COMP 610
Information Structures

http://www.cs.mcgill.ca/~breed/2018COMP610/home.html

Efficient Algorithms Require Appropriate Data Structures

Efficient Algorithms Require Appropriate Data Structures

- An ordered list is easier to search than an unordered list

Efficient Algorithms Require Appropriate Data Structures

- An ordered list is easier to search than an unordered list

- It is easier to multiply 1776 and 1954
  then to multiply MDCCLXXVI and MCMLIV,

What You Should Already Be Familiar With

What You Should Already Be Familiar With

- Arrays, Queues, Stacks, Linked Lists, Trees, and their implementations

What You Should Already Be Familiar With

- Arrays, Queues, Stacks, Linked Lists, Trees, and their implementations
- Analysis of Recursive Algorithms by solving recurrence relations.

What You Should Already Be Familiar With

- Arrays, Queues, Stacks, Linked Lists, Trees, and their implementations
- Analysis of Recursive Algorithms by solving recurrence relations.
- Graphs and Directed Graphs: Adjacency List and Adjacency Matrix Representation, Breadth First Search, Depth First Search, Shortest Path Algorithms, Minimum Weight Spanning Tree Algorithms (Prim and Kruskal)

What You Should Already Be Familiar With

- Arrays, Queues, Stacks, Linked Lists, Trees, and their implementations
- Analysis of Recursive Algorithms by solving recurrence relations.
- Graphs and Directed Graphs: Adjacency List and Adjacency Matrix Representation, Breadth First Search, Depth First Search, Shortest Path Algorithms, Minimum Weight Spanning Tree Algorithms (Prim and Kruskal)
- Quicksort, merge sort, and insertion sort and their worst case analysis

What We Will Focus On

What We Will Focus On

- Graph Algorithms, especially solving problems which look unrelated to graphs using graphs(14 lectures)

What We Will Focus On

- Graph Algorithms, especially solving problems which look unrelated to graphs using graphs(14 lectures)
- Heaps and Heapsort (2 lectures)

What We Will Focus On

- Graph Algorithms, especially solving problems which look unrelated to graphs using graphs(14 lectures)
- Heaps and Heapsort (2 lectures)
- Algorithms for Selection(2 lectures)

What We Will Focus On

- Graph Algorithms, especially solving problems which look unrelated to graphs using graphs(14 lectures)
- Heaps and Heapsort (2 lectures)
- Algorithms for Selection(2 lectures)
- Red-Black Trees(2 lectures)

What We Will Focus On

- Graph Algorithms, especially solving problems which look unrelated to graphs using graphs(14 lectures)
- Heaps and Heapsort (2 lectures)
- Algorithms for Selection(2 lectures)
- Red-Black Trees(2 lectures)
- Compression algorithms including JPEG compression (2 lectures)

This Week’s Example

This Week’s Example

- Solving 2-SAT via Depth First Search

This Week’s Example

- Solving 2-SAT via Depth First Search
- First reduce SAT to 3-SAT.

Boolean Variables And Their Negations

Boolean Variables And Their Negations

- *Boolean Variables* take the values true or fase.

Boolean Variables And Their Negations

- *Boolean Variables* take the values true or fase.
- We use $\neg$ for NOT, and say $\neg x$ is the *negation* of x. So $\neg x$ evaluates to true if x is false, and to false if x is true.

Boolean Variables And Their Negations

- *Boolean Variables* take the values true or fase.
- We use $\neg$ for NOT, and say $\neg x$ is the *negation* of x. So $\neg x$ evaluates to true if x is false, and to false if x is true.
- $\neg(\neg x)=x$.

Boolean Variables And Their Negations

- *Boolean Variables* take the values true or fase.
- We use $\neg$ for NOT, and say $\neg x$ is the *negation* of x. So $\neg x$ evaluates to true if x is false, and to false if x is true.
- $\neg(\neg x)=x$.
- We call x and $\neg x$ the *literals* associated with the Boolean variable x.

