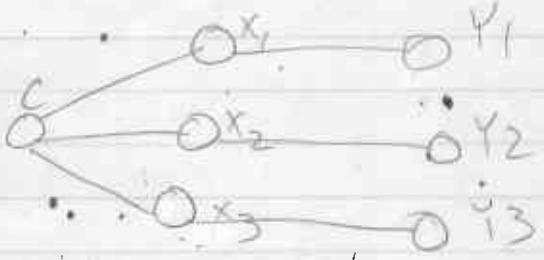
Group 1 Problems 1 & 10

Reachability Algorithm

1. Let S be a set of nodes. Initially set S = {start_node}
2. A node is called marked if it has ever appeared in S.
3. Repeat until either end_node is marked or S is empty.
   a) choose a node i in S and remove it from S. For each edge (i, j) out of i in E, if node j is unmarked, mark it and add it to S.
4. If end_node is marked then answer 'YES', otherwise answer 'NO.'



Reachability of node Y_3 from Y_1

$S = \{Y_1\}$

Choose node Y_1 from S and remove it. Mark and add X_1 to S.

$S = \{X_1\}$

Choose node X_1 from S and remove it. Mark and add C to S.

$S = \{C\}$

Choose node C from S and remove it. Mark it and add X_1 & X_2 to S.

$S = \{X_2, X_3\}$

Choose node X_2 from S and remove it. Mark it and add Y_2 to S

$S = \{Y_2, X_3\}$

Choose Y_2 from S and remove it. Node Y_2 connected to X_2 that already marked.

$S = \{X_3\}$

Choose node X_3 from S and remove it. Mark and add Y_3 to S

Y_3 destination node has been marked. Y_1 is connected to Y_3

CS 254 - Problem 2 of Practice Test

Joel C. Frank

October 3, 2006

1 Problem 2

Show that $n \log n = O(\sum_{i=1}^{n} \log i)$

Notice that:

$$\frac{n}{2} \log \frac{n}{2} = \sum_{i=\frac{n}{2}}^{n} \log \frac{n}{2} \leq \sum_{i=\frac{n}{2}}^{n} \log i \leq \sum_{i=1}^{n} \log i$$

Therefore, let $n_0 = 3$ such that:

$$\forall n \geq n_0 \quad n \log n \leq c * \frac{n}{2} \log \frac{n}{2}$$

It can be seen that if $c \leq 2$ this condition does not hold. Therefore, let $c = 3$.

Gmail talk BETA

Chris Pollett <cpollett@gmail.com>

problem 3

Heather Kwong <heatherkwong@gmail.com> **Tue, Oct 3, 2006 at 11:51 AM**
To: cpollett@gmail.com

Hi Professor Pollet,

My group was assigned problem 3, but this is the only thing we wrote down for solutions last night.

3. Explain how to reduce in polynomial time the problem of bipartite matching to the problem of network flow. Formulate a notion of tripartite matching. Is this problem also reducible to network flow?

To reduce in polynomial time the problem of bipartite matching to network flow, construct a network with nodes U U V U {s, t}, where s is the source and t is the sink and all capabilities equal to one. Now, the graph has a matching iff the resulting network flow has a flow of value n. Since we have reduced this problem to a MAX FLOW problem, we know we can solve this in polynomial time. (see page 12) .

Since tripartite matching is NP complete, this is not reducible to a network flow.

Tripartite Matching Problem.

Let $S \subseteq U \times V \times W$ where $|U| = |V| = |W| = n$

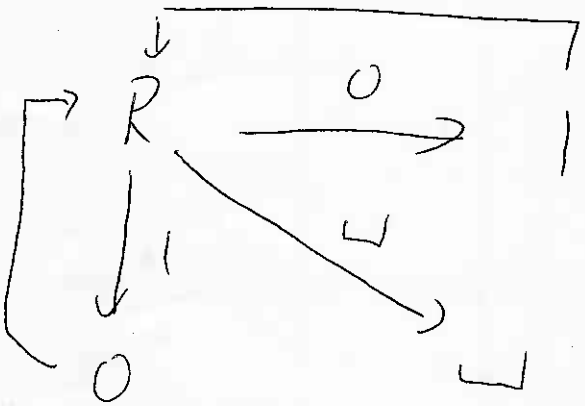
The problem is to find a set of n triples

$\{(u_1, v_1, w_1), \ldots (u_n, v_n, w_n)\}$ s.t.

if $i \neq j$ then $u_i \neq u_j$, $v_i \neq v_j$, & $w_i \neq w_j$

(4)

$p \in K$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
s	0	$(s, 1, \rightarrow)$
s	1	$(s, 0, \rightarrow)$
s	$\sqcup$	$(h, \sqcup, -)$
s	$\triangleright$	$(s, \triangleright, \rightarrow)$



5 On input $x = x_1 \cdots x_n$

(1) (a) Scan the input left to right.
At each square nondeterministically choose to overwrite a symbol x_i with the underscored version $\underline{x}_i$ or to not overwrite it.
(b) rewind

(2) Scan to the 1st underscore
(a) copy from the 1st underscore to the second underscore to tape 2.
If no second underscore before ⊔ seen halt no.
(b) copy from the 2nd underscore to the 1st space to tape 3.
(c) rewind all tapes.

(3) Scan square by square matching three tapes against each other until ~~(a) the tapes~~ tape 1 is on 1st underscore & tape 2 & tape 3 are on spaces.

(a) If ever don't match halt no.
(b) Otherwise, halt yes.

