PERFECT EDGE ® MICRO PERFORATED FOR CLEAN TEAR-OUTS

CS 374

Made in USA

Nørcom Inc. Griffin, GA 30224
Item #77070

1 SUBJECT

70 Sheets

WIDE RULED

Justin Yang <justiny2@illinois.edu>
Spring 2017

CS/ECE 374 Introduction to Algorithms & Models of Computation
Fall 2017

Lecture AL1
Professor Sariel Har-Peled (sariel@)
TR 11 am - 12:15 pm
1002 ECEB

Lab ADF
TA Xilin Yu (xilinyu2@)
WF 1-1:50 pm
1105 Siebel

CS 374 T 29 Aug

Algorithms & Models of Computation

Course webpage, Piazza

HW 0 due 6 Sept (Wed) at 10 am

Discussion sections W/F

Automata, algorithms, undecidibility/NP-completeness

Multiplication: naïve ($O(n^2)$)

Post Correspondence problem (dominoes matching top & bottom)

Halting problem ← Artin theorem

Strings

alphabet - finite set of symbols

ε is empty string

word is a finite string

induction/recursion

substring, prefix, suffix

$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$ - the set of all finite length strings

canonical order & countability of strings

countably infinite iff bijection b/t natural #s and set

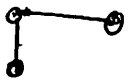
Is $\Sigma^* \times \Sigma^* = \{(x, y)\}$ countably infinite?

Languages

Power set of Σ^* is not countable (Cantor)

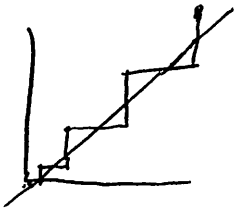
CS 374 HW 0

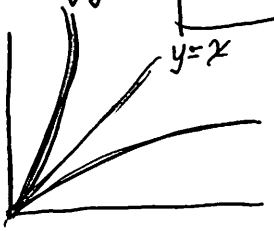
1. $G = (V, E)$ an undirected graph.
   $V = 2$.
   Either edge or not. $\Rightarrow$ equal degree.



always use strong induction for 374

4. $x_1 \ldots x_n$
   Pigeonhole.
3. a) Trivial
   b) $O(\log x \log y)$





374 Lab 1a

Prove that $|w| = |w^R|$ $\forall w$.

We prove this by induction on $|w|$.

Base case

$|w| = 0$

$w = \varepsilon = w^R$

$w^R = 0 \quad |w| = 0 = |w^R|$

Induction Hypothesis:

Assume for all strings z, s.t. $|z| < k$, that $|z| = |z^R|$.

Induction step:

Consider a string x, s.t. $|x| = k$

we need to show that $|x| = |x^R|$.

$x = aw$

$|x^R| = |(aw)^R|$

$= |w^R a|$ — by def of R

$= |w^R| + |a|$ — by Lemma 2

$= |w^R| + 1$ — by def of 1.1

$= |w| + 1$ — by IH

$= x$ — by def of 1.1

3, Assume the result of 2.

Base case $|w| = 0 \Rightarrow w = \varepsilon$ ✓

Induct on $|w|$. Assume $\forall w$ s.t. $|w| < k$, $(wR)R = w$.

Then for any w s.t. $|w| = k$, $w = ax$, where $|a| = 1$.

$\Rightarrow wR = (ax)R = xRaR$

$\Rightarrow (wR)R = (xRaR)R = (aR)R \cdot (xR)R$

$= ax.$

---

2. WTS $(w \cdot z)R = zR \cdot wR$

By induction on the length of w, $|w|$.

Base case $|w| = 0$. $\Rightarrow w = \varepsilon$

Hence $(w \cdot z)R = (\varepsilon \cdot z)R = zR$

$= zR \cdot \varepsilon = zR \cdot wR$.

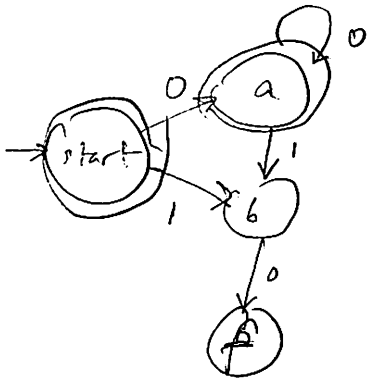
Assume for all strings w, $|w| < k$ the claim.

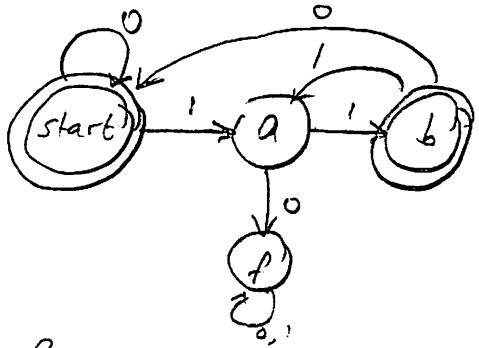
Let $w = ax$ for a a character.