Boolean Formulas

Boolean Formulas

- A literal is a formula

Boolean Formulas

- A literal is a formula
- If $F_1,\ldots,F_l$ are Boolean Formulas so is F_1 AND F_2 AND .... AND F_l

Boolean Formulas

- A literal is a formula
- If $F_1,\ldots,F_l$ are Boolean Formulas so is F_1 AND F_2 AND .... AND F_l
- If $F_1,\ldots,F_l$ are Boolean Formulas so is F_1 OR F_2 OR .... OR F_l

Some Boolean Formulas

- F_1 $\neg x_1$
- F_2 x_1 *AND* x_2
- F_3 (x_1 OR x_2) AND $\neg x_1$ AND $\neg x_2$
- F_4 (x_1 OR $\neg x_2$) AND (x_2 OR $\neg x_3$) AND (x_3 OR $\neg x_4$) AND (x_4 OR $\neg x_1$)
- F_5 (x_1 OR $\neg x_2$) AND (x_2 OR $\neg x_3$) AND (x_3 OR $\neg x_4$) AND (x_4 OR $\neg x_1$)
  AND (x_1 OR x_2 OR x_3 OR x_4) AND ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$)
- F_6 ($\neg x_1$ OR x_2 OR x_3) AND (x_1 OR $\neg x_2$) AND ($\neg x_3$ OR $\neg x_4$) AND
  (x_2 OR x_3) AND ($\neg x_1$ OR $\neg x_2$)

Evaluating Boolean Formulas Given A Truth Assignment

Evaluating Boolean Formulas Given A Truth Assignment

- If x is assigned the value true then x evaluates to true and $\neg x$ to false

Evaluating Boolean Formulas Given A Truth Assignment

- If x is assigned the value true then x evaluates to true and $\neg x$ to false
- If x is assigned the value false then x evaluates to false and $\neg x$ to true

Evaluating Boolean Formulas Given A Truth Assignment

- If x is assigned the value true then x evaluates to true and $\neg x$ to false
- If x is assigned the value false then x evaluates to false and $\neg x$ to true
- F_1 AND F_2 AND…AND F_l evaluates to true if all the F_i evaluate to true. Otherwise it evaluates to false.

Evaluating Boolean Formulas Given A Truth Assignment

- If x is assigned the value true then x evaluates to true and $\neg x$ to false
- If x is assigned the value false then x evaluates to false and $\neg x$ to true
- F_1 AND F_2 AND…AND F_l evaluates to true if all the F_i evaluate to true. Otherwise it evaluates to false.
- F_1 OR F_2 OR…OR F_l evaluates to true any of the F_i evaluate to true. Otherwise it evaluates to false.

Satisfiable Formulas

Satisfiable Formulas

Consider a set X of Boolean variables and a formula *F over* this set of formulas (i.e. the only literals appearing in *F* are a variable in X or its negation).

Satisfiable Formulas

Consider a set X of Boolean variables and a formula *F over* this set of formulas (i.e. the only literals appearing in *F* are a variable in X or its negation).

An assignment of true or false to each variable in X *satisfies F* if *F* evaluates to true given this assignment.

Satisfiable Formulas

Consider a set X of Boolean variables and a formula *F over* this set of formulas (i.e. the only literals appearing in *F* are a variable in X or its negation).

An assignment of true or false to each variable in X *satisfies F* if *F* evaluates to true given this assignment.

F is *satisfiable* if there is a truth assignment which satisfies it.

Which of Our Formulas Are Satisfiable?

- F_1 $\neg x_1$
- F_2 x_1 *AND* x_2
- F_3 (x_1 OR x_2) AND $\neg x_1$ AND $\neg x_2$
- F_4 (x_1 OR $\neg x_2$) AND (x_2 OR $\neg x_3$) AND (x_3 OR $\neg x_4$) AND (x_4 OR $\neg x_1$)
- F_5 (x_1 OR $\neg x_2$) AND (x_2 OR $\neg x_3$) AND (x_3 OR $\neg x_4$) AND (x_4 OR $\neg x_1$)
  AND (x_1 OR x_2 OR x_3 OR x_4) AND ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$)
- F_5 ($\neg x_1$ OR x_2 OR x_3) AND (x_1 OR $\neg x_2$) AND ($\neg x_3$ OR $\neg x_4$) AND
  (x_2 OR x_3) AND ($\neg x_1$ OR $\neg x_2$)

Conjuctive Normal Form

- A Formula is in *conjuctive normal form*, or *CNF* if it is the AND of a set of formulas, called *clauses* each of which is the OR of a set of literals.

