Linear Programming Continued!

Algorithms..

Designed..

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

Make a graph.

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

  Make a graph.
  Run algorithm.

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

  Make a graph.
  Run algorithm.
  Pull out answer.

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

Make a graph.
Run algorithm.
Pull out answer.

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

Make a graph.
Run algorithm.
Pull out answer.

Examples:

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

> Make a graph.
> Run algorithm.
> Pull out answer.

Examples:
Shortest path.

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

Make a graph.
Run algorithm.
Pull out answer.

Examples:
Shortest path.
Dynamic programming and DAG!

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

Make a graph.
Run algorithm.
Pull out answer.

Examples:
Shortest path.
Dynamic programming and DAG!

Peas and carrots.

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

Make a graph.
Run algorithm.
Pull out answer.

Examples:
Shortest path.
Dynamic programming and DAG!

Peas and carrots. Express as a linear program!

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

> Make a graph.
> Run algorithm.
> Pull out answer.

Examples:
Shortest path.
Dynamic programming and DAG!

Peas and carrots. Express as a linear program!
Production.

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

Make a graph.
Run algorithm.
Pull out answer.

Examples:
Shortest path.
Dynamic programming and DAG!

Peas and carrots. Express as a linear program!
Production. Express as a linear program!

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

> Make a graph.
> Run algorithm.
> Pull out answer.

Examples:
Shortest path.
Dynamic programming and DAG!

Peas and carrots. Express as a linear program!
Production. Express as a linear program!

Different from subroutine call.

Algorithms..

Designed..

..multiplication, sorts, depth first search, breadth first search, greedy, dynamic programs.

Used previous algorithms as subroutines.
E.g. sort for greedy mst.

Another method.

> Make a graph.
> Run algorithm.
> Pull out answer.

Examples:
Shortest path.
Dynamic programming and DAG!

Peas and carrots. Express as a linear program!
Production. Express as a linear program!

Different from subroutine call.

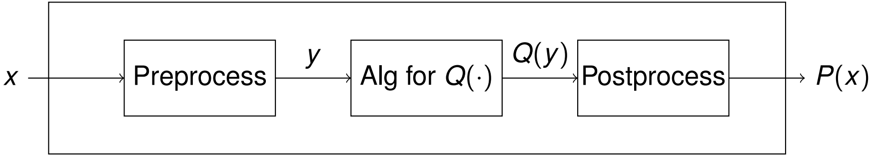
Pre/post process is “easy”.

Reduction.

Algorithm for P.

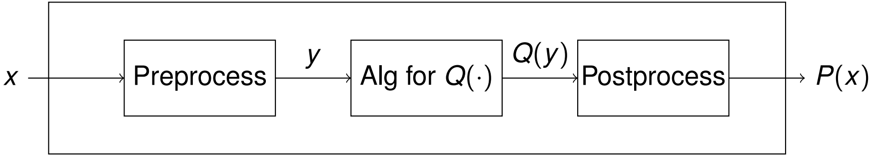
Reduction.

Algorithm for P. ...from algorithm for Q.



Reduction.

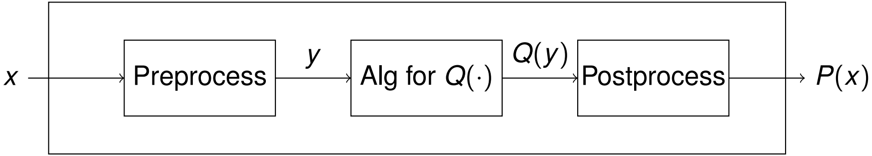
Algorithm for P. ...from algorithm for Q.



Peas and Carrots, Production. Instances.

Reduction.

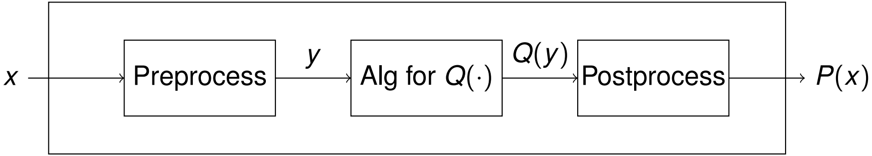
Algorithm for P. ...from algorithm for Q.



Peas and Carrots, Production. Instances.
General problem reduction to linear program.

Reduction.

Algorithm for P. ...from algorithm for Q.



Peas and Carrots, Production. Instances.
General problem reduction to linear program.

New problem: Maximum Flow.

Max-Flow Problem.

Max-Flow Problem.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.

Max-Flow Problem.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.

Max-Flow Problem.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.
2. Conservation Constraints:

Max-Flow Problem.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.
2. Conservation Constraints:
"flow into v" = "flow out of v" (if not s or t.)

Max-Flow Problem.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.

2. Conservation Constraints:
"flow into v" = "flow out of v" (if not s or t.)
Algorithm adds flow, say f, to path from s to t.

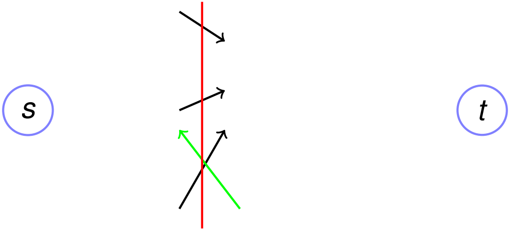
Max-Flow Problem.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.

2. Conservation Constraints:
"flow into v" = "flow out of v" (if not s or t.)
Algorithm adds flow, say f, to path from s to t.

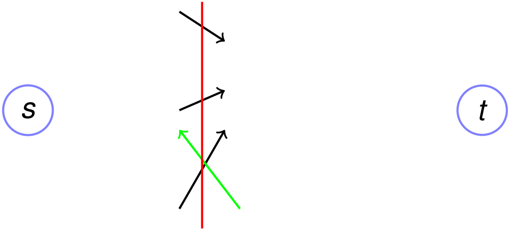
Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Optimality: upper bound.

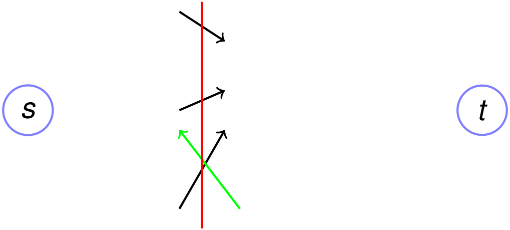
s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

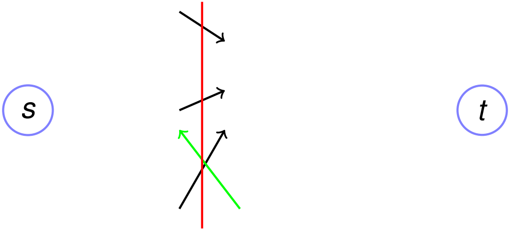


Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T

Optimality: upper bound.

s-*t* Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

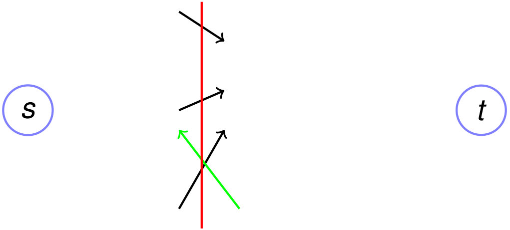


Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



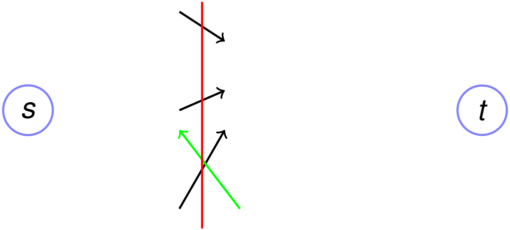
Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



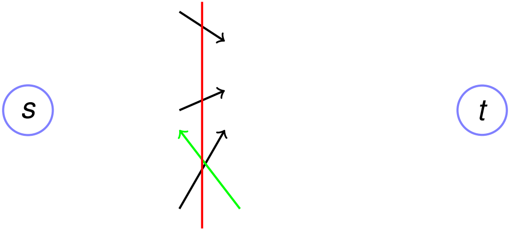
Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