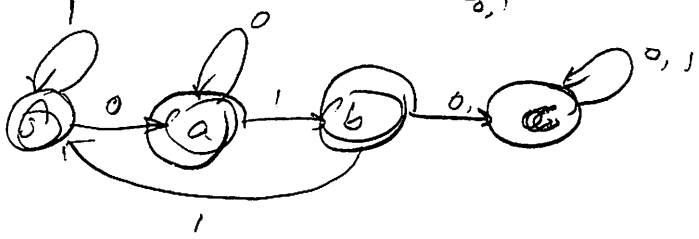
$(w \cdot z)R = (a \cdot x \cdot z)R = (x \cdot (az))R$

$= (xz)R \cdot a = zRxR aR = zR(ax)R = zRwR$

□.







$(1* \cancel{000} 0* 1*)*$

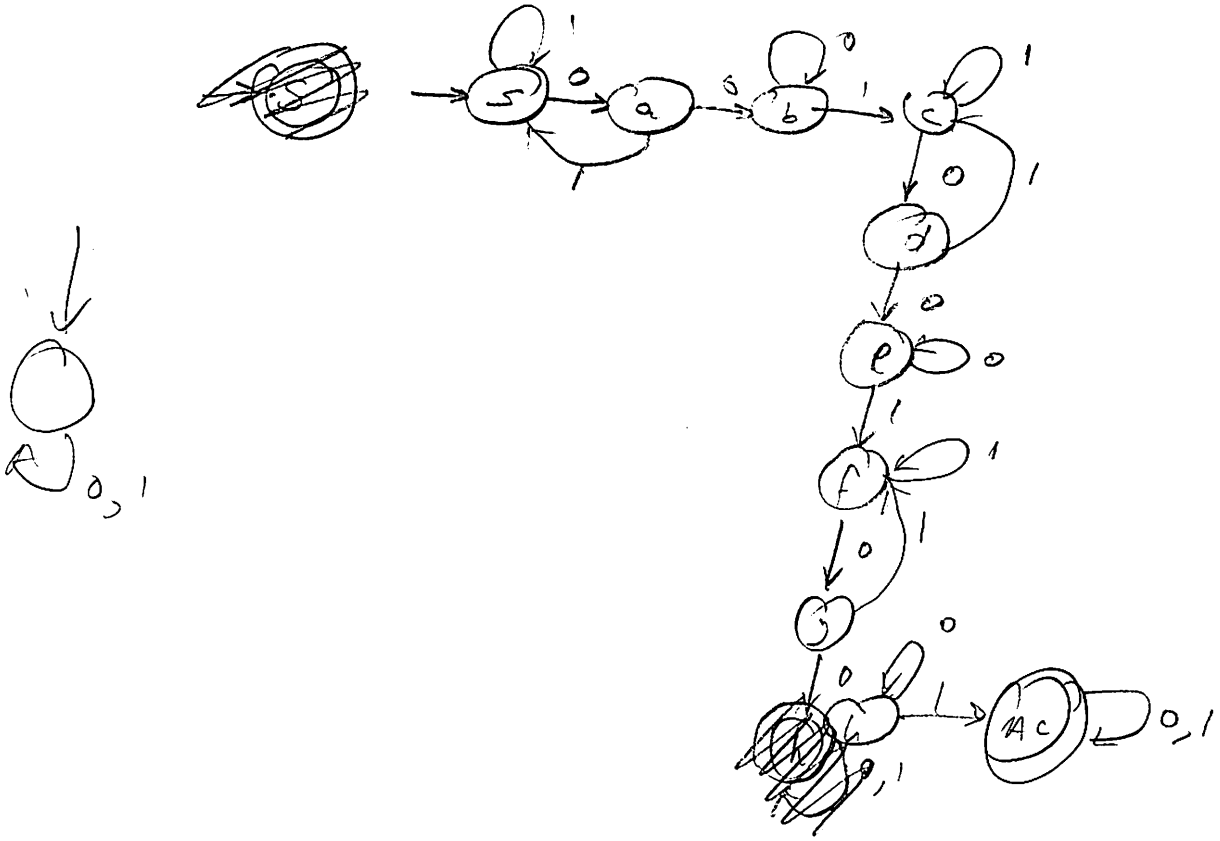
~~1 0 1 0~~

$1* 00*11$

~~(ε+1)~~ $(0* 11* 0*)*$ ~~(ε+1)~~

$1*$ $1*$

0 11 0 11 000



cs 374 R 31 Aug

Regular Languages

ε = empty word = ""

$\emptyset = \{\}$

$L = \{a\}$, $L^2 = \{aa\}$, $L^i = \{\underbrace{aa\ldots a}_{i}\}$, $a \in \Sigma$