- F_1 $\neg x_1$ F_2 x_1 *AND* x_2 F_3 (x_1 OR x_2) AND $\neg x_1$ AND $\neg x_2$
- F_4 (x_1 OR $\neg x_2$) AND (x_2 OR $\neg x_3$) AND (x_3 OR $\neg x_4$) AND (x_4 OR $\neg x_1$)
- F_4 (x_1 OR $\neg x_2$) AND (x_2 OR $\neg x_3$) AND (x_3 OR $\neg x_4$) AND (x_4 OR $\neg x_1$)
  AND (x_1 OR x_2 OR x_3 OR x_4) AND ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$)
- F_5 ($\neg x_1$ OR x_2 OR x_3) AND (x_1 OR $\neg x_2$) AND ($\neg x_3$ OR $\neg x_4$) AND
  (x_2 OR x_3) AND ($\neg x_1$ OR $\neg x_2$)

SATISFIABILITY

- Input: A CNF Formula *F* given as a number *n* of variables, a set *X* of n Boolean variables, a number *l* of clauses, and for *i* between 1 and *l*, a list L_i of the literals in the ith clause C_i each of which is *x* or $\neg x$ for some *x* in *X*.

- Question: Is *F* satisfiable?

The Importance of Satisfiability’s Compexity.

The Importance of Satisfiability’s Compexity.

SATISFIABILITY is an NP-Complete Problem.

The Importance of Satisfiability’s Compexity.

SATISFIABILITY is an NP-Complete Problem. This means there is a million dollar prize for determining if there is a polynomial time algorithm to solve it.

The Importance of Satisfiability's Compexity.

SATISFIABILITY is an NP-Complete Problem. This means there is a million dollar prize for determining if there is a polynomial time algorithm to solve it. It also means that an efficient algorithm to solve it can be used to solve any problem for which there is an efficient algorithm.

Reducing SAT to 3-SAT

Reducing SAT to 3-SAT

- A k-SAT instance is an instance of SATISFIABILITY where each clause contains k literals.

Reducing SAT to 3-SAT

- A k-SAT instance is an instance of SATISFIABILITY where each clause contains k literals.
- We show now that there is a polynomial time algorithm which given a CNF formula *F* creates a CNF formula *F'* all of whose clauses contain exactly 3 literals, which is satisfiable if and only if *F* is Satisfiable.

Reducing SAT to 3-SAT

- A k-SAT instance is an instance of SATISFIABILITY where each clause contains k literals.
- We show now that there is a polynomial time algorithm which given a CNF formula *F* creates a CNF formula *F'* all of whose clauses contain exactly 3 literals, which is satisfiable if and only if *F* is Satisfiable.
- So determining if there is a polytime algorithm for 3-SAT is worth a million dollars, and very general problems can be encoded as 3-SAT instances.

Replacing A Short Clause

Suppose clause C_i has $l(i)$ literals: $y^i_1,\ldots,y^i_{l(i)}$

If $l(i)$=1 then we replace C_i by y^i_1 OR y^i_1 OR y^i_1

If $l(i)$=2 then we replace C_i by y^i_1 OR y^i_2 OR y^i_1

If $l(i)$=3 then we replace C_i by itself y^i_1 OR y^i_2 OR y^i_3

Replacing A Long Clause

Suppose clause C_i has $l(i)$ literals: $y^i_1,\ldots,y^i_{l(i)}$

If $l(i)>3$ then we replace C_i by $l(i)-2$ clauses using the

Variables $y^i_1,\ldots,y^i_{l(i)},nsy^i_2,\ldots,nsy^i_2$ as follows:

Clause D^i_1 is y^i_1 OR y^i_2 OR nsy^i_2

For j with $2\leq j \leq l(i)-3$, Clause D^i_j is $\neg nsy^i_j$ OR y^i_{j+1} OR nsy^i_{j+1}

Clause $D^i_{l(i)-2}$ is $\neg nsy^i_{lli)-2}$ OR $y^i_{l(i)-1}$ OR $y^i_{l(i)}$

Replacing A Formula

Replacing A Formula

- $(x_1 \text{ OR } \neg x_2)$

AND

$(x_1 \text{ OR } x_2 \text{ OR } x_3 \text{ OR } x_4 \text{ OR } \neg x_5)$

AND

$(\neg x_1 \text{ OR } \neg x_2 \text{ OR } \neg x_3 \text{ OR } \neg x_4 \text{ OR } x_5 \text{ OR } x_6)$

Replacing A Formula

- $(x_1 \text{ OR } \neg x_2)$

AND

$(x_1 \text{ OR } x_2 \text{ OR } x_3 \text{ OR } x_4 \text{ OR } \neg x_5)$ =>

AND

$(\neg x_1 \text{ OR } \neg x_2 \text{ OR } \neg x_3 \text{ OR } \neg x_4 \text{ OR } x_5 \text{ OR } x_6)$

