CS3230 – Design and Analysis of Algorithms (S1 AY2024/25)

Lecture 3a: Proof of Correctness

Correctness of an algorithm

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.
  - Iterative algorithms.
  - Recursive algorithms.

$\mathbf{Fib}(n)$
- If $n \leq 1$, return n.
- Else, return $\mathbf{Fib}(n-1) + \mathbf{Fib}(n-2)$.

$\mathbf{IFib}(n)$
- If $n \leq 1$
  - return n
- Else,
  - prev2 $=$ 0
  - prev1 $=$ 1
  - for i $= 2$ to n
    - temp $=$ prev1
    - prev1 $=$ prev1$+$prev2
    - prev2 $=$ temp
  - return prev1

While (some condition is met)
- **Do** { some work }

For ($i = 1, 2, \ldots$ up to $i = n$)
- **Do** { some work }

Iterative algorithms

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.

- Analysis of an iterative algorithm:
  - **Loop invariant:**
    - Some desirable conditions that should be satisfied at the start of each iteration.

Iterative algorithms

While (some condition is met)
- **Do** { some work }

For ($i = 1, 2, \ldots$ up to $i = n$)
- **Do** { some work }

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.

- Analysis of an iterative algorithm:
  - **Loop invariant:**
    - Some desirable conditions that should be satisfied at the start of each iteration.
  - **Initialization:**
    - The loop invariant is true at the start of the first iteration.

Iterative algorithms

While (some condition is met)
- **Do** { some work }

For ($i = 1, 2, \ldots$ up to $i = n$)
- **Do** { some work }

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.

- Analysis of an <u>iterative algorithm</u>:
  - **Loop invariant:**
    - Some desirable conditions that should be satisfied at the start of each iteration.
  - **Initialization:**
    - The loop invariant is true at the start of the first iteration.
  - **Maintenance**:
    - If the loop invariant is satisfied at the start of the current iteration, then the loop invariant must be satisfied at <u>the start of the next iteration</u>.

Equivalently, the end of the current iteration.

Iterative algorithms

While (some condition is met)
- **Do** { some work }

For $(i = 1, 2, \ldots \text{up to } i = n)$
- **Do** { some work }

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.

- Analysis of an <u>iterative algorithm</u>:
  - **Loop invariant:**
    - Some desirable conditions that should be satisfied at the start of each iteration.
  - **Initialization:**
    - The loop invariant is true at the start of the first iteration.
  - **Maintenance**:
    - If the loop invariant is satisfied at the start of the current iteration, then the loop invariant must be satisfied at <u>the start of the next iteration</u>.

      Equivalently, the end of the current iteration.
  - **Termination**:
    - Loop invariant at the end of the last iteration → The algorithm outputs a correct answer.

Iterative algorithms

While (some condition is met)
- **Do** { some work }

For $(i = 1, 2, \ldots \text{ up to } i = n)$
- **Do** { some work }

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.

- Analysis of an <u>iterative algorithm</u>:
  - **Loop invariant:** Induction hypothesis
    - Some desirable conditions that should be satisfied at the start of each iteration.
  - **Initialization:** Base case
    - The loop invariant is true at the start of the first iteration.
  - **Maintenance**: Inductive step
    - If the loop invariant is satisfied at the start of the current iteration, then the loop invariant must be satisfied at the start of the next iteration.
  - **Termination**: The proof by induction implies the correctness of the algorithm
    - Loop invariant at the end of the last iteration → The algorithm outputs a correct answer.

Fibonacci numbers

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.

$\mathbf{IFib}(n)$
- If $n \leq 1$
  - return n
- Else,
  - $\text{prev2} = 0$
  - $\text{prev1} = 1$
  - for $i = 2$ to n
    - $\text{temp} = \text{prev1}$
    - $\text{prev1} = \text{prev1} + \text{prev2}$
    - $\text{prev2} = \text{temp}$
  - return prev1

If $n \leq 1$, then the algorithm is correct.
- $\mathbf{Fib}(0) = 0$
- $\mathbf{Fib}(1) = 1$

Fibonacci numbers

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.

$\mathbf{IFib}(n)$
- If $n \leq 1$
  - return n
- Else,
  - prev2 $= 0$
  - prev1 $= 1$
  - for $i = 2$ to n
    - temp $=$ prev1
    - prev1 $=$ prev1$+$prev2
    - prev2 $=$ temp
  - return prev1

Now, consider the case of $n \geq 2$.

Loop invariant:
- At the start of iteration i,
  - prev2 $= \mathbf{Fib}(i-2)$
  - prev1 $= \mathbf{Fib}(i-1)$

Fibonacci numbers

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.

$\mathbf{IFib}(n)$
- If $n \leq 1$
  - return n
- Else,
  - prev2 $= 0$
  - prev1 $= 1$
  - for $i = 2$ to n
    - temp $=$ prev1
    - prev1 $=$ prev1+prev2
    - prev2 $=$ temp
  - return prev1

Now, consider the case of $n \geq 2$.

Loop invariant:
- At the start of iteration i,
  - prev2 $= \mathbf{Fib}(i-2)$
  - prev1 $= \mathbf{Fib}(i-1)$

Initialization:
- At the start of iteration $i = 2$,
  - prev2 $= \mathbf{Fib}(0) = 0$
  - prev1 $= \mathbf{Fib}(1) = 1$

Fibonacci numbers

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.