$L^* = \{\varepsilon, a, aa, aaa, \ldots\}$

L^* always contains the empty word

Kleene star, $*$

$\{w \mid w$ contains "cs 374" as a substring$\}$

$\Sigma^* \{cs374\} \Sigma^* \subseteq \Sigma^*$

- Regular Expressions - pattern defines valid strings
- drop the braces $\{\}$ for language denotation

$(r_1 + r_2)$ denotes the language $R_1 \cup R_2$

$(r_1 r_2)$ " " $R_1 R_2$

$(r_1)^*$ " " R_1^*

$(a+b)(b+c) = ab + ac + bb + bc$

same priority as order of operations in algebra ↔ regex ↔ algebra mapping

regular expressions are not unique

$a^+ = a^* \cdot \{\varepsilon\}$

$0^* + (0^*10^*10^*10^*)^*$ all binary strings with a multiple of three 1s (and ε)

$\emptyset 0 = \emptyset$

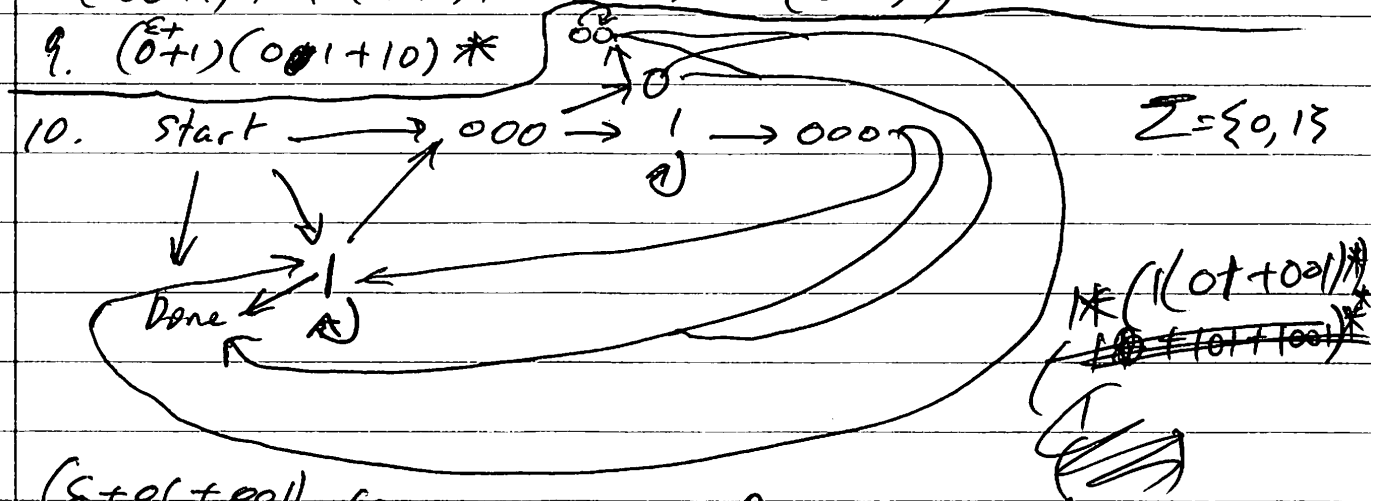
$(\varepsilon + 1)(01)(\varepsilon + 0)$: binary strings that alternate b/t 0 & 1

$(\varepsilon + 0)(1 + 10)^*$: binary strings where 0s don't appear consecutively (ie, no 00)

odd 1s: $0^*10^*(0^*10^*10^*)^*$

Discussion 016 F 1 Sept. SF Xilin 1pm

1. $(0+1)^* \; 000(0+1)^*$
2. ~~$(\varepsilon+0+00)1^*((\varepsilon+0+00)1^*)^*$~~
3. $(1^*(0000^*)1^*)^*$
4. $[(\varepsilon+0+00)1)^* \; 0^*$
5. $(0+1)^* \; 0(0+1)^* \; 0(0+1)^* \; 0 \; (0+1)^*$
6. $\varepsilon+(0+1)+(0+1)(0+1)+$ ~~$(0+1)^3$~~ $(001+010+011+100+101+110+111)+(0+1)(0+1)(0+1)(0+1)^*$
7. ~~$(0+1)$~~ $(01+10)^*(\varepsilon+1)$
8. $((0+1)^* \; 0(0+1)^* \; 0(0+1)^* \; 1 \; (0+1)^*) +$
   $((0+1)^* \; 0(0+1)^* \; 1 \; (0+1)^* \; 0(0+1)^*) +$
   $((0+1)^* \; 1 \; (0+1)^* \; 0(0+1)^* \; 0(0+1)^*)$
9. $(\varepsilon+0+1)(0+1+10)^*$



$(\varepsilon+01+001)((000(1^*000+0(00)^*))^*1^*(1(01+001)^*)^*)^*(\varepsilon+0+00)+(01+001)^*$