Replacing A Formula

- $(x_1 \text{ OR } \neg x_2)$ $(x_1 \text{ OR } \neg x_2 \text{ OR } \neg x_2)$ AND

AND

$(x_1 \text{ OR } x_2 \text{ OR } x_3 \text{ OR } x_4 \text{ OR } \neg x_5)$ =>

AND

$(\neg x_1 \text{ OR } \neg x_2 \text{ OR } \neg x_3 \text{ OR } \neg x_4 \text{ OR } x_5 \text{ OR } x_6)$

Replacing A Formula

- $(x_1$ OR $\neg x_2)$

AND

$(x_1$ OR x_2 OR x_3 OR x_4 OR $\neg x_5)$

AND

$(\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR $x_6)$

$(x_1$ OR $\neg x_2$ OR $\neg x_2)$ AND

$(x_1$ OR x_2 OR $nsy^2{}_2)$ AND

=> $(\neg nsy^2{}_2$ OR x_3 OR $nsy^2{}_3)$ AND

$(\neg nsy^2{}_3$ OR x_4 OR $\neg x_5)$ AND

Replacing A Formula

- $(x_1$ OR $\neg x_2)$

AND

$(x_1$ OR x_2 OR x_3 OR x_4 OR $\neg x_5)$

AND

$(\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR $x_6)$

$(x_1$ OR $\neg x_2$ OR $\neg x_2)$ AND

$(x_1$ OR x_2 OR $nsy^2{}_2)$ AND

=> $(\neg nsy^2{}_2$ OR x_3 OR $nsy^2{}_3)$ AND

$(\neg nsy^2{}_3$ OR x_4 OR $\neg x_5)$ AND

$(\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2)$ AND

$(\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3)$ AND

$(\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4)$ AND

$(\neg nsy^3{}_4$ OR x_5 OR $x_6)$

Restricting A Satisfying Truth Assignment

- If for some clause C_i with l(i)>3,a truth assignment satisfies the clauses replacing C_i then its restriction to X satisfies C_i.

Proof by example: Suppose ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

($\neg nsy^3{}_4$ OR x_5 OR x_6)

Restricting A Satisfying Truth Assignment

- If for some clause C_i with l(i)>3,a truth assignment satisfies the clauses replacing C_i then its restriction to X satisfies C_i.

Proof by example: Suppose ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

($\neg nsy^3{}_4$ OR x_5 OR x_6)

Restricting A Satisfying Truth Assignment

Supposing ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

($\neg nsy^3{}_4$ OR x_5 OR x_6)

Restricting A Satisfying Truth Assignment

Supposing ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

F F by supposition

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

($\neg nsy^3{}_4$ OR x_5 OR x_6)

Restricting A Satisfying Truth Assignment

Supposing ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

F F T since replacement satisfied

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

($\neg nsy^3{}_4$ OR x_5 OR x_6)

Restricting A Satisfying Truth Assignment

Supposing ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

F F T

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

F since $nsy^3{}_2$ is true

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

($\neg nsy^3{}_4$ OR x_5 OR x_6)

Restricting A Satisfying Truth Assignment

Supposing ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

F F T

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

F F *by* supposition

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

($\neg nsy^3{}_4$ OR x_5 OR x_6)

Restricting A Satisfying Truth Assignment

Supposing ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

F F T

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

F F T since replacement is satisfied

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

($\neg nsy^3{}_4$ OR x_5 OR x_6)

Restricting A Satisfying Truth Assignment

Supposing ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

F F T

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

F F T

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

F since $nsy^3{}_3$ is true

($\neg nsy^3{}_4$ OR x_5 OR x_6)

Restricting A Satisfying Truth Assignment

Supposing ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

F F T

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

F F T

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

F F by supposition

($\neg nsy^3{}_4$ OR x_5 OR x_6)

Restricting A Satisfying Truth Assignment

Supposing ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

F F T

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

F F T

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

(replacement is satisfied) F F T

($\neg nsy^3{}_4$ OR x_5 OR x_6)

Restricting A Satisfying Truth Assignment

Supposing ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

F F T

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

F F T

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

F F T

($\neg nsy^3{}_4$ OR x_5 OR x_6)

since $nsy^3{}_4$ is true F

Restricting A Satisfying Truth Assignment

Supposing ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

F F T

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

F F T

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

F F T

($\neg nsy^3{}_4$ OR x_5 OR x_6)

since $nsy^3{}_4$ is true by supposition

F F F

Restricting A Satisfying Truth Assignment

Supposing ($\neg x_1$ OR $\neg x_2$ OR $\neg x_3$ OR $\neg x_4$ OR x_5 OR x_6) is not satisfied but the following replacement is:

($\neg x_1$ OR $\neg x_2$ OR $nsy^3{}_2$) AND

F F T

($\neg nsy^3{}_2$ OR $\neg x_3$ OR $nsy^3{}_3$) AND

F F T

($\neg nsy^3{}_3$ OR $\neg x_4$ OR $nsy^3{}_4$) AND

F F T

($\neg nsy^3{}_4$ OR x_5 OR x_6)

since $nsy^3{}_4$ is true by supposition contradiction F F F

Extending A Satisfying Decomposition from *F* to *F'*

For each clause C_i with $l(i)>3$ set nsy^i_j to the evaluation of:

$\neg y^i_1$ AND $\neg y^i_2$ AND …. AND $\neg y^i_j$

Get replacement clause structure:

F F T

F F T……

F T F

T ? F

T?F….

T ? ?

Our Reduction from SATISFIABILITY to 3-SAT Is Complete

Reducing 2-SAT to A Graph Theory Problem

The Implicaton Graph.

The Implicaton Graph.

For any truth assignment which satisifies l_1 OR l_2, we have both that $\neg l_1$ implies l_2 and that $\neg l_2$ implies l_1 . I.e. $\neg l_1 \Rightarrow l_2$ and $\neg l_2 \Rightarrow l_1$

The Implicaton Graph.

For any truth assignment which satisifies l_1 OR l_2, we have both that $\neg l_1$ implies l_2 and that $\neg l_2$ implies l_1 . I.e. $\neg l_1 \Rightarrow l_2$ and $\neg l_2 \Rightarrow l_1$

For any 2-SAT Formula *F* over a set X of Boolean Variables, the implication directed graph for *F, G(F)* satisfies:

The Implication Graph.

For any truth assignment which satisifies l_1 OR l_2, we have both that $\neg l_1$ implies l_2 and that $\neg l_2$ implies l_1 . I.e. $\neg l_1 \Rightarrow l_2$ and $\neg l_2 \Rightarrow l_1$

For any 2-SAT Formula *F* over a set X of Boolean Variables, the implication directed graph for *F, G(F)* satisfies:

$V(G(F)) = \{x, \neg x \mid x \in X\}$,

$E(G(F)) = \{(\neg l_1, l_2), (\neg l_2, l_1) \mid l_1$ OR l_2 is a clause of $F\}$

Satisfying Truth Assignment
And Good Labellings of The Implication Graph

Satisfying Truth Assignment And Good Labellings of The Implication Graph

By a *good labelling* of an implication graph we mean a labelling of the vertices of G(F) so that no literal has the same label as its negation and no edge has its tail labelled true and its head labelled false.

Satisfying Truth Assignment And Good Labellings of The Implication Graph

By a *good labelling* of an implication graph we mean a labelling of the vertices of G(F) so that no literal has the same label as its negation and no edge has its tail labelled true and its head labelled false.

By the construction of the implication graph we see that satisfying truth assignments for a formula are in 1-1 correspondence with the good labellings of its implication graph.

Satisfying Truth Assignment And Good Labellings of The Implication Graph

By a *good labelling* of an implication graph we mean a labelling of the vertices of G(F) so that no literal has the same label as its negation and no edge has its tail labelled true and its head labelled false.

By the construction of the implication graph we see that satisfying truth assignments for a formula are in 1-1 correspondence with the good labellings of its implication graph.

On Wednesday we will see an algorithm to determine if an implication graph has a good labelling and if so find one. Today we discuss some preliminaries.