IFib(n)
- If $n \leq 1$
  - return n
- Else,
  - prev2 $= 0$
  - prev1 $= 1$
  - for $i = 2$ to n
    - temp $=$ prev1 ($\mathbf{Fib}(i-1)$); temp $= \mathbf{Fib}(i-1)$
    - prev1 $=$ prev1+prev2 ($\mathbf{Fib}(i-1) + \mathbf{Fib}(i-2)$); prev1 $= \mathbf{Fib}(i)$
    - prev2 $=$ temp ($\mathbf{Fib}(i-1)$); prev2 $= \mathbf{Fib}(i-1)$
  - return prev1

Now, consider the case of $n \geq 2$.

Loop invariant:
- At the start of iteration i,
  - prev2 $= \mathbf{Fib}(i-2)$
  - prev1 $= \mathbf{Fib}(i-1)$

Initialization:
- At the start of iteration $i = 2$,
  - prev2 $= \mathbf{Fib}(0) = 0$
  - prev1 $= \mathbf{Fib}(1) = 1$

Maintenance:
- Suppose at the start of iteration i,
  - prev2 $= \mathbf{Fib}(i-2)$
  - prev1 $= \mathbf{Fib}(i-1)$
- Then at the end of the iteration,
  - prev2 $= \mathbf{Fib}(i-1)$
  - prev1 $= \mathbf{Fib}(i)$

Fibonacci numbers

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.

$\mathbf{IFib}(n)$
- If $n \leq 1$
  - return n
- Else,
  - prev2 $= 0$
  - prev1 $= 1$
  - for $i = 2$ to n
    - temp $=$ prev1
    - prev1 $=$ prev1+prev2
    - prev2 $=$ temp
  - return prev1

Now, consider the case of $n \geq 2$.

Loop invariant:
- At the start of iteration i,
  - prev2 $= \mathbf{Fib}(i-2)$
  - prev1 $= \mathbf{Fib}(i-1)$

Initialization:
- At the start of iteration $i = 2$,
  - prev2 $= \mathbf{Fib}(0) = 0$
  - prev1 $= \mathbf{Fib}(1) = 1$

Maintenance:
- Suppose at the start of iteration i,
  - prev2 $= \mathbf{Fib}(i-2)$
  - prev1 $= \mathbf{Fib}(i-1)$
- Then at the end of the iteration,
  - prev2 $= \mathbf{Fib}(i-1)$
  - prev1 $= \mathbf{Fib}(i)$

$i = n$

Termination:
- The algorithm returns $\mathbf{Fib}(n)$.

Question 1 @ VisuAlgo online quiz

- What is a suitable loop invariant to analyze this sorting algorithm?

  at the start of iteration j

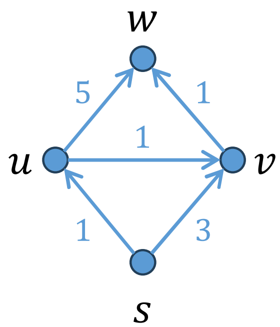
  - A is sorted.
  - $A[1..j-1]$ is sorted.
  - $x \leq y$ for all $x \in A[1..j-1]$ and $y \in A[j..n]$.
  - ($A[1..j-1]$ is sorted) and ($x \leq y$ for all $x \in A[1..j-1]$ and $y \in A[j..n]$).

SelectionSort$(A[1..n])$
- For $j = 1$ to $n - 1$
  - Select $s \in \{j, j+1, \ldots, n\}$ so that $A[s]$ is a smallest number in $A[j..n]$.
  - Swap $A[j]$ and $A[s]$.

VisuAlgo (Selection sort): https://visualgo.net/en/sorting?mode=Selection

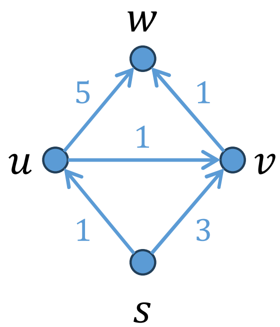
Weighted directed graphs

- Let $G = (V, E)$ be a directed graph.
- Each edge $e \in E$ has a positive weight $w(e)$.
  - If $(u, v) \in E$, then $w(u, v) = w(e)$.
  - If $(u, v) \notin E$, then $w(u, v) = \infty$.



Single-source shortest paths

- Let $G = (V, E)$ be a directed graph.
- Each edge $e \in E$ has a positive weight $w(e)$.
  - If $(u, v) \in E$, then $w(u, v) = w(e)$.
  - If $(u, v) \notin E$, then $w(u, v) = \infty$.
- **Goal:** Given a source $s \in V$, compute the shortest-path distance from s to all vertices.

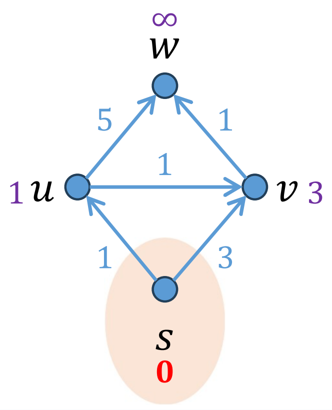


$\mathbf{dist}(s, s) = 0$
$\mathbf{dist}(s, u) = 1$
$\mathbf{dist}(s, v) = 2$
$\mathbf{dist}(s, w) = 3$

Dijkstra's algorithm

Dijkstra$(G = (V, E), s \in V)$

- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\min_{u \in R}\big(d(u) + w(u, v)\big)$
  - $d(v) = \min_{u \in R}\big(d(u) + w(u, v)\big)$
  - Add v to R

