Warm-up

Problem 1. Sort the following functions in increasing order of asymptotic growth

$$n, n^3, n\log n, n^n, \frac{3^n}{n^2}, n!, \sqrt{n}, 2^n$$

Problem 2. Sort the following functions in increasing order of asymptotic growth

$$\log\log n, \log n!, 2^{\log\log n}, n^{\frac{1}{\log n}}$$

Problem 3. Consider the following pseudo-code fragment.

```
def stars(n):
  for i in [1:n]:
    print '*' i many times
```

a) Using the O-notation, upperbound the running time of STARS.

b) Using the Ω-notation, lowerbound the running time of STARS to show that your upperbound is in fact asymptotically tight.

Problem 4. Recall the problem we covered in lecture: Given an array A with n entries, find $0 \le i < j < n$ maximizing $A[i] + \cdots + A[j]$.

Prove that the following algorithm is incorrect: Compute the array B as described in the lectures. Find i minimizing $B[i]$, find j maximizing $B[j+1]$, return (i, j).

Come up with the smallest example possible where the proposed algorithm fails.

Problem solving

Problem 5. Given an array A consisting of n integers, we want to compute the upper triangle matrix C where

$$C[i][j] = \frac{A[i] + A[i+1] + \cdots + A[j]}{j - i + 1}$$

for $0 \le i \le j < n$. Consider the following algorithm for computing C:

a) Using the O-notation, upperbound the running time of SUMMING-UP.

b) Using the Ω-notation, lowerbound the running time of SUMMING-UP.

```
def summing_up(A)
  C ← new matrix of len(A) by len(A)
  for i in [0:n-1]
    for j in [i:n-1]
      compute average of entries A[i:j]
      store result in C[i, j]
  return C
```

Problem 6. Come up with a more efficient algorithm for computing the above matrix $C[i][j] = \frac{A[i]+A[i+1]+\cdots+A[j]}{j-i+1}$ for $0 \leq i \leq j < n$. Your algorithm should run in $O(n^2)$ time.

Problem 7. Give a formal proof of the transitivity of the O-notation. That is, for function f, g, and h show that if $f(n) = O(g(n))$ and $g(n) = O(h(n))$ then $f(n) = O(h(n))$.

Problem 8. Given an array with n integer values, we would like to know if there are any duplicates in the array. Design an algorithm for this task and analyze its time complexity.

Advanced problem solving

Problem 9. Recall the problem we covered in lecture: Given an array A with n entries, find $0 \leq i < j < n$ maximizing $A[i] + \cdots + A[j]$. Consider the following algorithm that computes two candidate solutions and returns the best of the two.

```
def opt_strategy(A)
  # find (i, j) maximizing A[i] + ... + A[j]

  curr_val, curr_ans ← 0, (None, None)
  n ← len(A)
  # compute B[i] = A[0] + .. + A[i-1]
  B ← new array[n + 1]
  B[0] ← 0
  for i in [0:n]:
    B[i+1] ← B[i] + A[i]

  j_0 ← index j : 0 < j < n maximizing B[j + 1]
  i_0 ← index i : i < j_0 minimizing B[i]

  i_1 ← index i : i < n minimizing B[i]
  j_1 ← index j : i_1 < j < n maximizing B[j + 1]

  return best solution among (i_0, j_0) and (i_1, j_1)
```

a) Prove or disprove the correctness of the algorithm.

b) Analyze its time complexity.