Introduction to CS 5114

T. M. Murali

January 22, 2013

Course Information

- Instructor
  - T. M. Murali, 2160B Torgerson, 231-8534, murali@cs.vt.edu
  - Office Hours: 9:30am–11:30am Thursdays and by appointment
- Teaching assistant
  - Chreston Miller, chmille3@vt.edu
  - Office Hours: to be announced

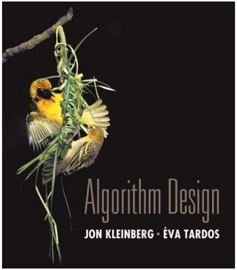
Course Information

- Instructor
  - T. M. Murali, 2160B Torgerson, 231-8534, murali@cs.vt.edu
  - Office Hours: 9:30am–11:30am Thursdays and by appointment
- Teaching assistant
  - Chreston Miller, chmille3@vt.edu
  - Office Hours: to be announced
- Class meeting time
  - TR 2pm–3:15pm, Torgerson 1030, NVC 113

Course Information

- Instructor
  - T. M. Murali, 2160B Torgerson, 231-8534, murali@cs.vt.edu
  - Office Hours: 9:30am–11:30am Thursdays and by appointment
- Teaching assistant
  - Chreston Miller, chmille3@vt.edu
  - Office Hours: to be announced
- Class meeting time
  - TR 2pm–3:15pm, Torgerson 1030, NVC 113
- Keeping in Touch
  - Course web site `http://courses.cs.vt.edu/~cs5114/spring2013`, updated regularly through the semester
  - Scholar web site: grades and homework/exam solutions
  - Scholar mailing list: announcements

Required Course Textbook



- Algorithm Design
- Jon Kleinberg and Éva Tardos
- Addison-Wesley
- 2006
- ISBN: 0-321-29535-8

Course Goals

- Learn methods and principles to construct algorithms.
- Learn techniques to analyze algorithms mathematically for correctness and efficiency (e.g., running time and space used).
- Course roughly follows the topics suggested in textbook
  - Measures of algorithm complexity
  - Greedy algorithms
  - Divide and conquer
  - Dynamic programming
  - Network flow problems
  - NP-completeness
  - Coping with intractability
  - Approximation algorithms
  - Randomized algorithms

Required Readings

- Reading assignment available on the website.
- Read **before** class.

Lecture Slides

- Will be available on class web site.
- Usually posted just before class.
- Class attendance is extremely important.

Lecture Slides

- Will be available on class web site.
- Usually posted just before class.
- Class attendance is extremely important. Lecture in class contains significant and substantial additions to material on the slides.

Homeworks

- Posted on the web site $\approx$ one week before due date.
- Prepare solutions digitally but hand in hard-copy.

Homeworks

- Posted on the web site $\approx$ one week before due date.
- Prepare solutions digitally but hand in hard-copy.
    - Solution preparation recommended in $\LaTeX$.

Examinations

- Take-home midterm.
- Take-home final (comprehensive).
- Prepare digital solutions (recommend LaTeX).

Grades

- Homeworks: ≈ 8, 60% of the grade.
- Take-home midterm: 15% of the grade.
- Take-home final: 25% of the grade.

What is an Algorithm?

What is an Algorithm?

Chamber's A set of prescribed computational procedures for solving a problem; a step-by-step method for solving a problem.

Knuth, TAOCP An algorithm is a finite, definite, effective procedure, with some input and some output.

Origin of the word "Algorithm"

1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja'far Mohammed ben Musa.

Origin of the word “Algorithm”

1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja’far Mohammed ben Musa.

2. From Al Gore, the former U.S. vice-president who invented the internet.

Origin of the word "Algorithm"

1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja'far Mohammed ben Musa.
2. From Al Gore, the former U.S. vice-president who invented the internet.
3. From the Greek *algos* (meaning "pain," also a root of "analgesic") and *rythmos* (meaning "flow," also a root of "rhythm").

Origin of the word "Algorithm"

1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja'far Mohammed ben Musa.
2. From Al Gore, the former U.S. vice-president who invented the internet.
3. From the Greek *algos* (meaning "pain," also a root of "analgesic") and *rythmos* (meaning "flow," also a root of "rhythm"). *"Pain flowed through my body whenever I worked on CS 5114 homeworks."* – former CS 5114 student.

Origin of the word "Algorithm"

1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja'far Mohammed ben Musa. He wrote "Kitab al-jabr wa'l-muqabala," which evolved into today's high school algebra text.
2. From Al Gore, the former U.S. vice-president who invented the internet.
3. From the Greek *algos* (meaning "pain," also a root of "analgesic") and *rythmos* (meaning "flow," also a root of "rhythm"). *"Pain flowed through my body whenever I worked on CS 5114 homeworks."* – former CS 5114 student.