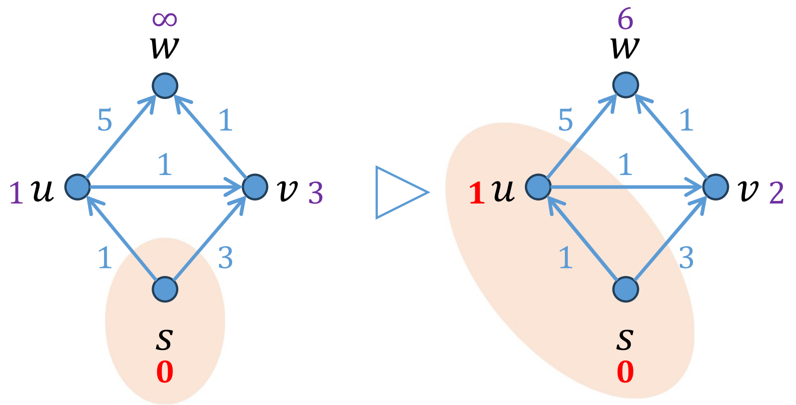


$\mathbf{dist}(s, s) = 0$
$\mathbf{dist}(s, u) = 1$
$\mathbf{dist}(s, v) = 2$
$\mathbf{dist}(s, w) = 3$

VisuAlgo (Dijkstra): https://visualgo.net/en/sssp?slide=7

Dijkstra's algorithm

Dijkstra$(G = (V, E), s \in V)$
- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\min_{u \in R} (d(u) + w(u, v))$
  - $d(v) = \min_{u \in R} (d(u) + w(u, v))$
  - Add v to R



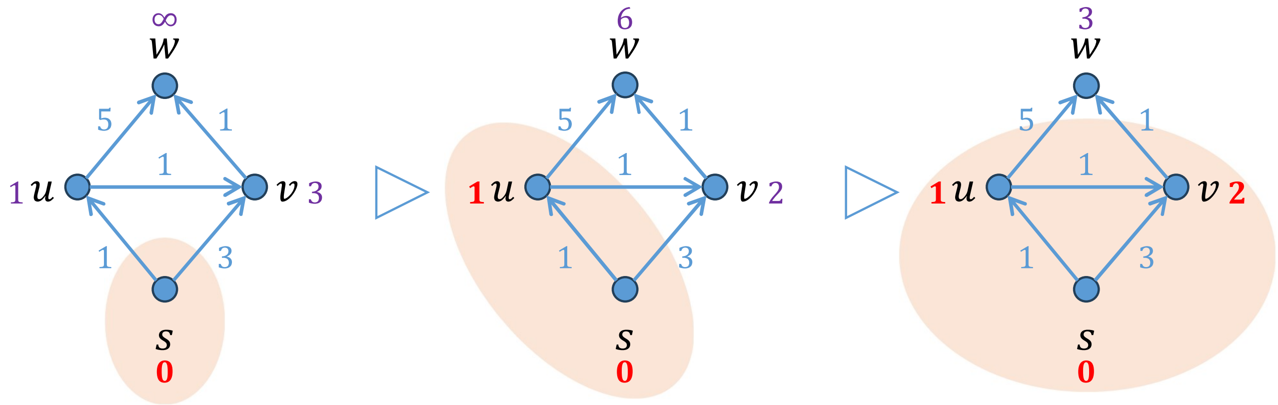
$\textbf{dist}(s, s) = 0$
$\textbf{dist}(s, u) = 1$
$\textbf{dist}(s, v) = 2$
$\textbf{dist}(s, w) = 3$

VisuAlgo (Dijkstra): https://visualgo.net/en/sssp?slide=7

Dijkstra's algorithm

Dijkstra$(G = (V, E), s \in V)$

- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\min_{u \in R}\big(d(u) + w(u, v)\big)$
  - $d(v) = \min_{u \in R}\big(d(u) + w(u, v)\big)$
  - Add v to R



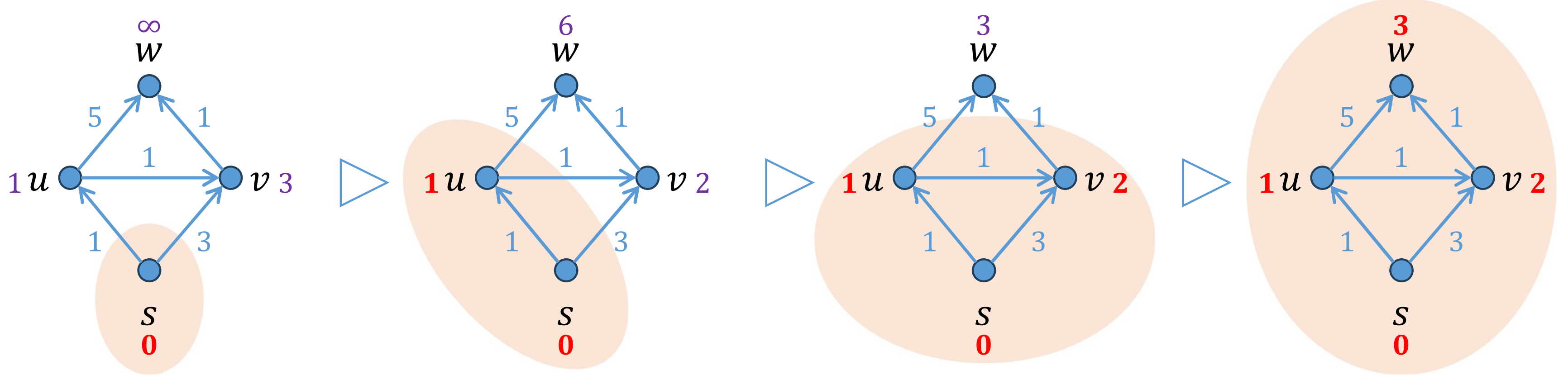
$\mathbf{dist}(s, s) = 0$
$\mathbf{dist}(s, u) = 1$
$\mathbf{dist}(s, v) = 2$
$\mathbf{dist}(s, w) = 3$