$(\varepsilon+01+001)((000(1^*000+0(00)^*))^*1^*(1(01+001)^*)^*)^*(\varepsilon+0+00)$

T 5 Sept Lecture

DFAs

transition function

Wed 6 Sept: Discussion 02a

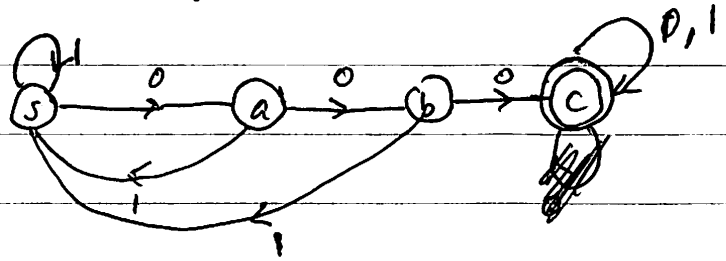
1pm
Abishek

abhishekn kar.net/courses/374

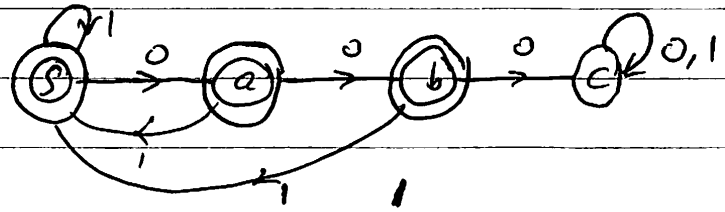
5 components to a DFA

- Input alphabet Σ
- States Q
- Transition function $D: Q \times \Sigma \to Q$
- Start state, s
- Accepting states $A \in Q$

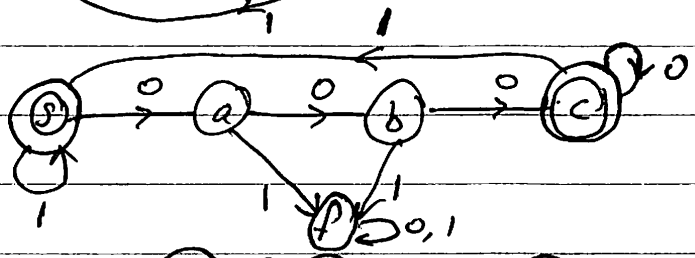
1.



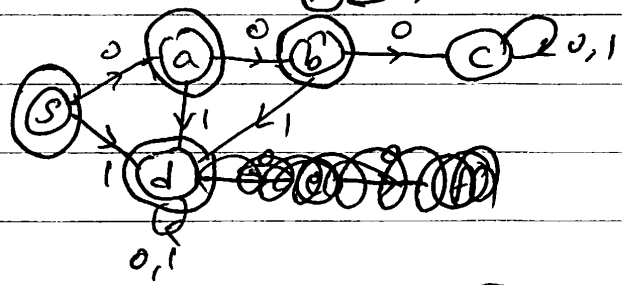
2.



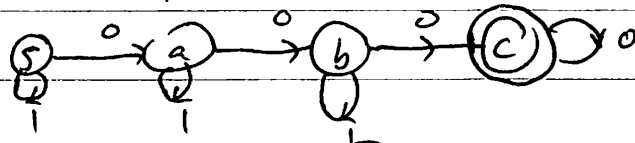
3.



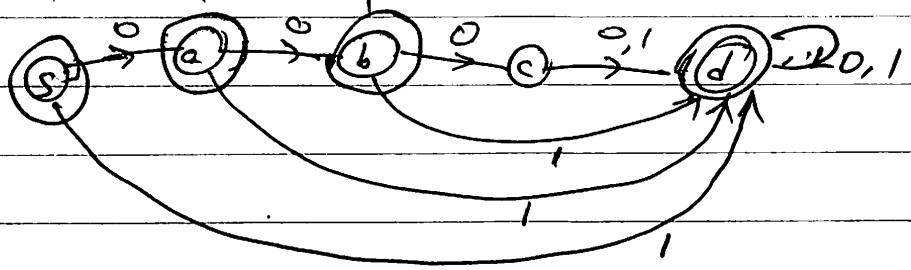
4.



