

Scalar Machine Model

- Any fundamental operation (+, -, *, /, ^, =, <, <=, >, >=, <>, and, or, not, etc.) takes one time unit.
 - If necessary, count *, /, ^ separately.
- Only unary or binary operations can be performed in unit time.
- Only scalar operands can be handled in unit time.

12

Big-Oh Notation

- “On the order of ...”
 - For large n, time grows as f(n)
 - (5n+4) = O(n)
 - Note special use of equal sign!
- Asymptotic behavior
 - Dominant term as problem size grows
- Variable n always represents the problem size.

20

Big-Oh Definition

$T(n) = O(f(n))$

if there are constants c
and n_0 such that

$T(n) \leq cf(n)$
when

21

Related Definitions

$O(f(n))$ Upper bound (\leq)

$\Omega(f(n))$ Lower bound (\geq)

$\Theta(f(n))$ Upper/lower bound

$o(f(n))$ Upper bound ($<$)

Use only big-oh?

22

Big-Oh of Aggregates

lf:

$T_1(n) = O(f(n))$

Then:

$T_1(n) + T_2(n) = \max(O(f(n)), O(g(n)))$

23

Logarithms

$\log_k n = O(\log n)$

for constants a, b and variable u:

$\log_a n = \log_a b \log_b n$

$\frac{\log_a n}{\log_a b} = \log_b n$

denominator is a constant!

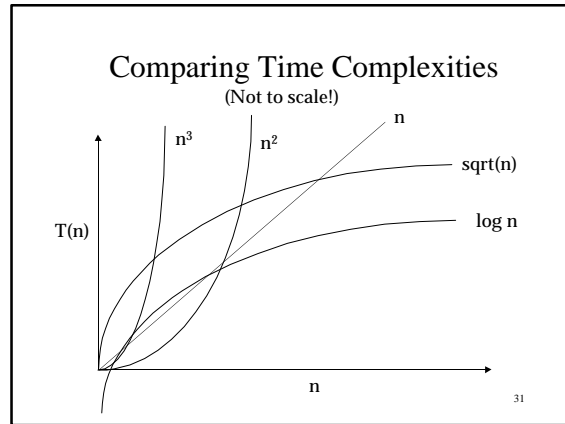
28

Logarithmic Time

- Better than linear
- Constant-time operation ...
- Cuts problem size by fraction
- Example: binary search
 - Cut problem in half with each comparison

$$T(n) = T(n/2) + C = O(\log n)$$

30



Big-Oh Counting Rules

- FOR loops
 - (iterations) * (loop statement(s) time)
- Nested loops
 - Product of loop sizes (iterations)
 - How many times body is executed
- Don't forget test time (part of body)

32

Big-Oh Counting Rules

- Consecutive Sequential Statements
 - Only the longest one counts
- IF/ELSE
 - Test time (evaluate test expression)
 - Plus longer branch
 - "true" part or "false" part

33

Some Big-Oh Categories

- Constant
- Sublinear - faster than $O(n)$
 - $\log n$, \sqrt{n} .
- Linear - $O(n)$
- Polynomial - $O(n^k)$
- Exponential - $O(a^n)$

34

Find: Maximum Subsequence Sum

Find subsequence (in input sequence) with largest sum.

2, -3, 1.5, -1, 3, -2, -3, 3

Find: Maximum Subsequence Sum

Find subsequence (in input sequence) with largest sum.

2, -3, 1.5, -1, 3, -2, -3, 3
3.5

Maximum Subsequence Sum

Find subsequence (in input sequence) with largest sum.

Four different algorithms:

- Simple-minded - $O(n^3)$
- Improved - $O(n^2)$
- Recursive - $O(n \log n)$
- Optimal - $O(n)$

40

 $O(n^3)$ Max Subsequence Alg.

```
for (int i=0; i < size-1; i++)
{
  for (int j=i; j < size; j++)
  {
    sum = 0;
    for (int k=i; k <= j, k++)
      sum += seq[k];
    if (sum > max)
    {
      max = sum;
      record(sum, i, j);
    }
  }
}
```

41

 $O(n^2)$ Max Subsequence Alg.

```
for (int i=0; i < size-1; i++)
{
  sum = 0;
  for (int j=i; j < size; j++)
  {
    sum += seq[j];
    if (sum > max)
    {
      max = sum;
      record(sum, i, j);
    }
  }
}
```

42

 $O(n \log n)$ Max. Subseq. Alg.

- Divide sequence in half
 - Max subsequence is in either:
 - Left half
 - Right half
 - Spans Left and Right half
 - This decision can be made in $O(n)$ time.
- Recursively do this on Left & Right half!
 - This will be done $O(\log n)$ times!

43

 $O(n)$ Max. Subseq. Sum Alg.

- Start summing subsequence from left end. Whenever subsequence is negative, clear sum and update i to current index.
- Do the same starting at the right end, working left, updating j when subsequence is negative.
- Algorithm terminates when the two sliding pointers meet (or cross).

44

Find: Maximum Subsequence Sum

Find subsequence (in input sequence)
with largest sum by elimination