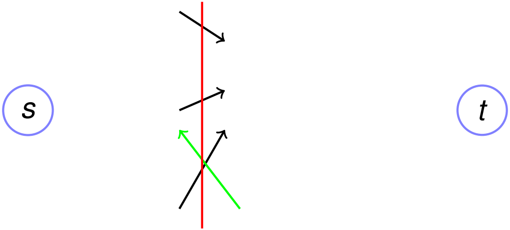
Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



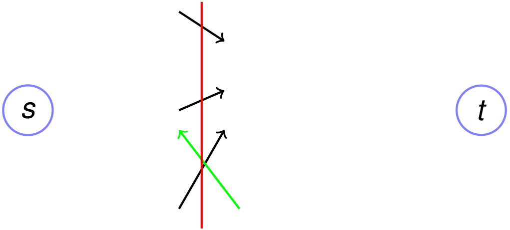
Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

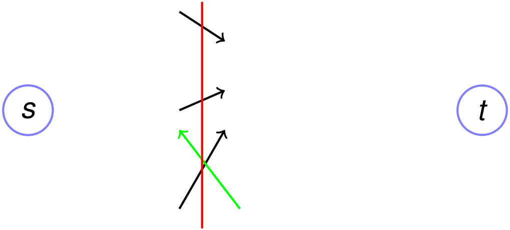
$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \to Z+$, the flow out of S (into T)

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

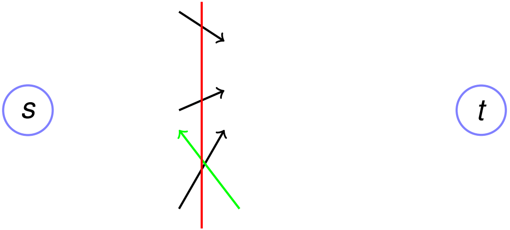
$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \rightarrow Z+$, the flow out of S (into T)

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

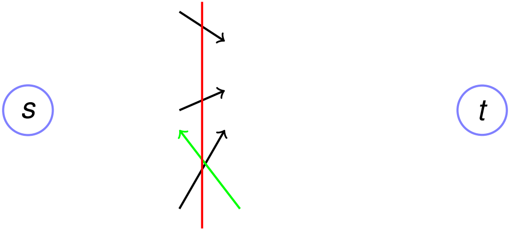
$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \rightarrow Z+$, the flow out of S (into T)
$\sum_{e \in S \times T} f_e$

Optimality: upper bound.

s-*t* Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

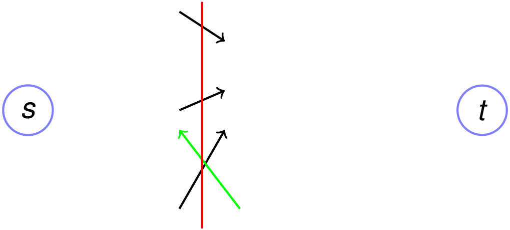
$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \rightarrow Z+$, the flow out of S (into T)
$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e$

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

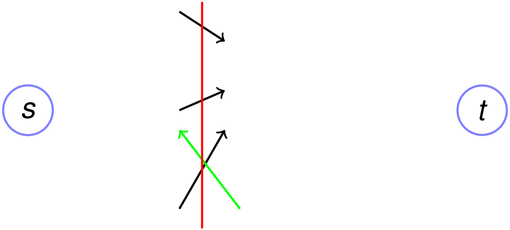
$C(S, T)$ - sum of capacities of all arcs from S to T
$C(S, T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \to Z+$, the flow out of S (into T)
$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \leq \sum_{e \in S \times T} c_e - \sum_{e \in T \times S} 0$

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

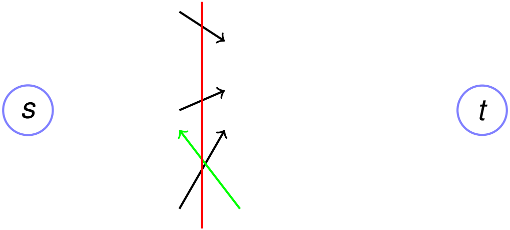
$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \to Z+$, the flow out of S (into T)
$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \leq \sum_{e \in S \times T} c_e - \sum_{e \in T \times S} 0 = C(S,T)$.

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

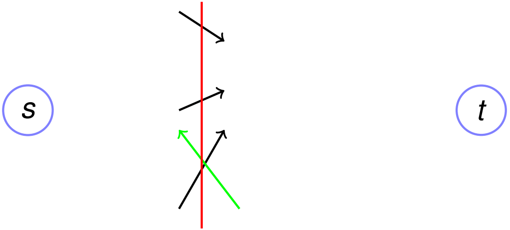
For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \rightarrow Z+$, the flow out of S (into T)
$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \leq \sum_{e \in S \times T} c_e - \sum_{e \in T \times S} 0 = C(S,T).$

$\rightarrow$ The value of any valid flow is at most $C(S,T)$!

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \to Z+$, the flow out of S (into T)
$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \leq \sum_{e \in S \times T} c_e - \sum_{e \in T \times S} 0 = C(S,T)$.

$\to$ The value of any valid flow is at most $C(S,T)$! $\square$

Optimality: max flow = min cut.

At termination of augmenting path algorithm.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.

No path with residual capacity!

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
  No path with residual capacity!

Depth first search only starting at s does not reach t.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
    No path with residual capacity!

Depth first search only starting at s does not reach t.

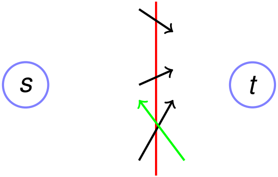
S be reachable nodes.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.

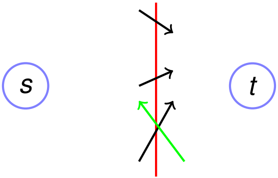
S be reachable nodes.



Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

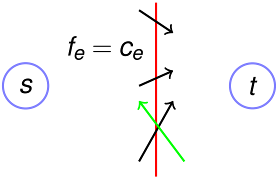
No arc with positive residual capacity leaving S

Optimality: max flow = min cut.

At termination of augmenting path algorithm.

No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

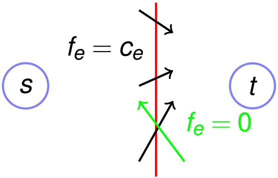
No arc with positive residual capacity leaving S

$\Longrightarrow$ All arcs leaving S are full.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

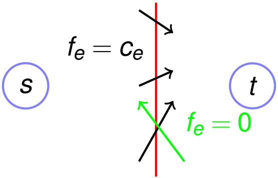
$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
  No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

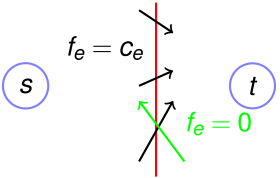
$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S,T)$.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

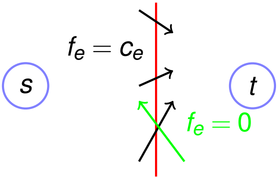
Total flow leaving S is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
  No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

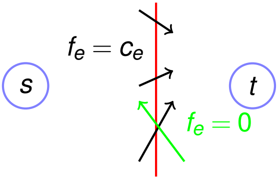
Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

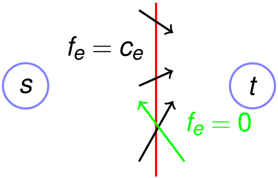
Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

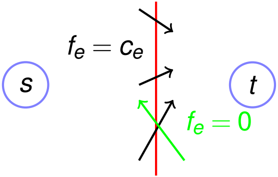
Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
  No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

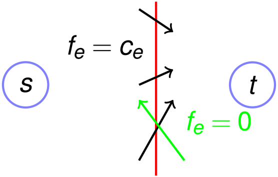
Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
$\rightarrow$

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S,T)$.

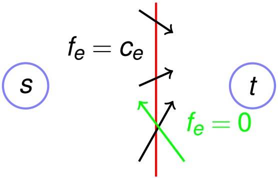
Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S,T)$. and Optimal is $\leq C(S,T)$.
$\rightarrow$ Flow is maximum!!