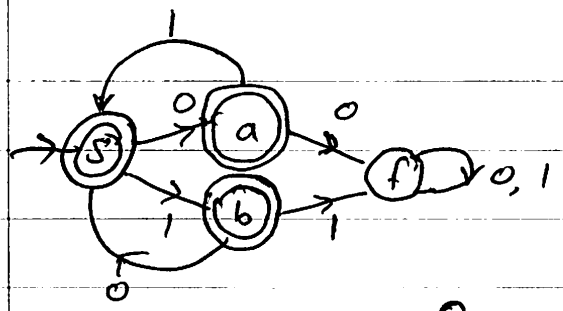
5



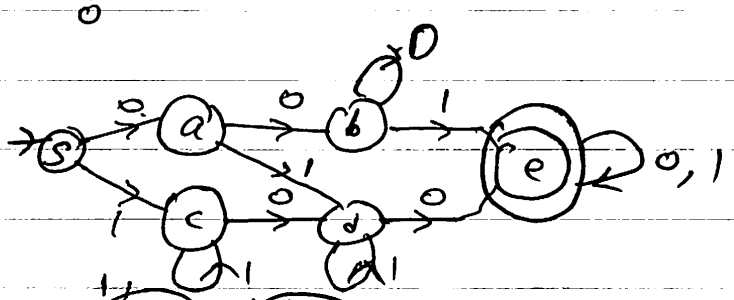
6



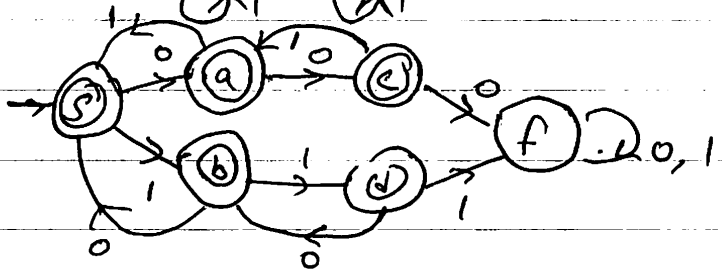
8



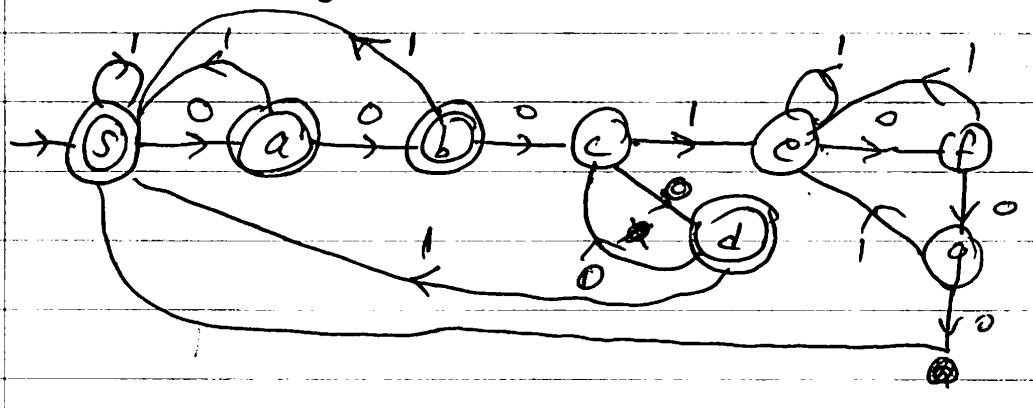
9



10



11



R 7 sept lecture 4

Non-Deterministic Finite Automata (NFAs)

set of states → power set of states

convert DFA → NFA: essentially no work:

change transition function to take a set of states (only one state in the case of DFA) instead of a single state.

ε transition allows for only one accept state.

Tues 12 Sept Lecture

NFAs cont, closure properties of Regular Languages

NFA needs to make decisions

δ^*

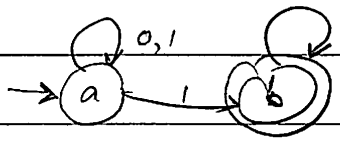
NFA $N = (Q, \Sigma, \delta, s, A)$

DFA $M = (Q', \Sigma, \delta', s', A')$

$Q' = P(Q)$ (includes $\emptyset = \{\}$) — completely invalid input

$s' = \epsilon\text{-reach}(s) = \delta^*(s, \epsilon)$

$\delta'(X, a) = \bigcup_{q \in X} \delta^*(q, a)$ for each $X \subseteq Q$, $a \in \Sigma$



DFA contains states that are not reachable from state
⇒ build incrementally from state to avoid.
Incremental algorithm

Closure Properties of Regular Languages
Negation easy w/ DFA. swap accepting states
Regular languages closed under many operations!

Regex to NFA (GNFA)
Convert NFA → generalized NFA (transitions are regex)
Create explicit init and AC states

Thurs 14 Sept Lecture 6

Proving Non-regularity

Thm Languages accepted by DFAs, NFAs, and regular expressions are the same.

Prove non-regularity.

Pf By contradiction. Show states are infinite.

Assume FTSOC $\delta^*(s, x) = \delta^*(s, y)$ for strings x, y.

Fooling set.

