

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.

```

1  def stars(A):
2      for i in [1:n]:
3          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 $0 \leq i < j < n$ maximizing $A[i] + \dots + A[j]$.

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

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

for $0 \leq i \leq j < n$. 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.

```
1  def summing_up(A)
2      C = new matrix of len(A) by len(A)
3      for i in [0:n]
4          for j in [i:n]
5              compute average of entries A[i:j+1]
6              store result in C[i, j]
7      return C
```

Problem 6. Come up with a more efficient algorithm for computing the above matrix $C[i][j] = \frac{A[i]+A[i+1]+\dots+A[j]}{j-i+1}$ 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.