VisuAlgo (Dijkstra): https://visualgo.net/en/sssp?slide=7

Dijkstra's algorithm

Dijkstra$(G = (V, E), s \in V)$

- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\min_{u \in R} (d(u) + w(u, v))$
  - $d(v) = \min_{u \in R} (d(u) + w(u, v))$
  - Add v to R



$\mathbf{dist}(s, s) = 0$
$\mathbf{dist}(s, u) = 1$
$\mathbf{dist}(s, v) = 2$
$\mathbf{dist}(s, w) = 3$

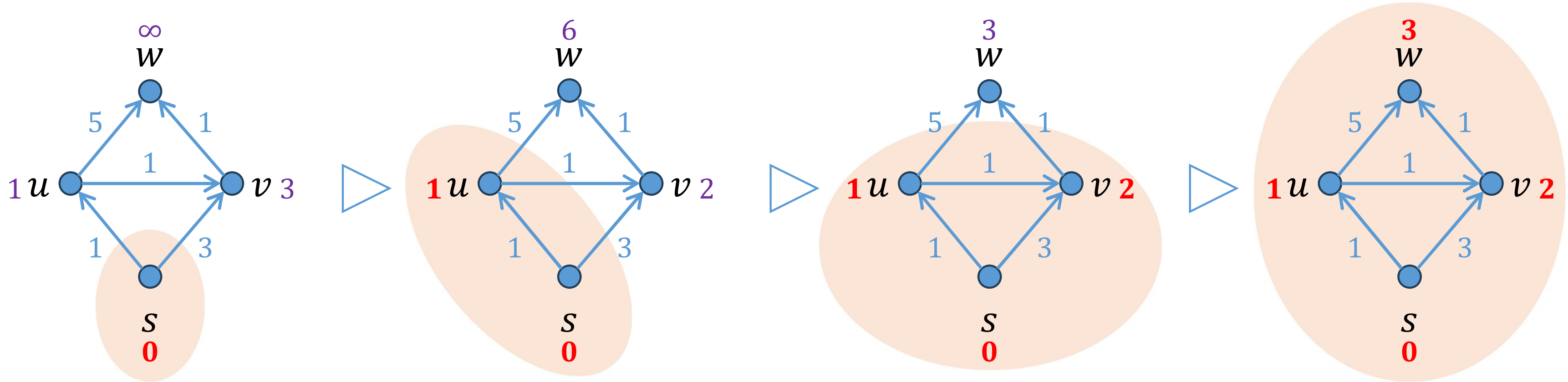
VisuAlgo (Dijkstra): https://visualgo.net/en/sssp?slide=7

Proof of correctness

Loop invariant:
- At the start of an iteration:
  - $\forall u \in R,\ d(u) = \mathbf{dist}(s,u)$

The computed distances are correct.

Dijkstra$(G = (V,E), s \in V)$
- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\mathbf{min}_{u \in R}\big(d(u) + w(u,v)\big)$
  - $d(v) = \mathbf{min}_{u \in R}\big(d(u) + w(u,v)\big)$
  - Add v to R



$\mathbf{dist}(s,s) = 0$
$\mathbf{dist}(s,u) = 1$
$\mathbf{dist}(s,v) = 2$
$\mathbf{dist}(s,w) = 3$

VisuAlgo (Dijkstra): https://visualgo.net/en/sssp?slide=7

Proof of correctness

Loop invariant:

- At the start of an iteration:
  - $\forall u \in R,\ d(u) = \mathbf{dist}(s, u)$

The computed distances are correct.

$d(s) = 0$ is correct.

$\mathbf{Dijkstra}(G = (V, E), s \in V)$

- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\min_{u \in R}\big(d(u) + w(u, v)\big)$
  - $d(v) = \min_{u \in R}\big(d(u) + w(u, v)\big)$
  - Add v to R

✓ **Initialization**

Maintenance

Termination

Loop invariant:

- At the start of an iteration:
  - $\forall u \in R,\ d(u) = \mathbf{dist}(s,u)$

The computed distances are correct.

Proof of correctness

Dijkstra$(G = (V,E), s \in V)$

- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\min_{u \in R}\big(d(u) + w(u,v)\big)$
  - $d(v) = \min_{u \in R}\big(d(u) + w(u,v)\big)$
  - Add v to R

✓ Initialization

Maintenance

✓ **Termination**

At the end of the computation, $R = V$

Proof of correctness

Loop invariant:

- At the start of an iteration:
  - $\forall u \in R,\ d(u) = \mathbf{dist}(s,u)$

The computed distances are correct.

