdio.h>

int main() {
  int n = 8; /* array length */
  int X[] = {1, 3, 4, 6, 2, 4, 4, 0};
  int M[n]; for (int i=0; i<n; i++) M[i]=0;
  int i;

  int k = 0;
  M[0] = X[0];
  for(i = 1; i < n; i++){
    if (X[i] >= M[k]) {
      k++;
      M[k] = X[i];
      }
    else if (X[i] < M[0]) {
      M[0] = X[i];
      }
    else {
      /* find j such that M[j-1] <=X[i]<M[j] */
      /* Possibility 1: naive search
      int j = i;
      while (M[j-1] > X[i])
        j--; */
      /* Possibility 2: binary search */
      int b, j = 1;
      for (b = (k+1)/2; M[j-1] > X[i] || X[i] >= M[j]; b /= 2)
        if (M[j-1] > X[i])
          j -= b;
        else
          j += b;
      M[j] = X[i];
    }
  }
  printf("Longest upsequence has length %d\n", k + 1);
  return 0;
}
```

Exercise 6 (Linked List)

Reconsider your linked list implementation of last week, try to eliminate the bugs it has (if it has any) and provide a demo program that is equipped with a user interface. The user interface should consist of a menu that has entries for

- adding elements to a list,
- removing elements from a list,
- printing a list,
- inserting elements in order and
- searching for a specific element.