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S,T)$.

Valid flow $\implies$ all that flow from source.

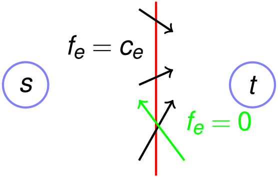
Value of flow equals value of $C(S,T)$. and Optimal is $\leq C(S,T)$.
$\rightarrow$ Flow is maximum!!

Cut is minimum $s-t$ cut too!

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
  No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

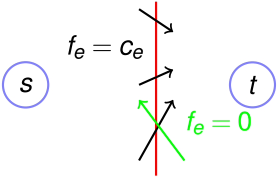
Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
$\rightarrow$ Flow is maximum!!

Cut is minimum $s - t$ cut too!
"any flow" $\leq$ "any cut"

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\Longrightarrow$ All arcs leaving S are full.

$\Longrightarrow$ No arcs into S have flow.

Total flow leaving S is $C(S,T)$.

Valid flow $\Longrightarrow$ all that flow from source.

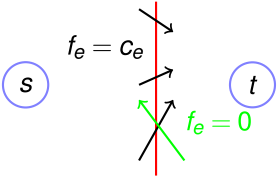
Value of flow equals value of $C(S,T)$. and Optimal is $\leq C(S,T)$.
$\rightarrow$ Flow is maximum!!

Cut is minimum $s-t$ cut too!
"any flow" $\leq$ "any cut" and

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
  No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

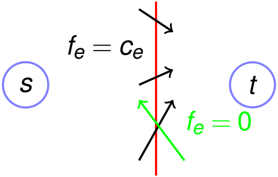
Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
$\rightarrow$ Flow is maximum!!

Cut is minimum $s - t$ cut too!
"any flow" $\leq$ "any cut" and this flow $=$ this cut.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\Longrightarrow$ All arcs leaving S are full.

$\Longrightarrow$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

Valid flow $\Longrightarrow$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
$\rightarrow$ Flow is maximum!!

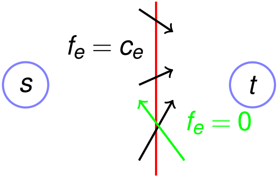
Cut is minimum $s - t$ cut too!
"any flow" $\leq$ "any cut" and this flow $=$ this cut.

$\rightarrow$

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
$\rightarrow$ Flow is maximum!!

Cut is minimum $s - t$ cut too!
"any flow" $\leq$ "any cut" and this flow $=$ this cut.
$\rightarrow$ Maximum flow and minimum $s - t$ cut!

Celebrated max flow -minimum cut theorem.

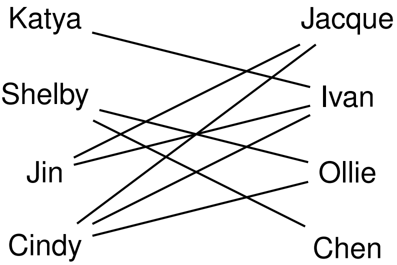
Theorem: In any flow network, the maximum s-t flow is equal to the minimum cut.

Bipartite Matching

Given a bipartite graph: $B = (L, R, E)$ where $E \subseteq L \times R$.

Bipartite Matching

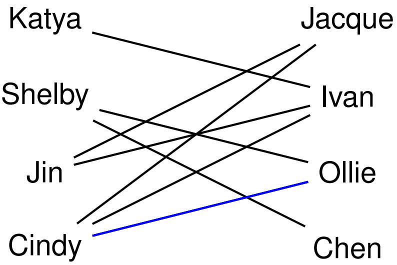
Given a bipartite graph: $B = (L, R, E)$ where $E \subseteq L \times R$.



Find largest subset of edges (“matches”) which are one to one.

Bipartite Matching

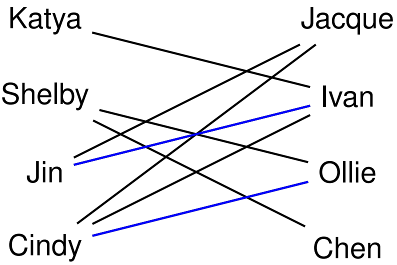
Given a bipartite graph: $B = (L, R, E)$ where $E \subseteq L \times R$.



Find largest subset of edges ("matches") which are one to one.

Bipartite Matching

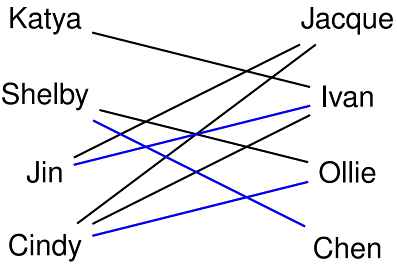
Given a bipartite graph: $B = (L, R, E)$ where $E \subseteq L \times R$.



Find largest subset of edges ("matches") which are one to one.

Bipartite Matching

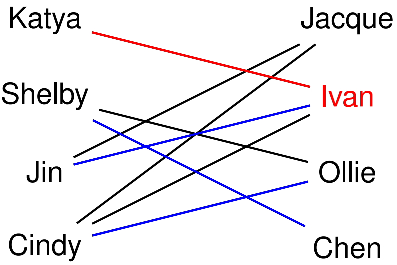
Given a bipartite graph: $B = (L, R, E)$ where $E \subseteq L \times R$.



Find largest subset of edges ("matches") which are one to one.

Bipartite Matching

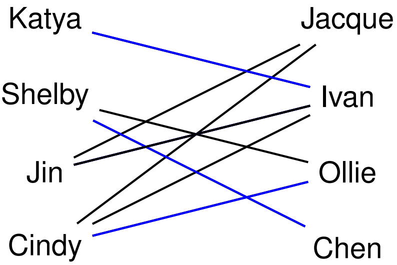
Given a bipartite graph: $B = (L, R, E)$ where $E \subseteq L \times R$.



Find largest subset of edges ("matches") which are one to one.

Bipartite Matching

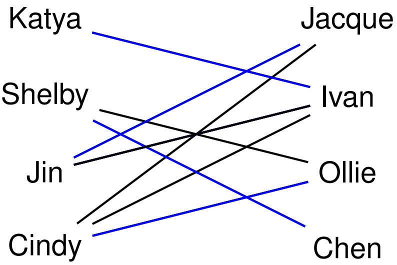
Given a bipartite graph: $B = (L, R, E)$ where $E \subseteq L \times R$.



Find largest subset of edges ("matches") which are one to one.

Bipartite Matching

Given a bipartite graph: $B = (L, R, E)$ where $E \subseteq L \times R$.



Find largest subset of edges ("matches") which are one to one.

Bipartite Matching

Algorithm by “Reduction.”:

From matching problem produce flow problem.

Bipartite Matching

Algorithm by "Reduction.":

From matching problem produce flow problem.

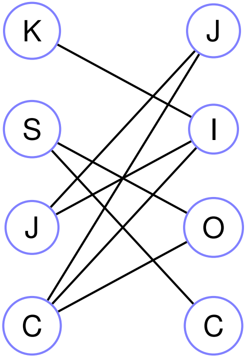
From flow solution produce matching solution.

Bipartite Matching

Algorithm by “Reduction.”:

From matching problem produce flow problem.

From flow solution produce matching solution.

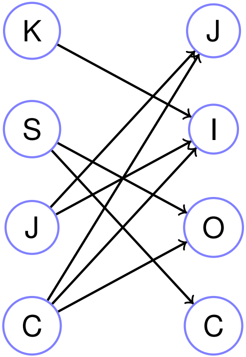


Bipartite Matching

Algorithm by “Reduction.”:

From matching problem produce flow problem.

From flow solution produce matching solution.

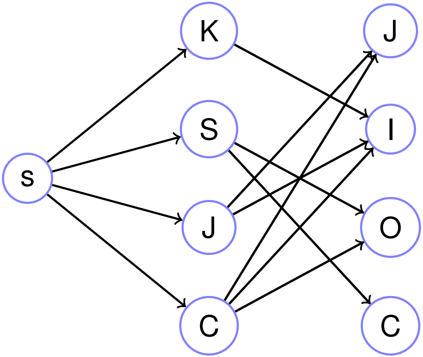


Bipartite Matching

Algorithm by “Reduction.”:

From matching problem produce flow problem.