$\mathbf{Dijkstra}(G = (V,E), s \in V)$

- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\min_{u \in R}\big(d(u) + w(u,v)\big)$
  - $d(v) = \min_{u \in R}\big(d(u) + w(u,v)\big)$
  - Add v to R

Claim: For the selected vertex v:

- $\mathbf{dist}(s,v) = \min_{u \in R}\big(d(u) + w(u,v)\big)$

✓ Initialization

Maintenance

✓ Termination

Loop invariant:

- At the start of an iteration:
  - $\forall u \in R,\ d(u) = \mathbf{dist}(s,u)$

The computed distances are correct.

Proof of correctness

$\mathbf{Dijkstra}(G = (V, E), s \in V)$

- $d(s) = 0$
- $R = \{s\}$
- $\mathbf{While}\ R \neq V$
  - Select $v \in V \setminus R$ to minimize $\min_{u \in R}\left(d(u) + w(u,v)\right)$
  - $d(v) = \min_{u \in R}\left(d(u) + w(u,v)\right)$
  - Add v to R

Claim: For the selected vertex v:

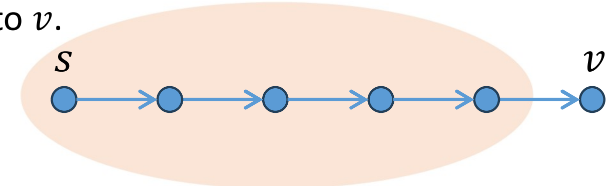
- $\mathbf{dist}(s,v) = \min_{u \in R}\left(d(u) + w(u,v)\right)$

✓ Initialization

Maintenance

✓ Termination

Proof: Consider a shortest path P from s to v.



All vertices in $P \setminus \{v\}$ must be in R.

- Otherwise, we should have selected the first vertex that is not in R.

Loop invariant:
- At the start of an iteration:
  - $\forall u \in R,\ d(u) = \mathbf{dist}(s,u)$

The computed distances are correct.

Proof of correctness

$\mathbf{Dijkstra}(G = (V,E), s \in V)$
- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\min_{u \in R}\big(d(u) + w(u,v)\big)$
  - $d(v) = \min_{u \in R}\big(d(u) + w(u,v)\big)$
  - Add v to R

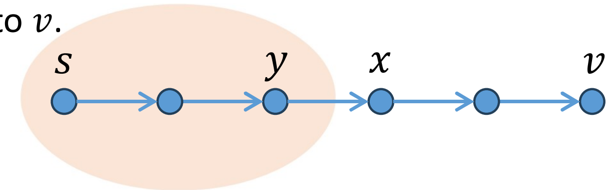
Claim: For the selected vertex v:
- $\mathbf{dist}(s,v) = \min_{u \in R}\big(d(u) + w(u,v)\big)$

✓ Initialization

Maintenance

✓ Termination

Proof: Consider a shortest path P from s to v.



All vertices in $P \setminus \{v\}$ must be in R.
- Otherwise, we should have selected the first vertex that is not in R.
  x

$$\min_{u \in R}\big(d(u) + w(u,v)\big) \geq \mathbf{dist}(s,v) > \mathbf{dist}(s,x) = d(y) + w(y,x) \geq \min_{u \in R}\big(d(u) + w(u,x)\big)$$

$\mathbf{dist}(s,u)$ $\mathbf{dist}(s,y)$

Loop invariant:

- At the start of an iteration:
  - $\forall u \in R,\ d(u) = \mathbf{dist}(s,u)$

The computed distances are correct.

Proof of correctness

$\mathbf{Dijkstra}(G = (V,E), s \in V)$

- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\mathbf{min}_{u \in R}\big(d(u) + w(u,v)\big)$
  - $d(v) = \mathbf{min}_{u \in R}\big(d(u) + w(u,v)\big)$
  - Add v to R

Claim: For the selected vertex v:

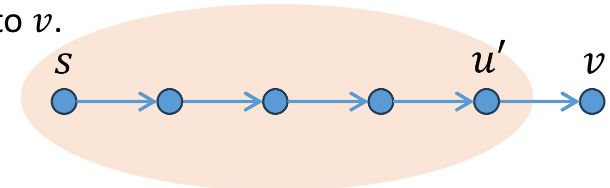
✓ $\mathbf{dist}(s,v) = \mathbf{min}_{u \in R}\big(d(u) + w(u,v)\big)$

✓ Initialization

Maintenance

✓ Termination

Proof: Consider a shortest path P from s to v.



All vertices in $P \setminus \{v\}$ must be in R.

$$\mathbf{min}_{u \in R}\big(\underbrace{d(u)}_{\mathbf{dist}(s,u)} + w(u,v)\big) \geq \mathbf{dist}(s,v) = \underbrace{d(u')}_{\mathbf{dist}(s,u')} + w(u',v) \geq \mathbf{min}_{u \in R}\big(d(u) + w(u,v)\big)$$

Proof of correctness

Loop invariant:
- At the start of an iteration:
  - $\forall u \in R,\ d(u) = \mathbf{dist}(s,u)$

The computed distances are correct.