6

Simulating RAMs on TMs

Theorem If L is recognized by a RAM in time f(n) then it is in TIME($O(f(n)^3)$).

Proof: Let P be a RAM program. We will simulate it by a seven tape machine. The first tape will be used to hold the input string and it will never be overwritten. The second tape will be used to represent the content of all the registers. This will be represented by a sequence of semicolon separated pairs i, v. Here i says the register (which may be 0) and v says its value. When a register is updated we copy the pair to the end of our sequence, update the value, then X over the old value. An example sequence might be: 0, 101; XXX 1, 10; _

The runtime for results comes because this tape can be shown (see book) to grow as $O(f(n)^2)$.

The states of M are split into m groups where m in the number of instructions in P. Each group implements one instruction. Tape 3 is used to store the current program counter. This is initially 1. At the start of the simulation tape 2 is initialize to the input configuration of a register machine based on the contents of the input tape. Thereafter, at the start of simulating an instruction. The program counter is read and the start state of the group of states of M for that instruction is entered. (see next slide)

Proof Continued

An instruction is then processed, tape 2 is updated, and the program counter on tape 3 is updated, then the next step can be simulated and so on. Most instructions are reasonably straightforward to carry out: To process an instruction that uses indirect addressing of the form (j), tape 4 is used to store the value k of the register j so that we can then go access register k on tape 2. For operations like Add and Sub, tapes 5 and 6 are used to store the operands and tape 7 is used to compute the result. If the RAM halts, the contents of register 0 (the accumulator) are looked up on tape 2, and the TM accepts if the value is positive.

(7) If L is recursively enumerable then let M be a TM w/c shows this.

So $x \in L \Rightarrow$ M halts yes on x

$x \notin L \Rightarrow$ M runs forever.

Let f be the Turing computable function w/ con input a string x copies x to a second tape. Then writes the string description of M to the 1st tape followed by a ',' . Finally, f copies x from the second tape to the squares after this comma.

So diagrammatically

$$x \xrightarrow{f} \langle M, x \rangle$$

We note $x \in L$ iff M on x halts iff $\langle M, x \rangle \in H$

So we have shown L reduces to H where L was an arbitrary r.e. language.

8/ $P = \{\langle M \rangle \mid L(M)$ consist of binary string of even # of 1$\}$

Using Rice's theorem:

- There are some $\langle M_1 \rangle$ s.t. $L(M_1) = \{w \mid w$ has an even # 1's$\}$ and there are some $\langle M_2 \rangle$ s.t. $L(M_2)$ is not this language.
- For any $M_1 \& M_2$ s.t. $L(M_1) = L(M_2)$ either $\in L(M_1) = L(M_2) = L$, or $L(M_1) = L(M_2) \neq L$. So not both Rice's properties are satisfied.

$\Rightarrow$ P is undecidable

Without Rice theorem: Assume P is decidable by M_P. Let $\langle M, x \rangle$ be an input to the halting problem. Given this input we can build a machine that could be built w/c outputs the code of the following machine:

Let M' = "on input y

1. Write x to a new tape
2. Simulate M on input x
3. If M halts, check if y has even # of 1's

If yes, halt YES
else halt NO.

Now $\langle M, x \rangle \in H$ iff $M''(\langle M, x \rangle) = \langle M' \rangle \in P$

So if P is decidable by M_P then the procedure

1. run M'' on the input
2. run M_P on the result output yes if M_P halts yes otherwise output no.

decides the halting problem. But we know this is impossible. $\therefore M_P$ can't exist $\therefore P$ undecidable

9)

X_1	X_2	X_3	P
F	F	F	F
F	F	T	T
F	T	F	T
F	T	T	F
T	F	F	T
T	F	T	F
T	T	F	F
T	T	T	T

CNF

$(\neg X_1 \vee \neg X_2 \vee \neg X_3) \wedge$

$(\neg X_1 \vee X_2 \vee X_3) \wedge$

$(X_1 \vee \neg X_2 \vee X_3) \wedge$

$(X_1 \vee X_2 \vee \neg X_3)$

DNF

$(\neg X_1 \wedge \neg X_2 \wedge X_3) \vee$

$(\neg X_1 \wedge X_2 \wedge \neg X_3) \vee$

$(X_1 \wedge \neg X_2 \wedge \neg X_3) \vee$

$(X_1 \wedge X_2 \wedge X_3)$

Let $m = \frac{2^n}{2n}$. We upper bound the number of possible circuits using c connectives with m or fewer gates as $((n+2+c)\cdot m^2)^m$ using same analysis as in the $c=3$ (AND, OR, NOT) case. Taking log of this yields:

$$m \log((n+2+c)\cdot m^2)$$

$$= \frac{2^n}{2n}\left[\log(n+2+c) + 2\log\left(\frac{2^n}{2n}\right)\right]$$

$$= \frac{2^n}{2n}\left[2n + \log(n+2+c) - 2\log 2n\right]$$

$$= 2^n\left[1 - \log\left[\frac{(2n)^2}{n+2+c}\right]/2n\right]$$

$$= 2^n\left[1 + \frac{\log(n+2+c) - \log(2n)^2}{2n}\right]$$

there are 2^{2^d} many d-ary gates

So if $d = k\log n$

then $\log(n+2+2^{2^{k\log n}}) > 2^{k\log n} = n^k$

giving the above

$$\geq 2^n\left[1 + \frac{n^k - \log(2n)^2}{2n}\right]$$

So if $k \geq 1$ the term ≥ 1 in square brackets grows as n grow and the whole expression $< 2^n$ and the analysis fails