Design and Analysis of Algorithms I

Asymptotic Analysis

The Gist

Motivation

Importance: vocabulary for the design and analysis of algorithms (e.g., "big-Oh" notation)

- "sweet spot" for high-level reasoning about algorithms
- coarse enough to suppress architecture/language/compiler-dependent details
- sharp enough to make useful comparisons between different algorithms, especially on large inputs (e.g., sorting or integer multiplication)

Asymptotic Analysis

High-level idea: suppress constant factors and lower-order terms.

too system-dependent

irrelevant for large inputs

Example: equate $6n\log_2 n + 6n$ with just $n \log n$.

Terminology: running time is $O(n \log n)$

["big-Oh" of $n \log n$]

where n = input size (e.g., length of input array)

Example: One Loop

Problem: does array A contain the integer t?

given A (array of length n)
and t (an integer)

for i = 1 to n
  if A[i] == t return TRUE
return FALSE

Question: what is the running time?

(A) $O(1)$
(B) $O(\log n)$
(C) $O(n)$
(D) $O(n^2)$

Example: Two Loops

given A, B (arrays of length n) and t (an integer) [does A or B contain t?]

```
for i = 1 to n
    if A[i] == t return TRUE
for i = 1 to n
    if B[i] == t return TRUE
return FALSE
```

Question: running time?

(A) $O(1)$

(B) $O(\log n)$

(C) $O(n)$

(D) $O(n^2)$

Example: Two Nested Loops

Problem: do arrays A, B have a number in common?

given arrays A, B of length n

```
for i=1 to n
    for j=1 to n
        if A[i]==B[j] return TRUE
return FALSE
```

Question: running time?

(A) $O(1)$
(B) $O(\log n)$
(C) $O(n)$
(D) $O(n^2)$

Example: Two Nested Loops (II)

Problem: does array A have duplicate entries?

given array A of length n

```
for i = 1 to n
    for j = i+1 to n
        if A[i] == A[j] return TRUE
return FALSE
```

Question: running time?

(A) $O(1)$

(B) $O(\log n)$

(C) $O(n)$

(D) $O(n^2)$