Infinite fooling sets

sliding window of the last k bits of the input

Non-regularity via closure properties

pumping lemma

Myhill-Nerode Theorem

Indistinguishability

Tues 19 Sept. lecture

Canonical non-CFL (context free language)

Counting lemma

Parse Trees or Derivation Trees

~~Inherent~~ Inherently ambiguous

$L = \{a^n b^m c^k \mid n = m \text{ or } m = k\}$

context-free languages are not closed under intersection

Prove $L = L(G)$ by $L \subseteq L(G)$ and $L(G) \subseteq L$

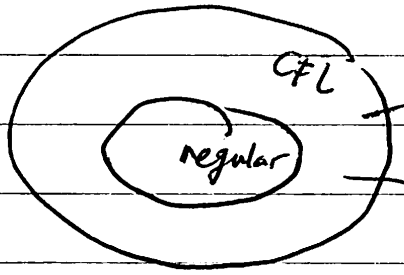
Normal forms, ~~Kuroda, Beibutun forms~~, Chomsky, Greibach

Pushdown Automata (PDA): an NFA coupled with a stack

Wed 20 Sept Discussion

A CFG $G = (V, T, P, S)$

- V: nonterminal symbols
- T: terminal symbols
- P: productions
- S: start



$\{1^{2n}0^n\}$ $\quad S \to w \Rightarrow \delta$

$\{1^m 0^n \mid m \neq n\}$ $\quad S \xrightarrow{+} w \Rightarrow \delta^+$

$S \to 1 \mid 11 \mid S \mid \epsilon \mid SS \mid SA$

$A \to$

Apply rules until you only have terminal symbols

1. CFG for $\{0^{2n}1^n \mid n \geq 0\}$

~~$S \to 00S1 \mid \epsilon$~~

Q. $\{0^m 1^n \mid m \neq 2n\}$

~~$S \to 00S1$~~

$m < 2n \mid m > 2n$

$S \to \epsilon \mid 00$

$S \to 00S1 \mid \epsilon$

Hint: Cases for relation of m & $2n$ when $m \neq 2n$

Hint: Think of canceling 0's & 1's in the same fashion

0^+1^+

- $m > 2n$
- $m < 2n$

00001111

$S \to 00S1 \mid A \mid B \mid C$ $\quad$ U

$A \to A1 \mid 1$ $\quad 1^+$

$B \to 0B \mid 0$ $\quad 0^+$

$C \to 0A \mid 0$ $\quad 01^+$

if $S \xrightarrow{+} w$

$w = \underbrace{000\ldots0}_{2k} u \underbrace{11\ldots1}_{k}$ $\quad u = (0^+ + 1^+ + 01^+)$

3 $\{0,1\}^* \setminus \{0^{2n}1^n \mid n \geq 0\}$

e.g. 0^m1^n, $m \neq 2n$

or 1010

or 01110

0^*1^*

$S_1 \{0^m1^n \mid m \neq 2n\}$

Hint: L is the union of a CFL and a regular language

↳ $(0+1)^*10(0+1)^*$

$S_0 \to S_1 | S_2$

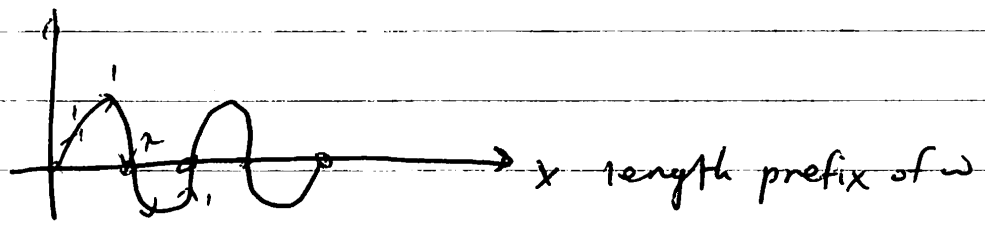
$S_2 \to 0S | 1S | 10D$

$D \to D0 | D1 | \epsilon$

CFL not closed under complement

4. $\{w \in \{0,1\}^* \mid \#(0,w) = 2\#(1,w)\}$

$\Delta = \#(0,w) - 2\#(1,w)$



$S \to SS | 0051 | 05150 | 1500$

Thurs 21 September Lecture

Turing machines

Halting problem & undecidibility

Binary search: divide interval into ≈ equal size — recurse on one side/subproblem
~ log n

1. $A[1 \ldots n]$ s.t. $A[1] < A[2] < \ldots < A[n]$

1.A. ~~start at~~
if $n=1$: — log n time
  ~~if~~ $A[1]=1$
  if return 1
  return False.
mid ← $\lfloor \frac{n}{2} \rfloor$
if ~~or~~ $A[mid] = mid$
  return mid
if $A[mid] > mid$:
  ret Recurse($A[1 \ldots mid-1]$)
ret Recurse ($A[mid+1 \ldots n]$)

1.B. $A[1] > 0$
If $A[1] \geq 1$
  return null
return 1
constant-time

2. $A[1 \ldots n]$
$A[1] > A[2]$ & $A[n-1] \leq A[n]$
Def local min := $A[x]$ s.t. $A[x-1] \geq A[x] \leq A[x+1]$