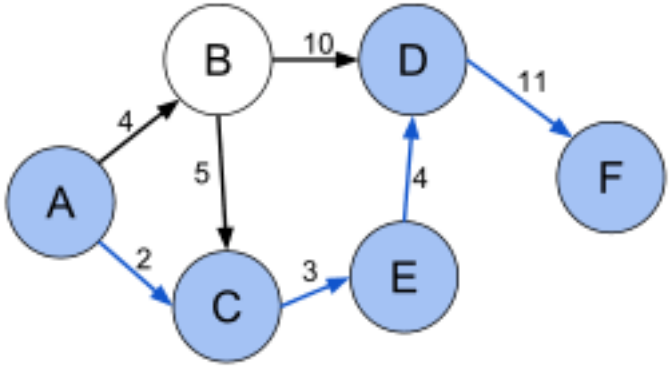
Reachability in Directed Graphs

Reachability in Directed Graphs

u is reachable from *v* if there is a directed path from *u* to *v.*

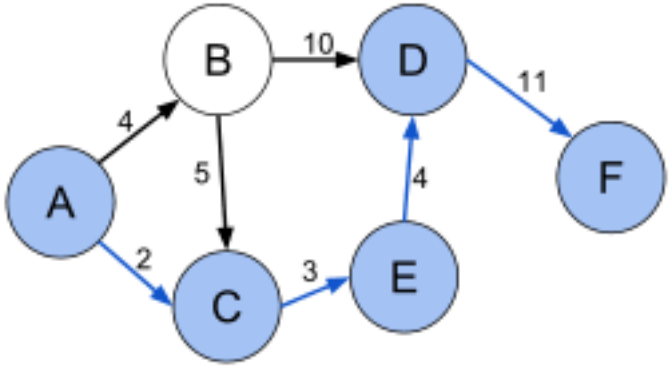
Reachability in Directed Graphs

u is reachable from *v* if there is a directed path from *u* to *v.*
Here F is reachable from A but A is reachable from no other vertex.



Reachability in Directed Graphs

u is reachable from *v* if there is a directed path from *u* to *v.*
Here F is reachable from A but A is reachable from no other vertex. Note reachability is transitive. If we can reach *v* from *u* and *w* from *v* we can reach *w* from *u.*



Strong Components

Strong Components

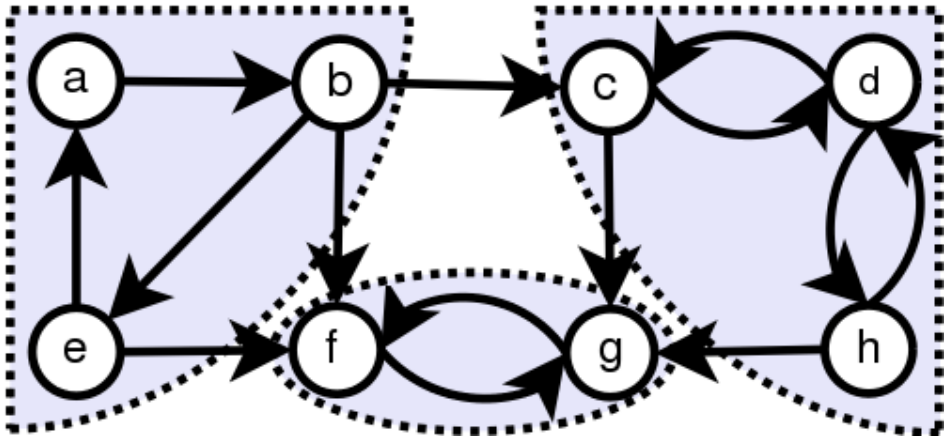
- A Strong Component is a maximal set S such that for every u and v in S, u is reachable from v and v is reachable from u.

Strong Components

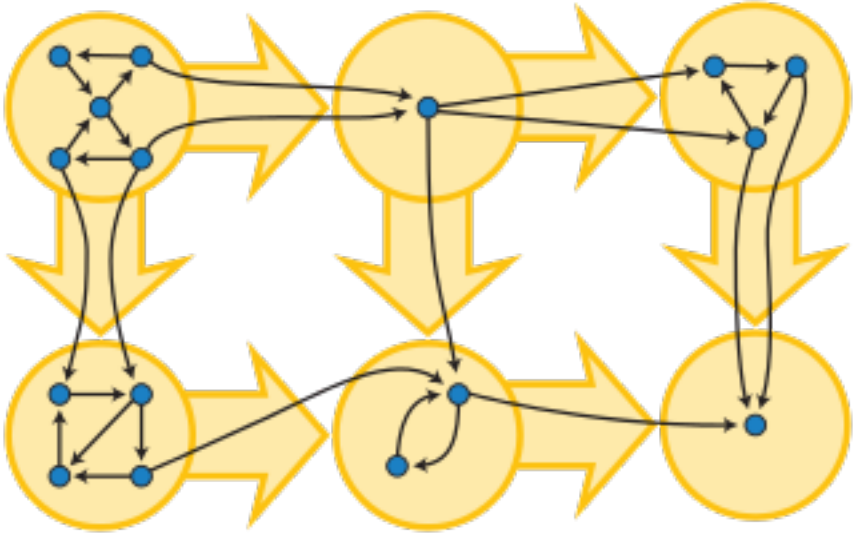
- A Strong Component is a maximal set S such that for every u and v in S, u is reachable from v and v is reachable from u.