Problem Example

Find Minimum

INSTANCE: Nonempty list $x_1, x_2, \ldots, x_n$ of integers.

SOLUTION: Pair (i, x_i) such that $x_i = \min\{x_j \mid 1 \leq j \leq n\}$.

Algorithm Example

Find-Minimum$(x_1, x_2, \ldots, x_n)$

$i \leftarrow 1$

for $j \leftarrow 2$ **to** n

  do if $x_j < x_i$

    then $i \leftarrow j$

return (i, x_i)

Running Time of Algorithm

Find-Minimum$(x_1, x_2, \ldots, x_n)$

    $i \leftarrow 1$

    for $j \leftarrow 2$ **to** n

        do if $x_j < x_i$

            then $i \leftarrow j$

    return (i, x_i)

Running Time of Algorithm

Find-Minimum$(x_1, x_2, \ldots, x_n)$

```
i ← 1
for j ← 2 to n
    do if x_j < x_i
          then i ← j
return (i, x_i)
```

- At most $2n - 1$ assignments and $n - 1$ comparisons.

Correctness of Algorithm: Proof 1

Find-Minimum($x_1, x_2, \ldots, x_n$)
    $i \leftarrow 1$
    for $j \leftarrow 2$ **to** n
        do if $x_j < x_i$
            then $i \leftarrow j$
    return (i, x_i)

Correctness of Algorithm: Proof 1

Find-Minimum($x_1, x_2, \ldots, x_n$)
    $i \leftarrow 1$
    for $j \leftarrow 2$ **to** n
        do if $x_j < x_i$
            then $i \leftarrow j$
    return (i, x_i)

- Proof by contradiction:

Correctness of Algorithm: Proof 1

Find-Minimum($x_1, x_2, \ldots, x_n$)
    $i \leftarrow 1$
    for $j \leftarrow 2$ **to** n
        do if $x_j < x_i$
            then $i \leftarrow j$
    return (i, x_i)

- Proof by contradiction: Suppose algorithm returns (k, x_k) but there exists $1 \leq l \leq n$ such that $x_l < x_k$ and x_l is the smallest element.

Correctness of Algorithm: Proof 1

Find-Minimum$(x_1, x_2, \ldots, x_n)$

$i \leftarrow 1$
for $j \leftarrow 2$ **to** n
  do if $x_j < x_i$
    then $i \leftarrow j$
return (i, x_i)

- Proof by contradiction: Suppose algorithm returns (k, x_k) but there exists $1 \leq l \leq n$ such that $x_l < x_k$ and x_l is the smallest element.
- Is $k < l$?

Correctness of Algorithm: Proof 1

Find-Minimum($x_1, x_2, \ldots, x_n$)
$i \leftarrow 1$
for $j \leftarrow 2$ **to** n
    do if $x_j < x_i$
        then $i \leftarrow j$
return (i, x_i)

- Proof by contradiction: Suppose algorithm returns (k, x_k) but there exists $1 \leq l \leq n$ such that $x_l < x_k$ and x_l is the smallest element.
- Is $k < l$? No. Since the algorithm returns (k, x_k), $x_k \leq x_j$, for all $k < j \leq n$. Therefore $l < k$.

Correctness of Algorithm: Proof 1

```
Find-Minimum(x_1, x_2, ..., x_n)
    i ← 1
    for j ← 2 to n
        do if x_j < x_i
            then i ← j
    return (i, x_i)
```

- Proof by contradiction: Suppose algorithm returns (k, x_k) but there exists $1 \leq l \leq n$ such that $x_l < x_k$ and x_l is the smallest element.
- Is $k < l$? No. Since the algorithm returns (k, x_k), $x_k \leq x_j$, for all $k < j \leq n$. Therefore $l < k$.
- What does the algorithm do when $j = l$? *It must set i to l*, since we have been told that x_l is the smallest element.
- What does the algorithm do when $j = k$ (which happens after $j = l$)? Since $x_l < x_k$, the value of i does not change.
- Therefore, the algorithm does not return (k, x_k) yielding a contradiction.

Correctness of Algorithm: Proof 2

```
Find-Minimum(x_1, x_2, ..., x_n)
    i ← 1
    for j ← 2 to n
        do if x_j < x_i
            then i ← j
    return (i, x_i)
```

- Proof by induction: What is true at the end of each iteration?