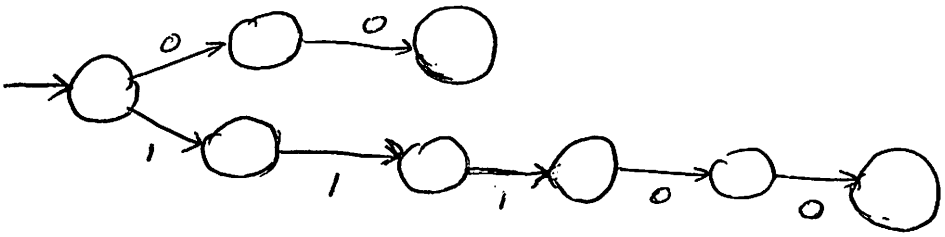
localmin($A[1 \ldots n]$):
  if $n=3$ { if hi-lo ≥ 100 Brute force.
  return $A[2]$.
← mid ← $\lfloor \frac{n}{2} \rfloor$
  if $A[mid] < A[mid+1]$:
  ret localmin($A[1 \ldots mid+1]$)
  ret localmin ($A[mid \ldots n]$)

3. $A[1, \ldots, n]$, $B[1, \ldots, n]$
median A ∪ B
Hint: compare median of A with median of B

$A[mid]=a$ & $B[mid]$
$= a > b$ $= b$

~~if~~ median (A, B):
  mid ← $\lfloor \frac{n}{2} \rfloor$ if ($n=1$) → ret min($A[1], B[1]$).
if ($A[mid] = B[mid]$) ret $A[mid]$
  ~~ret~~ if ($A[mid] > B[mid]$) ret ($A[1 \ldots mid]$, $B[mid \ldots n]$) else ret ($A[mid \ldots n]$, $B[1 \ldots$

$M = (Q, \Sigma, \delta, s, A)$ Strong induction



Wed 11 Oct Lab CS 374 backtracking

steps:

1. Define the recursive function

$f(x, y, z) = \ldots$

min # of para

min values of each para $(k_1, k_2, \ldots, k_t)$

2. Memoization

Goal: turn recursive algo to iterative one

Method: Fill cache table in a order

s.t. you can read off values from table

3. Running time analysis

(A) Calculate # distinct recursive calls

= # entries in cache table

$$\prod_{i=1}^{t} K_i$$

(B) Calculate how much time it takes

↓ for recursive function to calculate one value assuming other values it depends on are accessible by O(1)

O(1)

O(n)

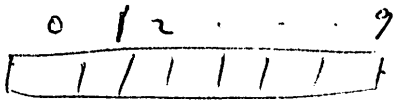
Total running time: (A) * (B)

1. $A[1, \ldots, n]$

length of longest strictly increasing subsequence

e.g. (3 1 4 1 5 9 . . .

parameters $0, \ldots, 9$




```
Subseq(A[1, ..., n])
  digits[0] ← 0s
  for i = 1 → n:
    ~~for d = 0 → 9:~~
      ~~New ← digits[A[i] - 1]~~
    ~~digits[A[i]] = 1 + max(A[0 ... A[i]])~~
    for d = A[i] → 9:
      digits[d] ← digits[A[i] - 1]
  return max(digits[0 ... 9])
```

---

① parameters

$OPT[i]$ = length of longest increasing subsequence of $A[i \ldots n]$ that starts with $A[i]$

$$OPT[i] = \begin{cases} 1 \text{ if } A[j] \le A[i] \ \forall\, j > 1 \\ 1 + \max_{\substack{j>i \\ A[j]>A[i]}} \{OPT[j]\} \quad \text{o.w.} \end{cases}$$

$\max \phi = 0$ → so no separate cases

return $\max_{i=1 \text{ to } n} OPT[i]$

---

2. $A[1, \ldots, n]$

length of longest decreasing subsequence

reduce to (1) by taking the reverse of the array

otherwise same algo except $A[j] \le A[i] \longrightarrow A[j] > A[i]$

$A[j] > A[i] \longrightarrow A[j] \le A[i]$

OPT+/-

return max(up, down)

3. $A[1, \ldots, n]$

length of longest alternating subsequence

e.g. < 3 , 4 , 5 , 2 6 5 3 5 8 9 7 >

1: ↗↘↗ up() $1 + \max_{\text{longer ele}}$(down)

2: ↘↗↘ down() $1 + \max_{\text{longer ele}}$(up)

CS 374 Th @ 19 Oct lecture

Maximum Weight Independant Set in a Tree

Recursively: consider the root and its subtrees

$OPT = \sum_{root \to} OPT(\text{children}) \quad OPT(\text{grandchildren})$ $\quad O(n) \cdot O(n) = O(n^2)$

Context free grammars: CYK algorithm

Chomsky Normal Form (CNF) → parse tree of const. deg.