From flow solution produce matching solution.

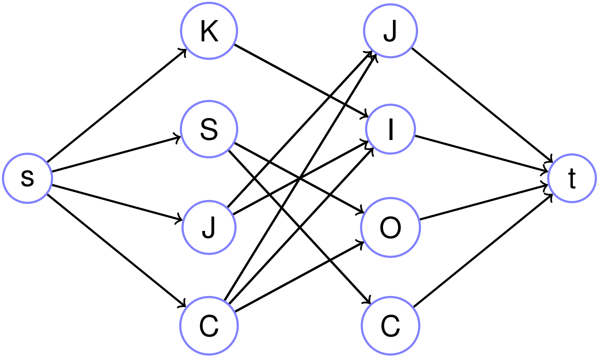


Bipartite Matching

Algorithm by “Reduction.”:

From matching problem produce flow problem.

From flow solution produce matching solution.

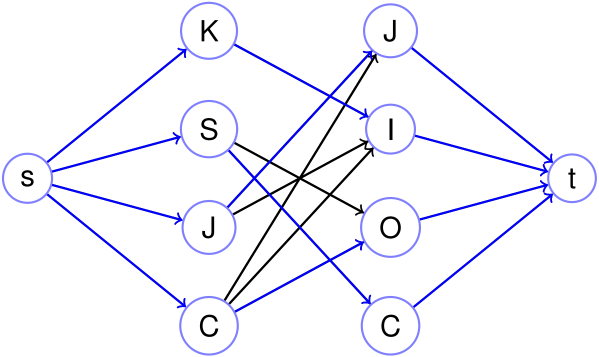


Bipartite Matching

Algorithm by “Reduction.”:

From matching problem produce flow problem.

From flow solution produce matching solution.

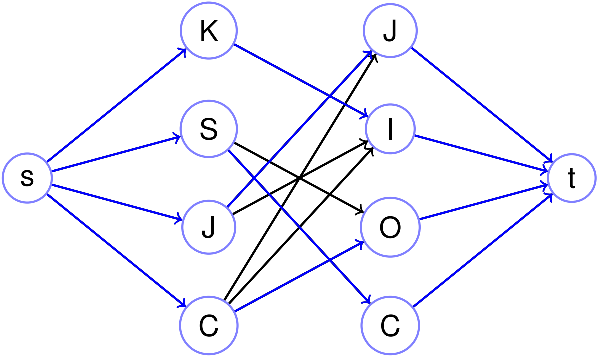


Bipartite Matching

Algorithm by “Reduction.”:

From matching problem produce flow problem.

From flow solution produce matching solution.



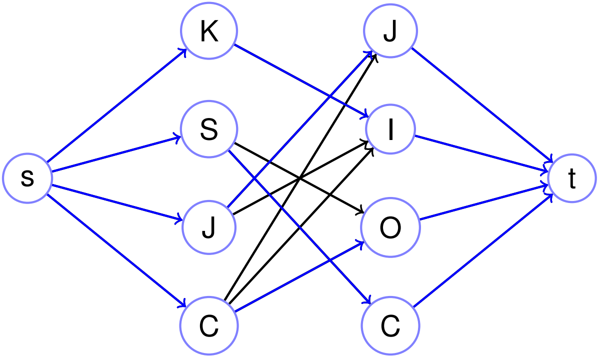
Max flow = Max Matching Size.

Bipartite Matching

Algorithm by "Reduction.":

From matching problem produce flow problem.

From flow solution produce matching solution.



Max flow = Max Matching Size.

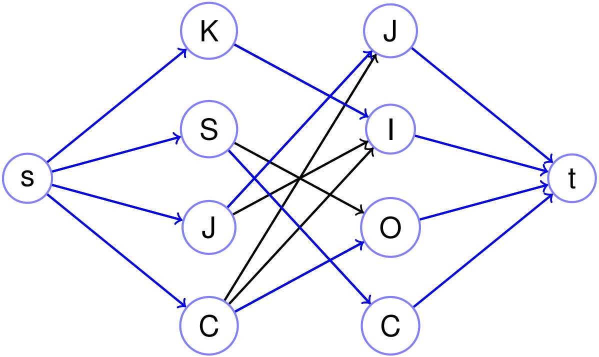
Flow is not integer necessarily....

Bipartite Matching

Algorithm by "Reduction.":

From matching problem produce flow problem.

From flow solution produce matching solution.



Max flow = Max Matching Size.

Flow is not integer necessarily....

Augmenting path algorithm gives integer flow.

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\leq 4 \\ x_2 &\leq 3 \\ x_1 + 2x_2 &\leq 7 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\leq 4 \\ x_2 &\leq 3 \\ x_1 + 2x_2 &\leq 7 \\ x_1, x_2 &\geq 0 \end{aligned}$$

One Solution: $x_1 = 1, x_2 = 3$.

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\le 4 \\ x_2 &\le 3 \\ x_1 + 2x_2 &\le 7 \\ x_1, x_2 &\ge 0 \end{aligned}$$

One Solution: $x_1 = 1, x_2 = 3$. Value is 25.

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\le 4 \\ x_2 &\le 3 \\ x_1 + 2x_2 &\le 7 \\ x_1, x_2 &\ge 0 \end{aligned}$$

One Solution: $x_1 = 1, x_2 = 3$. Value is 25.

Best possible?

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\le 4 \\ x_2 &\le 3 \\ x_1 + 2x_2 &\le 7 \\ x_1, x_2 &\ge 0 \end{aligned}$$

One Solution: $x_1 = 1, x_2 = 3$. Value is 25.

Best possible?

For any solution.
$x_1 \le 4$ and $x_2 \le 3$ ..

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\leq 4 \\ x_2 &\leq 3 \\ x_1 + 2x_2 &\leq 7 \\ x_1, x_2 &\geq 0 \end{aligned}$$

One Solution: $x_1 = 1, x_2 = 3$. Value is 25.

Best possible?

For any solution.
$x_1 \leq 4$ and $x_2 \leq 3$ ..
....so $x_1 + 8x_2 \leq 4 + 8(3) = 28$.
Added equation 1 and 8 times equation 2
yields bound on objective..

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\leq 4 \\ x_2 &\leq 3 \\ x_1 + 2x_2 &\leq 7 \\ x_1, x_2 &\geq 0 \end{aligned}$$

One Solution: $x_1 = 1, x_2 = 3$. Value is 25.

Best possible?

For any solution.
$x_1 \leq 4$ and $x_2 \leq 3$ ..
....so $x_1 + 8x_2 \leq 4 + 8(3) = 28$.
Added equation 1 and 8 times equation 2
yields bound on objective..
Better solution?

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\leq 4 \\ x_2 &\leq 3 \\ x_1 + 2x_2 &\leq 7 \\ x_1, x_2 &\geq 0 \end{aligned}$$

One Solution: $x_1 = 1, x_2 = 3$. Value is 25.

Best possible?

For any solution.
$x_1 \leq 4$ and $x_2 \leq 3$ ..
....so $x_1 + 8x_2 \leq 4 + 8(3) = 28$.
Added equation 1 and 8 times equation 2
yields bound on objective..
Better solution?
Better upper bound?

Duality.

$$\max x_1 + 8x_2$$

$$x_1 \leq 4$$
$$x_2 \leq 3$$
$$x_1 + 2x_2 \leq 7$$
$$x_1, x_2 \geq 0$$

One Solution: $x_1 = 1, x_2 = 3$. Value is 25.

Best possible?

For any solution.
$x_1 \leq 4$ and $x_2 \leq 3$ ..
....so $x_1 + 8x_2 \leq 4 + 8(3) = 28$.
Added equation 1 and 8 times equation 2
yields bound on objective..
Better solution?
Better upper bound?