$\mathbf{Dijkstra}(G = (V,E), s \in V)$
- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\min_{u \in R}\big(d(u) + w(u,v)\big)$
  - $d(v) = \min_{u \in R}\big(d(u) + w(u,v)\big)$
  - Add v to R

Claim: For the selected vertex v:
✓ $\mathbf{dist}(s,v) = \min_{u \in R}\big(d(u) + w(u,v)\big)$

✓ Initialization

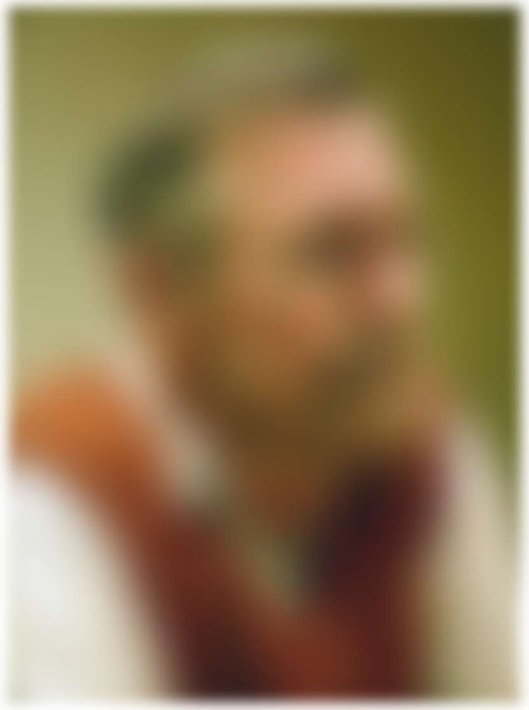
✓ **Maintenance**

✓ Termination

Question 2 @ VisuAlgo online quiz

Who is the **Master of Algorithms** pictured below?

- Stephen Cook
- Edsger Dijkstra
- Robert Tarjan
- Avi Wigderson



Efficiency

Dijkstra$(G = (V, E), s \in V)$

- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\mathbf{min}_{u \in R} \big(d(u) + w(u, v)\big)$
  - $d(v) = \mathbf{min}_{u \in R} \big(d(u) + w(u, v)\big)$
  - Add v to R

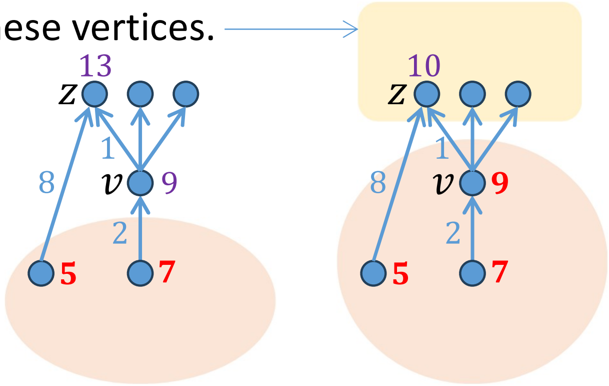
Efficiency

Dijkstra$(G = (V, E), s \in V)$
- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\min_{u \in R}\big(d(u) + w(u, v)\big)$
  - $d(v) = \min_{u \in R}\big(d(u) + w(u, v)\big)$
  - Add v to R

Observation: No need to compute $x(v) = \min_{u \in R}\big(d(u) + w(u, v)\big)$ from scratch for every iteration.

Adding v to R can <u>only</u> affect the x-value of these vertices.
- $x(z) \leftarrow \min\{x(z), d(v) + w(v, z)\}$

10 13 9 1



Efficiency

Dijkstra$(G = (V, E), s \in V)$
- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\min_{u \in R} \big(d(u) + w(u, v)\big)$
  - $d(v) = \min_{u \in R} \big(d(u) + w(u, v)\big)$
  - Add v to R

Observation: No need to compute $x(v) = \min_{u \in R} \big(d(u) + w(u, v)\big)$ from scratch for every iteration.

Dijkstra$(G = (V, E), s \in V)$
- $d(s) = 0$
- $x(v) = \infty$ for all $v \in V \setminus \{s\}$
- $R = \{s\}$
- For all $z \in V$ such that $(s, z) \in E$
  - $x(z) = \mathbf{min}\{x(z), d(s) + w(s, z)\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $x(v)$
  - $d(v) = x(v)$
  - Add v to R
  - For all $z \in V \setminus R$ such that $(v, z) \in E$
    - $x(z) = \mathbf{min}\{x(z), d(v) + w(v, z)\}$

Efficiency

Dijkstra$(G = (V, E), s \in V)$

- $d(s) = 0$
- $R = \{s\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $\min_{u \in R}\big(d(u) + w(u, v)\big)$
  - $d(v) = \min_{u \in R}\big(d(u) + w(u, v)\big)$
  - Add v to R

Observation: No need to compute $x(v) = \min_{u \in R}\big(d(u) + w(u, v)\big)$ from scratch for every iteration.

Dijkstra$(G = (V, E), s \in V)$

- $d(s) = 0$
- $x(v) = \infty$ for all $v \in V \setminus \{s\}$
- $R = \{s\}$
- For all $z \in V$ such that $(s, z) \in E$
  - $x(z) = \mathbf{min}\{x(z), d(s) + w(s, z)\}$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $x(v)$
  - $d(v) = x(v)$
  - Add v to R
  - For all $z \in V \setminus R$ such that $(v, z) \in E$
    - $x(z) = \mathbf{min}\{x(z), d(v) + w(v, z)\}$

▷

Dijkstra$(G = (V, E), s \in V)$

- $R = \emptyset$
- $x(v) = \infty$ for all $v \in V \setminus \{s\}$
- $x(s) = 0$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $x(v)$
  - $d(v) = x(v)$
  - Add v to R
  - For all $z \in V \setminus R$ such that $(v, z) \in E$
    - $x(z) = \mathbf{min}\{x(z), d(v) + w(v, z)\}$

Efficiency

- The algorithm can be implemented using a priority queue.