- By the transitivity of reachability, strong component are disjoint.

Strong Components in A Directed Graph



Strong Components in A Second Directed Graph



Reachability And Good Labellings

Reachability And Good Labellings

If a vertex *v* is labelled true in a good labelling so is every vertex *w* reachable from it (Proof by induction on the length of the path from *v* to *w)*

Reachability And Good Labellings

If a vertex *v* is labelled true in a good labelling so is every vertex *w* reachable from it (Proof by induction on the length of the path from *v* to *w)*

$v=v_1 \to v_2 \to v_3 \to \ldots . v_l = w$

Reachability And Good Labellings

If a vertex *v* is labelled true in a good labelling so is every vertex *w* reachable from it (Proof by induction on the length of the path from *v* to *w)*

$v=v_1 \to v_2 \to v_3 \to \ldots v_l = w$

T

Reachability And Good Labellings

If a vertex *v* is labelled true in a good labelling so is every vertex *w* reachable from it (Proof by induction on the length of the path from *v* to *w)*

$v=v_1 \to v_2 \to v_3 \to \ldots v_l = w$

T

T

Reachability And Good Labellings

If a vertex *v* is labelled true in a good labelling so is every vertex *w* reachable from it (Proof by induction on the length of the path from *v* to *w)*

$v=v_1 \to v_2 \to v_3 \to \ldots v_l = w$

T

T

T

Reachability And Good Labellings

If a vertex *v* is labelled true in a good labelling so is every vertex *w* reachable from it (Proof by induction on the length of the path from *v* to *w)*

$v=v_1 \to v_2 \to v_3 \to \ldots v_l = w$

T

T

T

T

Strong Components And Good Labellings

Strong Components And Good Labellings

In a good labelling for G(F) all the vertices in some strong component S must receive the same label.

Strong Components And Good Labellings

In a good labelling for G(F) all the vertices in some strong component S must receive the same label.

So if some variable and its negation lie in the same strong component than there can be no good labelling.

Strong Components And Good Labellings

In a good labelling for G(F) all the vertices in some strong component S must receive the same label.

So if some variable and its negation lie in the same strong component than there can be no good labelling.

We shall see that otherwise there is a good labelling.

Pairing The Strong Components

Pairing The Strong Components

By construction there is an edge of G(F) from a literal x to a literal y if and only there is an edge from $\neg x$ to $\neg$ y.

Pairing The Strong Components

By construction there is an edge of G(F) from a literal x to a literal y if and only there is an edge from ¬x to ¬ y.

Again applying the induction on the length of the path, we see that z is reachable from x precisely if ¬z is reachable from ¬x.

Pairing The Strong Components

By construction there is an edge of G(F) from a literal x to a literal y if and only there is an edge from $\neg x$ to $\neg y$.

Again applying the induction on the length of the path, we see that z is reachable from x precisely if $\neg z$ is reachable from $\neg x$.

Applying this twice if we let S_x be the strong component containing x we see that $S_{\neg x} = \{\neg y \mid y \text{ is in } S_x\}$

Good Orders

Good Orders

An ordering of the vertices of G is *good* if the vertices of each strong component are consecutive and for every arc xy for which x and y are in different strong components, x comes before y in the order.

Good Orders

An ordering of the vertices of G is *good* if the vertices of each strong component are consecutive and for every arc *xy* for which *x* and *y* are in different strong components, *x* comes before *y* in the order.

We claim and will show on Wednesday, that every implication graph in which no literal and its negation lie in the same strong component has a good order.

Good Orders

An ordering of the vertices of G is *good* if the vertices of each strong component are consecutive and for every arc xy for which x and y are in different strong components, x comes before y in the order.

We claim and will show on Wednesday, that every implication graph in which no literal and its negation lie in the same strong component has a good order.

We showed on the next slide that given such an order for such a graph, setting x to be true if it came after $\neg x$ in the ordering and false otherwise yields a good labelling.

Good Labellings From Good Orders.

Good Labellings From Good Orders.

Consider our labelling strategy given a good order.

Good Labellings From Good Orders.

Consider our labelling strategy given a good order.