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\leq 4 \\ x_2 &\leq 3 \\ x_1 + 2x_2 &\leq 7 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Solution value: 25.

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\leq 4 \\ x_2 &\leq 3 \\ x_1 + 2x_2 &\leq 7 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Solution value: 25.
Add equation 1 and 8 times equation 2 gives..

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\le 4 \\ x_2 &\le 3 \\ x_1 + 2x_2 &\le 7 \\ x_1, x_2 &\ge 0 \end{aligned}$$

Solution value: 25.
Add equation 1 and 8 times equation 2 gives..
$x_1 + 8x_2 \le 4 + 24 = 28$.

Duality.

$$\max x_1 + 8x_2$$

$$x_1 \leq 4$$
$$x_2 \leq 3$$
$$x_1 + 2x_2 \leq 7$$
$$x_1, x_2 \geq 0$$

Solution value: 25.
Add equation 1 and 8 times equation 2 gives..
$x_1 + 8x_2 \leq 4 + 24 = 28.$

Better way to add equations to get bound on function?

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\leq 4 \\ x_2 &\leq 3 \\ x_1 + 2x_2 &\leq 7 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Solution value: 25.
Add equation 1 and 8 times equation 2 gives..
$x_1 + 8x_2 \leq 4 + 24 = 28$.

Better way to add equations to get bound on function?
Sure:

Duality.

$$\max x_1 + 8x_2$$

$$x_1 \leq 4$$
$$x_2 \leq 3$$
$$x_1 + 2x_2 \leq 7$$
$$x_1, x_2 \geq 0$$

Solution value: 25.
Add equation 1 and 8 times equation 2 gives..
$x_1 + 8x_2 \leq 4 + 24 = 28$.

Better way to add equations to get bound on function?
Sure: 6 times equation 2 and 1 times equation 3.

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\le 4 \\ x_2 &\le 3 \\ x_1 + 2x_2 &\le 7 \\ x_1, x_2 &\ge 0 \end{aligned}$$

Solution value: 25.
Add equation 1 and 8 times equation 2 gives..
$x_1 + 8x_2 \le 4 + 24 = 28$.

Better way to add equations to get bound on function?
Sure: 6 times equation 2 and 1 times equation 3.

$x_1 + 8x_2 \le 6(3) + 7 = 25$.

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\leq 4 \\ x_2 &\leq 3 \\ x_1 + 2x_2 &\leq 7 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Solution value: 25.
Add equation 1 and 8 times equation 2 gives..
$x_1 + 8x_2 \leq 4 + 24 = 28$.

Better way to add equations to get bound on function?
Sure: 6 times equation 2 and 1 times equation 3.

$x_1 + 8x_2 \leq 6(3) + 7 = 25$.

Thus, the value is at most 25.

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\le 4 \\ x_2 &\le 3 \\ x_1 + 2x_2 &\le 7 \\ x_1, x_2 &\ge 0 \end{aligned}$$

Solution value: 25.
Add equation 1 and 8 times equation 2 gives..
$x_1 + 8x_2 \le 4 + 24 = 28$.

Better way to add equations to get bound on function?
Sure: 6 times equation 2 and 1 times equation 3.
$x_1 + 8x_2 \le 6(3) + 7 = 25$.

Thus, the value is at most 25.

The upper bound is same as solution!

Duality.

$$\max x_1 + 8x_2$$

$$\begin{aligned} x_1 &\le 4 \\ x_2 &\le 3 \\ x_1 + 2x_2 &\le 7 \\ x_1, x_2 &\ge 0 \end{aligned}$$

Solution value: 25.
Add equation 1 and 8 times equation 2 gives..
$x_1 + 8x_2 \le 4 + 24 = 28$.

Better way to add equations to get bound on function?
Sure: 6 times equation 2 and 1 times equation 3.

$x_1 + 8x_2 \le 6(3) + 7 = 25$.

Thus, the value is at most 25.

The upper bound is same as solution!

Proof of optimality!

Duality:example

Idea: Add up positive linear combination of inequalities to “get” upper bound on optimization function.

Duality:example

Idea: Add up positive linear combination of inequalities to “get” upper bound on optimization function.

Will this always work?

Duality:example

Idea: Add up positive linear combination of inequalities to “get” upper bound on optimization function.

Will this always work?

How to find best upper bound?

Duality: computing upper bound.

Best Upper Bound.

Multiplier	Inequality
y_1	$x_1 \leq 4$
y_2	$x_2 \leq 3$
y_3	$x_1 + 2x_2 \leq 7$

Adding equations thusly...

Duality: computing upper bound.

Best Upper Bound.

Multiplier	Inequality
y_1	$x_1 \leq 4$
y_2	$x_2 \leq 3$
y_3	$x_1 + 2x_2 \leq 7$

Adding equations thusly...

$$(y_1 + y_3)x_1 + (y_2 + 2y_3)x_2 \leq 4y_1 + 3y_2 + 7y_3.$$

Duality: computing upper bound.

Best Upper Bound.

Multiplier	Inequality
y_1	$x_1 \leq 4$
y_2	$x_2 \leq 3$
y_3	$x_1 + 2x_2 \leq 7$

Adding equations thusly...

$$(y_1 + y_3)x_1 + (y_2 + 2y_3)x_2 \leq 4y_1 + 3y_2 + 7y_3.$$

The left hand side should "dominate" optimization function:

Duality: computing upper bound.

Best Upper Bound.

Multiplier	Inequality
y_1	$x_1 \leq 4$
y_2	$x_2 \leq 3$
y_3	$x_1 + 2x_2 \leq 7$

Adding equations thusly...

$$(y_1 + y_3)x_1 + (y_2 + 2y_3)x_2 \leq 4y_1 + 3y_2 + 7y_3.$$

The left hand side should "dominate" optimization function:

If $y_1, y_2, y_3 \geq 0$

Duality: computing upper bound.

Best Upper Bound.

Multiplier	Inequality
y_1	$x_1 \leq 4$
y_2	$x_2 \leq 3$
y_3	$x_1 + 2x_2 \leq 7$

Adding equations thusly...

$$(y_1 + y_3)x_1 + (y_2 + 2y_3)x_2 \leq 4y_1 + 3y_2 + 7y_3.$$

The left hand side should "dominate" optimization function:

If $y_1, y_2, y_3 \geq 0$
and $y_1 + y_3 \geq 1$ and $y_2 + 2y_3 \geq 8$ then..

Duality: computing upper bound.

Best Upper Bound.

Multiplier	Inequality
y_1	$x_1 \leq 4$
y_2	$x_2 \leq 3$
y_3	$x_1 + 2x_2 \leq 7$

Adding equations thusly...

$$(y_1 + y_3)x_1 + (y_2 + 2y_3)x_2 \leq 4y_1 + 3y_2 + 7y_3.$$

The left hand side should "dominate" optimization function:

If $y_1, y_2, y_3 \geq 0$
and $y_1 + y_3 \geq 1$ and $y_2 + 2y_3 \geq 8$ then..

$$x_1 + 8x_2 \leq 4y_1 + 3y_2 + 7y_3$$

Duality: computing upper bound.

Best Upper Bound.

Multiplier	Inequality
y_1	$x_1 \leq 4$
y_2	$x_2 \leq 3$
y_3	$x_1 + 2x_2 \leq 7$

Adding equations thusly...

$$(y_1 + y_3)x_1 + (y_2 + 2y_3)x_2 \leq 4y_1 + 3y_2 + 7y_3.$$

The left hand side should "dominate" optimization function:

If $y_1, y_2, y_3 \geq 0$
and $y_1 + y_3 \geq 1$ and $y_2 + 2y_3 \geq 8$ then..

$$x_1 + 8x_2 \leq 4y_1 + 3y_2 + 7y_3$$

Find best y_i's to minimize upper bound?

The dual, the dual, the dual.

Find best y_i's to minimize upper bound?

The dual, the dual, the dual.

Find best y_i's to minimize upper bound?

Again: If you find $y_1, y_2, y_3 \geq 0$

The dual, the dual, the dual.

Find best y_i's to minimize upper bound?