Correctness of Algorithm: Proof 2

```
Find-Minimum(x_1, x_2, ..., x_n)
    i ← 1
    for j ← 2 to n
        do if x_j < x_i
              then i ← j
    return (i, x_i)
```

- Proof by induction: What is true at the end of each iteration?
- Claim: $x_i = \min\{x_m \mid 1 \leq m \leq j\}$, for all $1 \leq j \leq n$.
- Claim is true

Correctness of Algorithm: Proof 2

```
Find-Minimum(x_1, x_2, ..., x_n)
    i ← 1
    for j ← 2 to n
        do if x_j < x_i
                then i ← j
    return (i, x_i)
```

- Proof by induction: What is true at the end of each iteration?
- Claim: $x_i = \min\{x_m \mid 1 \leq m \leq j\}$, for all $1 \leq j \leq n$.
- Claim is true $\Rightarrow$ algorithm is correct (set $j = n$).

Correctness of Algorithm: Proof 2

Find-Minimum($x_1, x_2, \ldots, x_n$)
    $i \leftarrow 1$
    for $j \leftarrow 2$ **to** n
        do if $x_j < x_i$
            then $i \leftarrow j$
    return (i, x_i)

- Proof by induction: What is true at the end of each iteration?
- Claim: $x_i = \min\{x_m \mid 1 \leq m \leq j\}$, for all $1 \leq j \leq n$.
- Claim is true $\Rightarrow$ algorithm is correct (set $j = n$).
- Proof of the claim involves three steps.

1. Base case: $j = 1$ (before loop). $x_i = \min\{x_m \mid 1 \leq m \leq 1\}$ is trivially true.
2. Inductive hypothesis: Assume $x_i = \min\{x_m \mid 1 \leq m \leq j\}$.
3. Inductive step: Prove $x_i = \min\{x_m \mid 1 \leq m \leq j+1\}$.
    - In the loop, i is set to be $j+1$ if and only if $x_{j+1} < x_i$.
    - Therefore, x_i is the smallest of $x_1, x_2, \ldots, x_{j+1}$ after the loop ends.

Format of Proof by Induction

- Goal: prove some proposition $P(n)$ is true for all n.
- Strategy: prove base case, assume inductive hypothesis, prove inductive step.

Format of Proof by Induction

- Goal: prove some proposition $P(n)$ is true for all n.
- Strategy: prove base case, assume inductive hypothesis, prove inductive step.
- Base case: prove that $P(1)$ or $P(2)$ (or $P(\text{small number})$) is true.
- Inductive hypothesis: assume $P(k-1)$ is true.
- Inductive step: prove that $P(k-1) \Rightarrow P(k)$.

Format of Proof by Induction

- Goal: prove some proposition $P(n)$ is true for all n.
- Strategy: prove base case, assume inductive hypothesis, prove inductive step.
- Base case: prove that $P(1)$ or $P(2)$ (or $P(\text{small number})$) is true.
- Inductive hypothesis: assume $P(k-1)$ is true.
- Inductive step: prove that $P(k-1) \Rightarrow P(k)$.
- Why does this strategy work?

Sum of first n natural numbers

$$P(n) = \sum_{i=1}^{n} i =$$

Sum of first n natural numbers

$$P(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Sum of first n natural numbers

$$P(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Proof by Induction:

- Base case:

Sum of first n natural numbers

$$P(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Proof by Induction:

- Base case: $k = 1$: $P(1) = 1 = 1 \times 2/2$.

Sum of first n natural numbers

$$P(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Proof by Induction:

- Base case: $k = 1$: $P(1) = 1 = 1 \times 2/2$.
- Inductive hypothesis:

Sum of first n natural numbers

$$P(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Proof by Induction:

- Base case: $k = 1$: $P(1) = 1 = 1 \times 2/2$.
- Inductive hypothesis: assume $P(k) = k(k+1)/2$.

Sum of first n natural numbers

$$P(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Proof by Induction:

- Base case: $k = 1$: $P(1) = 1 = 1 \times 2/2$.
- Inductive hypothesis: assume $P(k) = k(k+1)/2$.
- Inductive step: Assuming $P(k) = k(k+1)/2$, prove that $P(k+1) = (k+1)(k+2)/2$

Sum of first n natural numbers

$$P(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Proof by Induction:

- Base case: $k = 1$: $P(1) = 1 = 1 \times 2/2$.
- Inductive hypothesis: assume $P(k) = k(k+1)/2$.
- Inductive step: Assuming $P(k) = k(k+1)/2$, prove that $P(k+1) = (k+1)(k+2)/2$

$$P(k+1) \;=\; \sum_{i=1}^{k+1} i =$$

Sum of first n natural numbers

$$P(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Proof by Induction:

- Base case: $k = 1$: $P(1) = 1 = 1 \times 2/2$.
- Inductive hypothesis: assume $P(k) = k(k+1)/2$.
- Inductive step: Assuming $P(k) = k(k+1)/2$, prove that $P(k+1) = (k+1)(k+2)/2$

$$P(k+1) = \sum_{i=1}^{k+1} i = \sum_{i=1}^{k} i + (k+1)$$

Sum of first n natural numbers

$$P(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Proof by Induction:

- Base case: $k = 1$: $P(1) = 1 = 1 \times 2/2$.
- Inductive hypothesis: assume $P(k) = k(k+1)/2$.
- Inductive step: Assuming $P(k) = k(k+1)/2$, prove that $P(k+1) = (k+1)(k+2)/2$

$$P(k+1) = \sum_{i=1}^{k+1} i = \sum_{i=1}^{k} i + (k+1) = \frac{k(k+1)}{2} + (k+1)$$

Sum of first n natural numbers

$$P(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Proof by Induction:

- Base case: $k = 1$: $P(1) = 1 = 1 \times 2/2$.
- Inductive hypothesis: assume $P(k) = k(k+1)/2$.
- Inductive step: Assuming $P(k) = k(k+1)/2$, prove that $P(k+1) = (k+1)(k+2)/2$

$$\begin{aligned} P(k+1) &= \sum_{i=1}^{k+1} i = \sum_{i=1}^{k} i + (k+1) = \frac{k(k+1)}{2} + (k+1) \\ &= (k+1)(\frac{k}{2} + 2) = \frac{(k+1)(k+2)}{2}. \end{aligned}$$

Recurrence Relation

Given

$$P(n) = \begin{cases} P(\lfloor \frac{n}{2} \rfloor) + 1 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

prove that

$$P(n) \leq$$

Recurrence Relation

Given

$$P(n) = \begin{cases} P(\lfloor \frac{n}{2} \rfloor) + 1 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

prove that

$$P(n) \leq 1 + \log_2 n.$$

Recurrence Relation

Given

$$P(n) = \begin{cases} P(\lfloor \frac{n}{2} \rfloor) + 1 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

prove that

$$P(n) \leq 1 + \log_2 n.$$

- Basis: $k = 1$: $P(1) = 1 \leq 1 + \log_2 1$.

Recurrence Relation

Given

$$P(n) = \begin{cases} P(\lfloor \frac{n}{2} \rfloor) + 1 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

prove that

$$P(n) \leq 1 + \log_2 n.$$

- Basis: $k = 1$: $P(1) = 1 \leq 1 + \log_2 1$.
- Inductive hypothesis: Assume $P(k) \leq 1 + \log_2 k$. Prove $P(k+1) \leq 1 + \log_2(k+1)$.

Recurrence Relation

Given

$$P(n) = \begin{cases} P(\lfloor \frac{n}{2} \rfloor) + 1 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

prove that

$$P(n) \leq 1 + \log_2 n.$$

- Basis: $k = 1$: $P(1) = 1 \leq 1 + \log_2 1$.
- Inductive hypothesis: Assume $P(k) \leq 1 + \log_2 k$. Prove $P(k+1) \leq 1 + \log_2(k+1)$.
- Inductive step: $P(k+1) =$

Recurrence Relation

Given

$$P(n) = \begin{cases} P(\lfloor \frac{n}{2} \rfloor) + 1 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

prove that

$$P(n) \leq 1 + \log_2 n.$$

- Basis: $k = 1$: $P(1) = 1 \leq 1 + \log_2 1$.
- Inductive hypothesis: Assume $P(k) \leq 1 + \log_2 k$. Prove $P(k+1) \leq 1 + \log_2(k+1)$.
- Inductive step: $P(k+1) = P(\lfloor \frac{k+1}{2} \rfloor) + 1$.

Recurrence Relation

Given

$$P(n) = \begin{cases} P(\lfloor \frac{n}{2} \rfloor) + 1 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

prove that

$$P(n) \leq 1 + \log_2 n.$$

- Basis: $k = 1$: $P(1) = 1 \leq 1 + \log_2 1$.
- Inductive hypothesis: Assume $P(k) \leq 1 + \log_2 k$. Prove $P(k+1) \leq 1 + \log_2 (k+1)$.
- Inductive step: $P(k+1) = P(\lfloor \frac{k+1}{2} \rfloor) + 1$.
- We are stuck since inductive hypothesis does not say anything about $P(\lfloor \frac{k+1}{2} \rfloor)$.

Strong Induction

- Use strong induction: In the inductive hypothesis, assume that $P(i)$ is true for all $i \leq k$.

$$P(k+1) = P(\lfloor \frac{k+1}{2} \rfloor) + 1$$

Strong Induction

- Use strong induction: In the inductive hypothesis, assume that $P(i)$ is true for all $i \leq k$.

$$\begin{aligned} P(k+1) &= P(\lfloor \frac{k+1}{2} \rfloor) + 1 \\ &\leq 1 + \log_2(\lfloor \frac{k+1}{2} \rfloor) + 1 \\ &\leq 1 + \log_2(k+1) - 1 + 1 = 1 + \log_2(k+1) \end{aligned}$$