Since the strong components are consecutive either all of S_x appears before all of $S_{\neg x}$ or all of S_x appears after all of $S_{\neg x}$=$\{\neg y \mid y \text{ is in } S_x\}$.

Good Labellings From Good Orders.

Consider our labelling strategy given a good order.

Since the strong components are consecutive either all of S_x appears before all of $S_{\neg x}$ or all of S_x appears after all of $S_{\neg x}=\{\neg y \mid y \text{ is in } S_x\}$.

So all the vertices in a strong component receive the same label.

Good Labellings From Good Orders.

Consider our labelling strategy given a good order.

Since the strong components are consecutive either all of S_x appears before all of $S_{\neg x}$ or all of S_x appears after all of $S_{\neg x}=\{\neg y \mid y \text{ is in } S_x\}$.

So all the vertices in a strong component receive the same label.

Suppose *xy* is an edge with *x* labelled true and *y* labelled false. S_x and S_y are distinct.

Good Labellings From Good Orders.

Consider our labelling strategy given a good order.

Since the strong components are consecutive either all of S_x appears before all of $S_{\neg x}$ or all of S_x appears after all of $S_{\neg x}=\{\neg y \mid y \text{ is in } S_x\}$.

So all the vertices in a strong component receive the same label.

Suppose xy is an edge with x labelled true and y labelled false, S_x and S_y are distinct. Because of the pairing of components $S_{\neg x}$ and $S_{\neg y}$ are also distinct.

Good Labellings From Good Orders.

Consider our labelling strategy given a good order.

Since the strong components are consecutive either all of S_x appears before all of $S_{\neg x}$ or all of S_x appears after all of $S_{\neg x}=\{\neg y \mid y \text{ is in } S_x\}$.

So all the vertices in a strong component receive the same label.

Suppose *xy* is an edge with *x* labelled true and *y* labelled false, S_x and S_y are distinct. Because of the pairing of components $S_{\neg x}$ and $S_{\neg y}$ are also distinct. By construction $(\neg y, \neg x)$ is an edge of *G(F).*

Good Labellings From Good Orders.

Consider our labelling strategy given a good order.

Since the strong components are consecutive either all of S_x appears before all of $S_{\neg x}$ or all of S_x appears after all of $S_{\neg x}=\{\neg y \mid y \text{ is in } S_x\}$.

So all the vertices in a strong component receive the same label.

Suppose *xy* is an edge with *x* labelled true and *y* labelled false, S_x and S_y are distinct. Because of the pairing of components $S_{\neg x}$ and $S_{\neg y}$ are also distinct. By construction $(\neg y, \neg x)$ is an edge of *G(F).* Since this is a good order it follows that the vertices of $S_{\neg y}$ precede those of $S_{\neg x}$ in the order.

Good Labellings From Good Orders.

Consider our labelling strategy given a good order.

Since the strong components are consecutive either all of S_x appears before all of $S_{\neg x}$ or all of S_x appears after all of $S_{\neg x}=\{\neg y \mid y \text{ is in } S_x\}$.

So all the vertices in a strong component receive the same label.

Suppose *xy* is an edge with *x* labelled true and *y* labelled false, S_x and S_y are distinct. Because of the pairing of components $S_{\neg x}$ and $S_{\neg y}$ are also distinct. By construction $(\neg y, \neg x)$ is an edge of *G(F).* Since this is a good order it follows that the vertices of $S_{\neg y}$ precede those of $S_{\neg x}$ in the order. Because *x* is labelled true, the vertices of $S_{\neg x}$ precede those of S_x in the order.

Good Labellings From Good Orders.

Consider our labelling strategy given a good order.

Since the strong components are consecutive either all of S_x appears before all of $S_{\neg x}$ or all of S_x appears after all of $S_{\neg x}=\{\neg y \mid y \text{ is in } S_x\}$.

So all the vertices in a strong component receive the same label.

Suppose *xy* is an edge with *x* labelled true and *y* labelled false, S_x and S_y are distinct. Because of the pairing of components $S_{\neg x}$ and $S_{\neg y}$ are also distinct. By construction $(\neg y, \neg x)$ is an edge of *G(F).* Since this is a good order it follows that the vertices of $S_{\neg y}$ precede those of $S_{\neg x}$ in the order. Because *x* is labelled true, the vertices of $S_{\neg x}$ precede those of S_x in the order. Hence the vertices of $S_{\neg y}$ precede those of S_y in the order, and *y* is labelled true, a contradiction.