Again: If you find $y_1, y_2, y_3 \geq 0$
and $y_1 + y_3 \geq 1$ and $y_2 + 2y_3 \geq 8$ then..

The dual, the dual, the dual.

Find best y_i's to minimize upper bound?

Again: If you find $y_1, y_2, y_3 \geq 0$
and $y_1 + y_3 \geq 1$ and $y_2 + 2y_3 \geq 8$ then..

$$x_1 + 8x_2 \leq 4y_1 + 3y_2 + 7y_3$$

The dual, the dual, the dual.

Find best y_i's to minimize upper bound?

Again: If you find $y_1, y_2, y_3 \geq 0$
and $y_1 + y_3 \geq 1$ and $y_2 + 2y_3 \geq 8$ then..

$$x_1 + 8x_2 \leq 4y_1 + 3y_2 + 7y_3$$

$$\begin{aligned} \min 4y_1 + 3y_2 + 7y_3 \\ y_1 + y_3 \geq 1 \\ y_2 + 2y_3 \geq 8 \\ y_1, y_2, y_3 \geq 0 \end{aligned}$$

The dual, the dual, the dual.

Find best y_i's to minimize upper bound?

Again: If you find $y_1, y_2, y_3 \geq 0$
and $y_1 + y_3 \geq 1$ and $y_2 + 2y_3 \geq 8$ then..
$x_1 + 8x_2 \leq 4y_1 + 3y_2 + 7y_3$

$$\begin{aligned} \min 4y_1 + 3y_2 + 7y_3 \\ y_1 + y_3 \geq 1 \\ y_2 + 2y_3 \geq 8 \\ y_1, y_2, y_3 \geq 0 \end{aligned}$$

A linear program.

The dual, the dual, the dual.

Find best y_i's to minimize upper bound?

Again: If you find $y_1, y_2, y_3 \geq 0$
and $y_1 + y_3 \geq 1$ and $y_2 + 2y_3 \geq 8$ then..

$$x_1 + 8x_2 \leq 4y_1 + 3y_2 + 7y_3$$

$$\begin{aligned} \min 4y_1 + 3y_2 + 7y_3 \\ y_1 + y_3 \geq 1 \\ y_2 + 2y_3 \geq 8 \\ y_1, y_2, y_3 \geq 0 \end{aligned}$$

A linear program.
The Dual linear program.

The dual, the dual, the dual.

Find best y_i's to minimize upper bound?

Again: If you find $y_1, y_2, y_3 \geq 0$
and $y_1 + y_3 \geq 1$ and $y_2 + 2y_3 \geq 8$ then..
$x_1 + 8x_2 \leq 4y_1 + 3y_2 + 7y_3$

$$\begin{aligned} \min\ 4y_1 + 3y_2 + 7y_3 \\ y_1 + y_3 \geq 1 \\ y_2 + 2y_3 \geq 8 \\ y_1, y_2, y_3 \geq 0 \end{aligned}$$

A linear program.
The Dual linear program.

Primal: $(x_1, x_2) = (1, 3)$; Dual: $(y_1, y_2, y_3) = (0, 6, 1)$.

The dual, the dual, the dual.

Find best y_i's to minimize upper bound?

Again: If you find $y_1, y_2, y_3 \geq 0$
and $y_1 + y_3 \geq 1$ and $y_2 + 2y_3 \geq 8$ then..
$x_1 + 8x_2 \leq 4y_1 + 3y_2 + 7y_3$

$$\begin{aligned} \min 4y_1 + 3y_2 + 7y_3 \\ y_1 + y_3 \geq 1 \\ y_2 + 2y_3 \geq 8 \\ y_1, y_2, y_3 \geq 0 \end{aligned}$$

A linear program.
The Dual linear program.

Primal: $(x_1, x_2) = (1,3)$; Dual: $(y_1, y_2, y_3) = (0,6,1)$.

Value of both is 25!

The dual, the dual, the dual.

Find best y_i's to minimize upper bound?

Again: If you find $y_1, y_2, y_3 \geq 0$
and $y_1 + y_3 \geq 1$ and $y_2 + 2y_3 \geq 8$ then..
$x_1 + 8x_2 \leq 4y_1 + 3y_2 + 7y_3$

$$
\begin{aligned}
\min 4y_1 + 3y_2 + 7y_3 \\
y_1 + y_3 \geq 1 \\
y_2 + 2y_3 \geq 8 \\
y_1, y_2, y_3 \geq 0
\end{aligned}
$$

A linear program.
The Dual linear program.

Primal: $(x_1, x_2) = (1, 3)$; Dual: $(y_1, y_2, y_3) = (0, 6, 1)$.

Value of both is 25!

Primal is optimal

The dual, the dual, the dual.

Find best y_i's to minimize upper bound?

Again: If you find $y_1, y_2, y_3 \geq 0$
and $y_1 + y_3 \geq 1$ and $y_2 + 2y_3 \geq 8$ then..
$x_1 + 8x_2 \leq 4y_1 + 3y_2 + 7y_3$

$$\begin{aligned} \min 4y_1 + 3y_2 + 7y_3 \\ y_1 + y_3 \geq 1 \\ y_2 + 2y_3 \geq 8 \\ y_1, y_2, y_3 \geq 0 \end{aligned}$$

A linear program.
The Dual linear program.

Primal: $(x_1, x_2) = (1, 3)$; Dual: $(y_1, y_2, y_3) = (0, 6, 1)$.

Value of both is 25!

Primal is optimal ... and dual is optimal!

The dual.

In general.

Primal LP	Dual LP
$\max c \cdot x$	$\min y^T b$
$Ax \leq b$	$y^T A \geq c$
$x \geq 0$	$y \geq 0$

The dual.

In general.

Primal LP	Dual LP
$\max c \cdot x$	$\min y^T b$
$Ax \leq b$	$y^T A \geq c$
$x \geq 0$	$y \geq 0$

Theorem: If a linear program has a bounded value, then its dual is bounded and has the same value.

The dual.

In general.

Primal LP	Dual LP
$\max c \cdot x$	$\min y^T b$
$Ax \leq b$	$y^T A \geq c$
$x \geq 0$	$y \geq 0$

Theorem: If a linear program has a bounded value, then its dual is bounded and has the same value.

Simplex Algorithm.

Simplex Algorithm.

Start at a vertex.

Simplex Algorithm.

Start at a vertex.

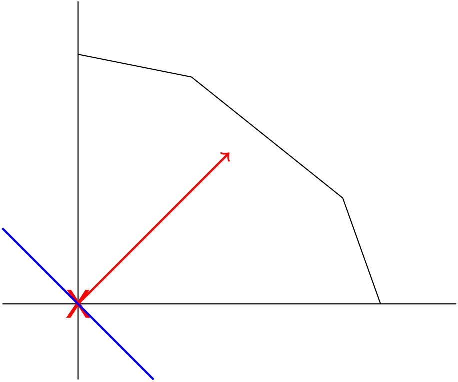
Move to better neighboring vertex.

Simplex Algorithm.

Start at a vertex.

Move to better neighboring vertex.

Until no better neighboring vertex.



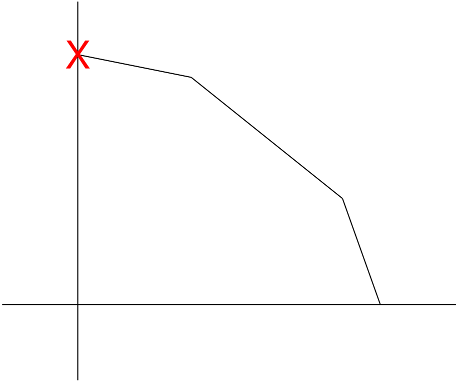
$$\max(x_1 + x_2)$$
$$7x_1 + 5x_2 \leq 20$$
$$4x_1 + 5x_2 \leq 21$$
$$2x_1 + 10x_2 \leq 33$$
$$x_1 \geq 0, x_2 \geq 0$$

Simplex Algorithm.

Start at a vertex.

Move to better neighboring vertex.

Until no better neighboring vertex.