$|V|$ **insert**

$|V|$ **extract-min**

At most $|E|$ **decrease-key**

Dijkstra$(G = (V, E), s \in V)$

- $R = \emptyset$
- $x(v) = \infty$ for all $v \in V \setminus \{s\}$
- $x(s) = 0$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $x(v)$
  - $d(v) = x(v)$
  - Add v to R
  - For all $z \in V \setminus R$ such that $(v, z) \in E$
    - $x(z) = \mathbf{min}\{x(z), d(v) + w(v, z)\}$

Efficiency

- The algorithm can be implemented using a priority queue.

	Insert	Extract-min	Decrease-key	Time complexity of Dijkstra's algorithm
Binary heap	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O\big((\lvert E\rvert + \lvert V\rvert)\log\lvert V\rvert\big)$
Fibonacci heap	$O(1)$	$O(\log n)$ amortized	$O(1)$ amortized	$O(\lvert E\rvert + \lvert V\rvert\log\lvert V\rvert)$

https://en.wikipedia.org/wiki/Priority_queue

Amortized = average over n operations.

Dijkstra$(G = (V, E), s \in V)$

- $R = \emptyset$
- $x(v) = \infty$ for all $v \in V \setminus \{s\}$ — $|V|$ **insert**
- $x(s) = 0$
- **While** $R \neq V$
  - Select $v \in V \setminus R$ to minimize $x(v)$ — $|V|$ **extract-min**
  - $d(v) = x(v)$
  - Add v to R
  - For all $z \in V \setminus R$ such that $(v, z) \in E$
    - $x(z) = \mathbf{min}\{x(z), d(v) + w(v, z)\}$ — At most $|E|$ **decrease-key**

Recursive algorithms

Fib(n)
- If $n \leq 1$, return n.
- Else, return $\mathbf{Fib}(n-1) + \mathbf{Fib}(n-2)$.

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.

- Analysis of a <u>recursive algorithm</u>:
  - **Base case:** without any recursive calls
    - Show that the algorithm is correct for the base case.
  - **Inductive step:** with recursive calls
    - Assume that the algorithm is correct for any input of size smaller than n.
    - Show that the algorithm is correct for any input of size n.

Recursive algorithms

$\mathbf{Fib}(n)$
- If $n \leq 1$, return n.
- Else, return $\mathbf{Fib}(n-1) + \mathbf{Fib}(n-2)$.

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.

- Analysis of a recursive algorithm:
  - **Base case:**
    - Show that the algorithm is correct for the base case.
  - **Inductive step:**
    - Assume that the algorithm is correct for any input of size smaller than n.
    - Show that the algorithm is correct for any input of size n.

If $n \leq 1$, the algorithm is correct:
- $\mathbf{Fib}(0) = 0$
- $\mathbf{Fib}(1) = 1$

Recursive algorithms

Fib(n)
- If $n \leq 1$, return n.
- Else, return $\mathbf{Fib}(n-1) + \mathbf{Fib}(n-2)$.

- **Goal:** For a given algorithm $\mathcal{A}$, prove that it is correct.

- Analysis of a recursive algorithm:
  - **Base case:**
    - Show that the algorithm is correct for the base case.
  - **Inductive step:**
    - Assume that the algorithm is correct for any input of size smaller than n.
    - Show that the algorithm is correct for any input of size n.

If $n \leq 1$, the algorithm is correct:
- $\mathbf{Fib}(0) = 0$
- $\mathbf{Fib}(1) = 1$

If $n > 1$, the algorithm is correct:
- $\mathbf{Fib}(n) = \mathbf{Fib}(n-1) + \mathbf{Fib}(n-2)$

Searching in a sorted array

- **Input:**
  - A <u>sorted</u> array A.
  - Two indices:
    - lb (lower bound)
    - ub (upper bound)
  - A number x.

- **Goal:**
  - Decide if $x \in A[\mathrm{lb}..\mathrm{ub}]$

If $(\mathrm{lb} > \mathrm{ub})$, the answer is **NO**.

Searching in a sorted array

- **Input:**
  - A sorted array A.
  - Two indices:
    - lb (lower bound)
    - ub (upper bound)
  - A number x.

BinarySearch$(A, \mathrm{lb}, \mathrm{ub}, x)$
- **If** $(\mathrm{lb} > \mathrm{ub})$, return **NO**.
- **Else**
  - $\mathrm{mid} \leftarrow \left\lfloor \frac{\mathrm{lb}+\mathrm{ub}}{2} \right\rfloor$.
  - **If** $(x = A[\mathrm{mid}])$, return **YES**.
  - **If** $(x > A[\mathrm{mid}])$, **BinarySearch**$(A, \mathrm{mid}+1, \mathrm{ub}, x)$.
  - **If** $(x < A[\mathrm{mid}])$, **BinarySearch**$(A, \mathrm{lb}, \mathrm{mid}-1, x)$.

