CS250 Winter 2003

Lecture 2: Algorithms
Binary search

Initially, low=0 and high=n-1. We compare k with the word in the middle position, that is, the one with rank mid=floor((low+high)/2).
There are three cases:

1. If k = word at rank mid, then we have found the word we were looking for and the search terminates successfully.

2. If k < word at rank mid, then we \textbf{recursively} on the first half of the range of ranks leftover, that is, low:=low and high:=mid-1.

3. If k > word at rank mid, then we \textbf{recursively} with low:=mid+1 and high:=high.
Algorithm BinarySearch(D,k,low,high)

if (low > high) then
  return(FAILURE)
else
  mid := floor((low+high)/2)

  if (k = word at rank mid) then
    return(word at rank mid)

  else if (k < word at rank mid) then
    return(BinarySearch(D,k,low,mid-1))

  else if (k > word at rank mid) then
    return(BinarySearch(D,k,mid+1,high))
Illustration of BinarySearch(D,28,0,9) in [2,4,5,7,8,9,14,22,28,31]

[2, 4, 5, 7, 8, 9, 14, 22, 28, 31]
low    mid    high

[2, 4, 5, 7, 8, 9, 14, 22, 28, 31]
  low    mid    high

[2, 4, 5, 7, 8, 9, 14, 22, 28, 31]
  low mid high
ALGORITHM product(a, b)

Input: a $\geq 0$ and b $\geq 0$ are integers.

Output: the product of a and b.

r $\leftarrow$ 0

for i $\leftarrow$ 1 to a do
    r $\leftarrow$ r + b

return r
\textbf{Algorithm} \texttt{product}(a, b)

- \textit{Input}: $a \geq 0$ and $b \geq 0$ are integers.
- \textit{Output}: the product of $a$ and $b$.

\textbf{if} $a > b$ \textbf{then}
\begin{itemize}
  \item $t \leftarrow a$
  \item $a \leftarrow b$
  \item $b \leftarrow t$
  \item $r \leftarrow 0$
\end{itemize}
\textbf{for} $i \leftarrow 1$ \textbf{to} $a$ \textbf{do}
\begin{itemize}
  \item $r \leftarrow r + b$
\end{itemize}
\textbf{return} $r$
A recursive definition of multiplication

\[ a \times b = \begin{cases} 
0 & \text{if } a = 0 \\
 b + (a - 1) \times b & \text{otherwise}
\end{cases} \]

Corresponding recursive algorithm for multiplication

```
ALGORITHM product(a, b)
    Input: b \geq a \geq 0 are integers.
    Output: the product of a and b.
    if a = 0 then
        return 0
    else
        return b + product(a - 1, b)
```
Multiplication of 3 and 5

1. calling product(3,5) returns $5 + \text{product}(2,5)$,
2. calling product(2,5) returns $5 + \text{product}(1,5)$,
3. calling product(1,5) returns $5 + \text{product}(0,5)$ and
4. calling product(0,5) returns 0.

Therefore:

1. $\text{product}(1,5)$ returns $5 + 0 = 5$,
2. $\text{product}(2,5)$ returns $5 + (5 + 0) = 10$ and
3. $\text{product}(3,5)$ returns $5 + (5 + (5 + 0)) = 15$. 
Standard Multiplication

and

Multiplication à la russe

Quadratic-time Methods
Another recursive definition of multiplication

\[ a \times b = \begin{cases} 
0 & \text{if } a = 0 \\
2b^{a/2} & \text{if } a > 0 \text{ is even} \\
b + 2b^{a-1/2} & \text{otherwise}
\end{cases} \]

Corresponding recursive algorithm for multiplication

```
ALGORITHM product(a, b)
    Input: b \geq a \geq 0 are integers.
    Output: the product of a and b.
    if a = 0 then
        return 0
    else
        h ← product([a/2], b)
        if a is even then
            return h + h
        else
            return b + h + h
```
Multiplication of 10 and 16

1. calling product(10,16) returns product(5,16) + product(5,16),
2. calling product(5,16) returns 16 + product(2,16) + product(2,16),
3. calling product(2,16) returns product(1,16) + product(1,16),
4. calling product(1,16) returns 16 + product(0,16) + product(0,16) and
5. calling product(0,16) returns 0.

Therefore:

1. product(1,16) returns $16 + 0 + 0 = 16$,
2. product(2,16) returns $16 + 16 = 2 \times 16$,
3. product(5,16) returns $16 + 2 \times 16 + 2 \times 16 = 5 \times 16$ and
4. product(10,16) returns $5 \times 16 + 5 \times 16 = 10 \times 16$. 

10
Binary representation of a

\[
\begin{align*}
a &= a_k 2^k + a_{k-1} 2^{k-1} + \ldots + a_1 2^1 + a_0 2^0 \quad \text{and} \\
\frac{a}{2} &= a_k 2^{k-1} + a_{k-1} 2^{k-2} + \ldots + a_1 2^0 + a_0 2^{-1}
\end{align*}
\]

How many recursions?

1. \( \text{product}(x,y) \) calls

2. \( \text{product}(\left\lfloor \frac{x}{2} \right\rfloor, y) \) which calls

3. \( \text{product}(\left\lfloor \frac{x}{2} \right\rfloor, y) \) which calls

   \[
   \left\lfloor \frac{x}{2} \right\rfloor
   \]

4. \( \text{product}(\left\lfloor \frac{x}{2} \right\rfloor, y) \) which calls

   ... which calls

   i-1. \( \text{product}(1,y) \) which calls

   i. \( \text{product}(0,y) \) which simply returns 0.
Comparison of $x$ and $2\log_2(x) + 2$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$2\log_2(x) + 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>8.3</td>
</tr>
<tr>
<td>10</td>
<td>8.6</td>
</tr>
<tr>
<td>11</td>
<td>8.9</td>
</tr>
<tr>
<td>12</td>
<td>9.2</td>
</tr>
<tr>
<td>13</td>
<td>9.4</td>
</tr>
<tr>
<td>14</td>
<td>9.6</td>
</tr>
<tr>
<td>15</td>
<td>9.8</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>32</td>
<td>12</td>
</tr>
<tr>
<td>64</td>
<td>14</td>
</tr>
<tr>
<td>128</td>
<td>16</td>
</tr>
<tr>
<td>256</td>
<td>18</td>
</tr>
<tr>
<td>512</td>
<td>20</td>
</tr>
<tr>
<td>1024</td>
<td>22</td>
</tr>
<tr>
<td>2048</td>
<td>24</td>
</tr>
<tr>
<td>4096</td>
<td>26</td>
</tr>
<tr>
<td>8192</td>
<td>28</td>
</tr>
<tr>
<td>16384</td>
<td>30</td>
</tr>
<tr>
<td>32768</td>
<td>32</td>
</tr>
</tbody>
</table>
What features of algorithms are studied?

- Design and development methodologies
- Representation of algorithms
- Representation of data
- Hardware design for execution
- Efficiency (time and space)
- Non-algorithmic and infeasible problems
Further reading

In Goodrich and Tamassia: Sections 7.1-7.3 (binary search is in 7.3), and Section 3.1 (notion of running time).