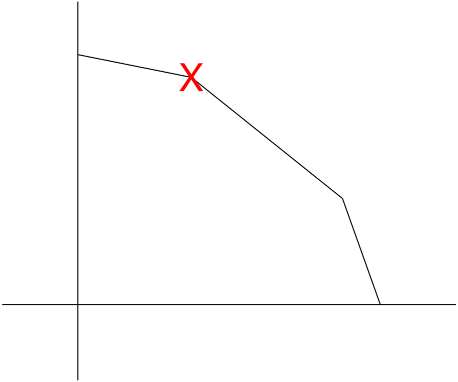
$$\max(x_1 + x_2)$$
$$7x_1 + 5x_2 \leq 20$$
$$4x_1 + 5x_2 \leq 21$$
$$2x_1 + 10x_2 \leq 33$$
$$x_1 \geq 0, x_2 \geq 0$$

Simplex Algorithm.

Start at a vertex.

Move to better neighboring vertex.

Until no better neighboring vertex.



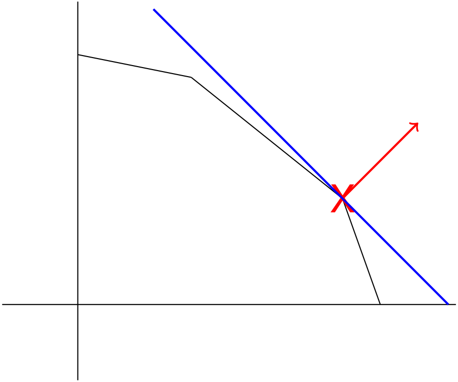
$$\max(x_1 + x_2)$$
$$7x_1 + 5x_2 \leq 20$$
$$4x_1 + 5x_2 \leq 21$$
$$2x_1 + 10x_2 \leq 33$$
$$x_1 \geq 0, x_2 \geq 0$$

Simplex Algorithm.

Start at a vertex.

Move to better neighboring vertex.

Until no better neighboring vertex.



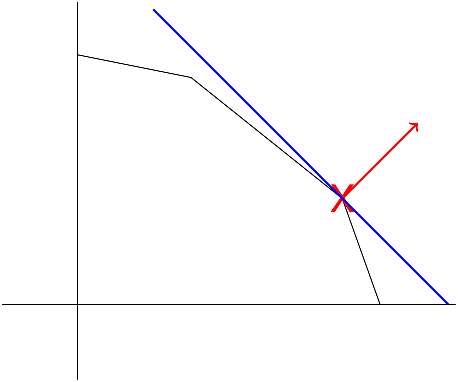
$$\max(x_1 + x_2)$$
$$7x_1 + 5x_2 \leq 20$$
$$4x_1 + 5x_2 \leq 21$$
$$2x_1 + 10x_2 \leq 33$$
$$x_1 \geq 0, x_2 \geq 0$$

Simplex Algorithm.

Start at a vertex.

Move to better neighboring vertex.

Until no better neighboring vertex.



$$\max(x_1 + x_2)$$
$$7x_1 + 5x_2 \leq 20$$
$$4x_1 + 5x_2 \leq 21$$
$$2x_1 + 10x_2 \leq 33$$
$$x_1 \geq 0, x_2 \geq 0$$

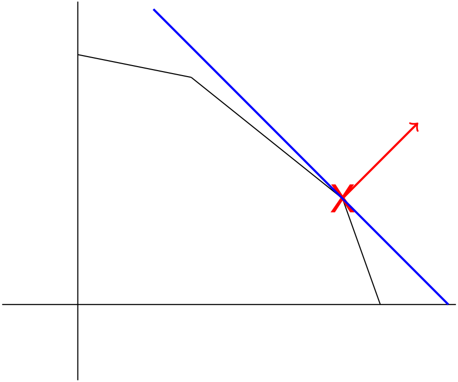
Why optimal?

Simplex Algorithm.

Start at a vertex.

Move to better neighboring vertex.

Until no better neighboring vertex.



$$\max(x_1 + x_2)$$
$$7x_1 + 5x_2 \leq 20$$
$$4x_1 + 5x_2 \leq 21$$
$$2x_1 + 10x_2 \leq 33$$
$$x_1 \geq 0, x_2 \geq 0$$

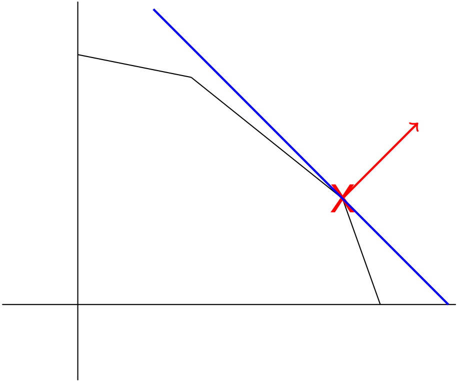
Why optimal? Draw line corresponding to $cx =$ current value.

Simplex Algorithm.

Start at a vertex.

Move to better neighboring vertex.

Until no better neighboring vertex.



$$\max(x_1 + x_2)$$
$$7x_1 + 5x_2 \leq 20$$
$$4x_1 + 5x_2 \leq 21$$
$$2x_1 + 10x_2 \leq 33$$
$$x_1 \geq 0, x_2 \geq 0$$

Why optimal? Draw line corresponding to $cx =$ current value.
Entire feasible region on "wrong" side.

Example: review.

$$\max x_1 + 8x_2$$
$$x_1 \le 4$$
$$x_2 \le 3$$
$$x_1 + 2x_2 \le 7$$
$$y_1, y_2, y_3 \ge 0$$

$$\min 4y_1 + 3y_2 + 7y_3$$
$$y_1 + y_3 \ge 1$$
$$y_2 + 2y_3 \ge 8$$
$$x_1, x_2 \ge 0$$

"Matrix form"

$$\max[1,8] \cdot [x_1, x_2]$$
$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} 4 \\ 3 \\ 7 \end{bmatrix}$$
$$[x_1, x_2] \ge 0$$

$$\min[4,3,7] \cdot [y_1, y_2, y_3]$$
$$[y_1, y_2, y_3] \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 2 \end{pmatrix} \ge \begin{bmatrix} 1 \\ 8 \end{bmatrix}$$
$$[y_1, y_2, y_3] \ge 0$$

Matrix equations.

$$\max[1,8]\cdot[x_1,x_2]$$
$$\begin{pmatrix}1 & 0\\ 0 & 1\\ 1 & 2\end{pmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}\le\begin{bmatrix}4\\ 3\\ 7\end{bmatrix}$$
$$[x_1,x_2]\ge 0$$

$$\min[4,3,7]\cdot[y_1,y_2,y_3]$$
$$[y_1,y_2,y_3]\begin{pmatrix}1 & 0\\ 0 & 1\\ 1 & 2\end{pmatrix}\ge\begin{bmatrix}1\\ 8\end{bmatrix}$$
$$[y_1,y_2,y_3]\ge 0$$

We can rewrite the above in matrix form.

$$A=\begin{pmatrix}1 & 0\\ 0 & 1\\ 1 & 2\end{pmatrix}\qquad c=[1,8]\quad b=[4,3,7]$$

The primal is $Ax\le b, \max c\cdot x, x\ge 0$.
The dual is $y^TA\ge c, \min b\cdot y, y\ge 0$.

Generality of Linear Programming.

Linear program solves many problems.

Generality of Linear Programming.

Linear program solves many problems.

How applicable is it?

Circuit Evaluation.

Circuit Evaluation:

Circuit Evaluation.

Circuit Evaluation:
Given: DAG of boolean gates:

Circuit Evaluation.

Circuit Evaluation:
Given: DAG of boolean gates:
  two input **AND**/**OR**.

Circuit Evaluation.

Circuit Evaluation:
Given: DAG of boolean gates:
  two input **AND/OR**.
  One input **NOT**.

Circuit Evaluation.

Circuit Evaluation:
Given: DAG of boolean gates:
  two input **AND**/**OR**.
  One input **NOT**.
TRUE/**FALSE** inputs.

Circuit Evaluation.