- **Goal:**
  - Decide if $x \in A[\mathrm{lb}..\mathrm{ub}]$

Searching in a sorted array

$x = 14$

$\text{mid} = 5$

$[2 \quad 7 \quad 14 \quad 33 \quad 41 \quad 50 \quad 77 \quad 80 \quad 82]$

$\text{lb} = 1$ $\qquad$ $\text{ub} = 9$

$\text{mid} = 2$

$[2 \quad 7 \quad 14 \quad 33 \quad 41 \quad 50 \quad 77 \quad 80 \quad 82]$

$\text{lb} = 1$ $\qquad$ $\text{ub} = 4$

$\text{mid} = 3$

$[2 \quad 7 \quad 14 \quad 33 \quad 41 \quad 50 \quad 77 \quad 80 \quad 82]$

$\text{lb} = 3$ $\qquad$ $\text{ub} = 4$

YES

BinarySearch$(A, \text{lb}, \text{ub}, x)$

- **If** $(\text{lb} > \text{ub})$, return **NO**.
- **Else**
  - $\text{mid} \leftarrow \left\lfloor \frac{\text{lb}+\text{ub}}{2} \right\rfloor$.
  - **If** $(x = A[\text{mid}])$, return **YES**.
  - **If** $(x > A[\text{mid}])$, **BinarySearch**$(A, \text{mid} + 1, \text{ub}, x)$.
  - **If** $(x < A[\text{mid}])$, **BinarySearch**$(A, \text{lb}, \text{mid} - 1, x)$.

Proof of correctness

- **Induction** on the array size:
  - $n = \text{ub} - \text{lb} + 1$

BinarySearch$(A, \text{lb}, \text{ub}, x)$
- **If** $(\text{lb} > \text{ub})$, return **NO**.
- **Else**
  - $\text{mid} \leftarrow \left\lfloor \frac{\text{lb}+\text{ub}}{2} \right\rfloor$.
  - **If** $(x = A[\text{mid}])$, return **YES**.
  - **If** $(x > A[\text{mid}])$, **BinarySearch**$(A, \text{mid}+1, \text{ub}, x)$.
  - **If** $(x < A[\text{mid}])$, **BinarySearch**$(A, \text{lb}, \text{mid}-1, x)$.

Proof of correctness

- **Induction** on the array size:
  - $n = \mathrm{ub} - \mathrm{lb} + 1$
- **Base case:**
  - If $n < 1$, the algorithm correctly returns **NO**.

BinarySearch$(A, \mathrm{lb}, \mathrm{ub}, x)$
- **If** $(\mathrm{lb} > \mathrm{ub})$, return **NO**.
- **Else**
  - $\mathrm{mid} \leftarrow \left\lfloor \frac{\mathrm{lb}+\mathrm{ub}}{2} \right\rfloor$.
  - **If** $(x = A[\mathrm{mid}])$, return **YES**.
  - **If** $(x > A[\mathrm{mid}])$, **BinarySearch**$(A, \mathrm{mid} + 1, \mathrm{ub}, x)$.
  - **If** $(x < A[\mathrm{mid}])$, **BinarySearch**$(A, \mathrm{lb}, \mathrm{mid} - 1, x)$.

Proof of correctness

- **Induction** on the array size:
  - $n = \text{ub} - \text{lb} + 1$

- **Inductive step:**
  - Assume the algorithm works correctly for any input of size smaller than n.

$\mathbf{BinarySearch}(A, \text{lb}, \text{ub}, x)$
- **If** $(\text{lb} > \text{ub})$, return **NO**.
- **Else**
  - $\text{mid} \leftarrow \left\lfloor \frac{\text{lb}+\text{ub}}{2} \right\rfloor$.
  - **If** $(x = A[\text{mid}])$, return **YES**.
  - **If** $(x > A[\text{mid}])$, $\mathbf{BinarySearch}(A, \text{mid} + 1, \text{ub}, x)$.
  - **If** $(x < A[\text{mid}])$, $\mathbf{BinarySearch}(A, \text{lb}, \text{mid} - 1, x)$.

If $(x = A[\text{mid}])$, the answer must be **YES**.

If $(x > A[\text{mid}])$, then:
- $(x \in A[\text{lb}..\text{ub}])$ if and only if $(x \in A[\text{mid} + 1..\text{ub}])$. A is sorted
- Therefore, the answer must be $\mathbf{BinarySearch}(A, \text{mid} + 1, \text{ub}, x)$. Induction hypothesis

The case of $(x < A[\text{mid}])$ is similar.

Efficiency

- Input size:
  - $n = \mathrm{ub} - \mathrm{lb} + 1$
- Subproblem input size:
  - $\leq \lfloor n/2 \rfloor$
- Depth of recursion:
  - $O(\log n)$
- Time complexity:
  - $T(n) \in O(\log n)$

$\mathbf{BinarySearch}(A, \mathrm{lb}, \mathrm{ub}, x)$
- **If** $(\mathrm{lb} > \mathrm{ub})$, return **NO**.
- **Else**
  - $\mathrm{mid} \leftarrow \left\lfloor \frac{\mathrm{lb}+\mathrm{ub}}{2} \right\rfloor$.
  - **If** $(x = A[\mathrm{mid}])$, return **YES**.
  - **If** $(x > A[\mathrm{mid}])$, $\mathbf{BinarySearch}(A, \mathrm{mid}+1, \mathrm{ub}, x)$.
  - **If** $(x < A[\mathrm{mid}])$, $\mathbf{BinarySearch}(A, \mathrm{lb}, \mathrm{mid}-1, x)$.

Acknowledgement

- The slides are modified from previous editions of this course and similar course elsewhere.
- **List of credits:**
  - Diptarka Chakraborty
  - Yi-Jun Chang
  - Erik Demaine
  - Steven Halim
  - Sanjay Jain
  - Wee Sun Lee
  - Charles Leiserson
  - Hon Wai Leong
  - Wing-Kin Sung