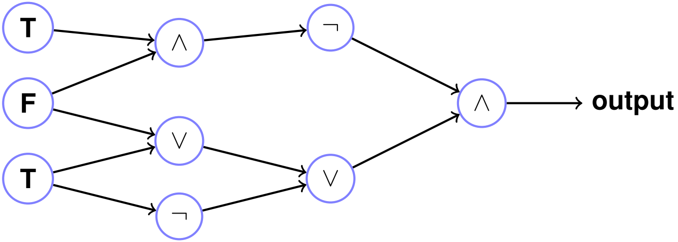
Circuit Evaluation:
Given: DAG of boolean gates:
two input **AND**/**OR**.
One input **NOT**.
TRUE/**FALSE** inputs.

Problem: What is the output of a specified Output Gate?

Circuit Evaluation.

Circuit Evaluation:
Given: DAG of boolean gates:
  two input **AND**/**OR**.
  One input **NOT**.
TRUE/**FALSE** inputs.

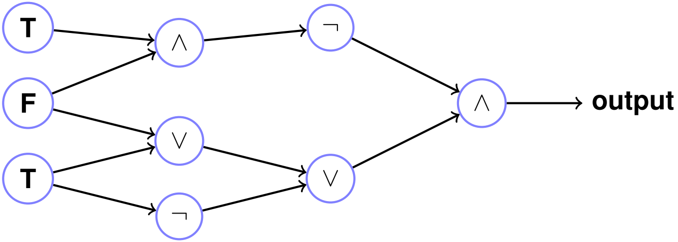
Problem: What is the output of a specified Output Gate?



Circuit Evaluation.

Circuit Evaluation:
Given: DAG of boolean gates:
  two input **AND**/**OR**.
  One input **NOT**.
TRUE/**FALSE** inputs.

Problem: What is the output of a specified Output Gate?



What is the value of the output?

Translation to linear program.

Variable for gate g: x_g.

Translation to linear program.

Variable for gate g: x_g.

Constraints:

Translation to linear program.

Variable for gate g: x_g.

Constraints:
$0 \leq x_g \leq 1$

Translation to linear program.

Variable for gate g: x_g.

Constraints:
$0 \leq x_g \leq 1$
Gate g is true gate: $x_g = 1$.

Translation to linear program.

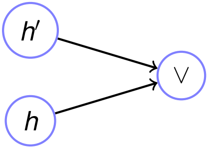
Variable for gate g: x_g.

Constraints:
$0 \leq x_g \leq 1$
Gate g is true gate: $x_g = 1$.
Gate g is false gate: $x_g = 0$.

Translation to linear program.

Variable for gate g: x_g.

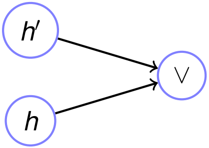
Constraints:
$0 \leq x_g \leq 1$
Gate g is true gate: $x_g = 1$.
Gate g is false gate: $x_g = 0$.



Translation to linear program.

Variable for gate g: x_g.

Constraints:
$0 \leq x_g \leq 1$
Gate g is true gate: $x_g = 1$.
Gate g is false gate: $x_g = 0$.

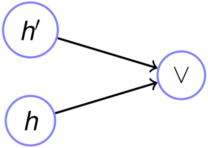


Translation to linear program.

Variable for gate g: x_g.

Constraints:
$0 \leq x_g \leq 1$
Gate g is true gate: $x_g = 1$.
Gate g is false gate: $x_g = 0$.

$x_g \geq x_h$

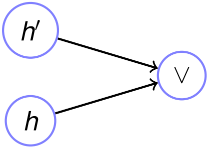


Translation to linear program.

Variable for gate g: x_g.

Constraints:
$0 \leq x_g \leq 1$
Gate g is true gate: $x_g = 1$.
Gate g is false gate: $x_g = 0$.

$x_g \geq x_h$
$x_g \geq x_{h'}$

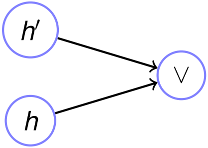


Translation to linear program.

Variable for gate g: x_g.

Constraints:
$0 \leq x_g \leq 1$
Gate g is true gate: $x_g = 1$.
Gate g is false gate: $x_g = 0$.

$$x_g \geq x_h$$
$$x_g \geq x_{h'}$$
$$x_g \leq x_h + x_{h'}$$

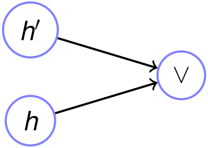


Translation to linear program.

Variable for gate g: x_g.

Constraints:
$0 \leq x_g \leq 1$
Gate g is true gate: $x_g = 1$.
Gate g is false gate: $x_g = 0$.

$$x_g \geq x_h$$
$$x_g \geq x_{h'}$$
$$x_g \leq x_h + x_{h'}$$



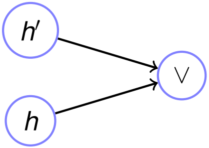
For $\wedge$ gate:
$x_g \leq x_h, \quad x_g \leq x_h'$

Translation to linear program.

Variable for gate g: x_g.

Constraints:
$0 \leq x_g \leq 1$
Gate g is true gate: $x_g = 1$.
Gate g is false gate: $x_g = 0$.

$$x_g \geq x_h$$
$$x_g \geq x_{h'}$$
$$x_g \leq x_h + x_{h'}$$



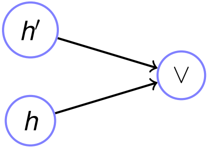
For $\wedge$ gate:
$x_g \leq x_h, \quad x_g \leq x'_h \qquad\qquad x_g \geq x_h + x_{h'} - 1$

Translation to linear program.

Variable for gate g: x_g.

Constraints:
$0 \leq x_g \leq 1$
Gate g is true gate: $x_g = 1$.
Gate g is false gate: $x_g = 0$.

$$x_g \geq x_h$$
$$x_g \geq x_{h'}$$
$$x_g \leq x_h + x_{h'}$$



For $\wedge$ gate:
$x_g \leq x_h, \quad x_g \leq x_h' \qquad\qquad x_g \geq x_h + x_{h'} - 1$

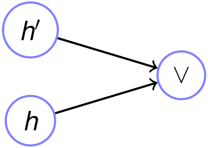
For $\neg$ gate: $x_g = 1 - x_h$.

Translation to linear program.

Variable for gate g: x_g.

Constraints:
$0 \leq x_g \leq 1$
Gate g is true gate: $x_g = 1$.
Gate g is false gate: $x_g = 0$.

$$x_g \geq x_h$$
$$x_g \geq x_{h'}$$
$$x_g \leq x_h + x_{h'}$$



For $\wedge$ gate:
$x_g \leq x_h, \quad x_g \leq x_h' \qquad\qquad x_g \geq x_h + x_{h'} - 1$

For $\neg$ gate: $x_g = 1 - x_h$.

x_o is 1 if and only if the circuit evaluates to true.

What does this mean?

The circuit value problem is completely general!

What does this mean?

The circuit value problem is completely general!

A computer program can be unfolded into a circuit.

What does this mean?

The circuit value problem is completely general!

A computer program can be unfolded into a circuit.

Each level is the circuit for a computer.

What does this mean?

The circuit value problem is completely general!

A computer program can be unfolded into a circuit.

Each level is the circuit for a computer.

The number of levels is the number of steps.

What does this mean?

The circuit value problem is completely general!

A computer program can be unfolded into a circuit.

Each level is the circuit for a computer.

The number of levels is the number of steps.

$\implies$ circuit value problems model computation.

What does this mean?

The circuit value problem is completely general!

A computer program can be unfolded into a circuit.

Each level is the circuit for a computer.

The number of levels is the number of steps.

$\implies$ circuit value problems model computation.

$\implies$ linear programs can model any polynomial time problem!

What does this mean?

The circuit value problem is completely general!

A computer program can be unfolded into a circuit.

Each level is the circuit for a computer.

The number of levels is the number of steps.

$\Longrightarrow$ circuit value problems model computation.

$\Longrightarrow$ linear programs can model any polynomial time problem!

Warning: existence proof, not generally efficient.