⇒ CYK: $S \to 01 \mid X1$

$Y \to 01 \mid X1$ $\quad 0^n 1^n, n > 0$

$X \to 0Y$

Graphs

State Space search

Tues 31 Oct Lecture

BFS (Breadth First Search)

- DFS/stack/good for graph structure (SCC)
- BFS/queue/good for distances (SSSP)

BFS: $O(n+m)$ time — simple modification of BFS

queue: FIFO/enqueue/dequeue

BFS- partitions tree into layers w/ equal distances from root (BFS tree) & shows cross lines (same height)

spanning tree undirected graph

$\{u,v\} \in E \Rightarrow dist(u) - dist(v) \leq 1$

0 = cut edge or cross edge in BFS tree $= 1$

$\exists$ odd cycles $\Leftrightarrow$ (iff) not a bipartite graph / no 2-coloring

$\exists$ edge b/t. 2 vertices in same layer

2-coloring: color layers alternating

Shortest paths & (Dijkstra's) alg

SSSP: unweighted edges → BFS $O(m+n)$

weighted edges → Dijkstra

intuition for a soln { add fake vertices in middle → BFS to make each edge of len 1 ↳ $O(mL+n)$ soln for L longest edge

floodfill

running time $O(nm)$ finding ith closest neighbor repeatedly

Use heap $O(\log n)$ for n elements

decreaseKey(u) — $O(1)$ time each amortized

~~$O(n)$ inserts~~ n inserts

deleteMin — $O(\log n)$ each for n times

Fibonacci heaps/Relaxed heaps

$O(n \log n + m)$ time total running

/ November Discussion

Dijkstra's algorithm:

① only non-negative edge length

Bellman-Ford for n negative length cycles $O(mn)$

② $O(m+n\log n)$

Recurrence & DP view:

$dist(s,u)$ = length shortest path s & u

$dist(s,u) = \min_{(v,u)\in E} \{dist(s,v) + l(v,u)\}$

• find unvisited vertex v that has min $dist(s,v)$

1. $G = (V, E)$

$l(e) \geq 0$ $\forall$ $e \in E$

$s \in V$, length of shortest cycle containing s

DFS/BFS/Dijkstra's

Dijkstra at s: → mark neighbors, set $dist(s,s) = \infty$.

cycle(s) = $\min(dist(s,v) + l(v,s))$ $\forall v$ s.t. (v,s)

$O(m+n)$ $O(m+n\log n)$

2. n cities

m flights

there can be multiple flights from A to B

flight fares

A to B: cheapest? (or multigraph)

V = cities E = cheapest direct flight A to B directed w/ len cheapest

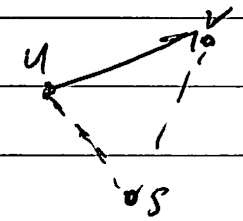
Dijkstra $O(m+n\log n)$

weight$(u) + l(u,v)$ + end weight

each airport → A_{in} / A_{out}

3 n galaxies

m intergalactic



$c(u,v) = 1$ if $d(u,s) < d(v,s)$

$c(u,v) = \infty$ if $d(u,s) > d(v,s)$

$G = (V, E)$ undirected

Wed 8 Nov Lab

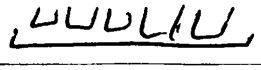
1 Class scheduling

n $s[1 \ldots n]$

$f[1 \ldots n]$

For each Greedy criterion give either a counterexample or proof

① x ends last, discard in conflict w/ x. recurse.

[1, 5] [2, 100] 

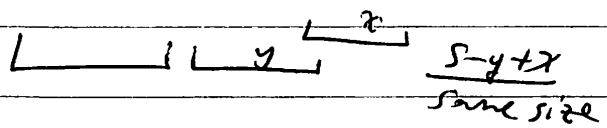
② x starts first, ...

[1, 5] [2, 100]

③ x starts last, . .

~~[2, 5] [1, 100]~~ proof: prove that $\exists$ optimum S s.t. $x \in S$

Assume optimum S s.t. $x \notin S$

~~[illegible]~~ 

Exchange argument

① decide criterion pick one element x

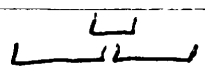
② prove that $\exists$ optimum that contains x (not contains)

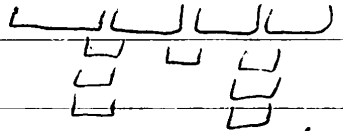
- Assume not true for any ~~[illegible]~~ optimum S
- find $y \in S$ s.t. $S - y + x$ is still feasible

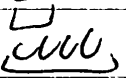
By contradiction

① ~~—~~

② optimum with least # of inversions

④ x w/ shortest duration 

⑤ x that conflicts w/ fewest classes 

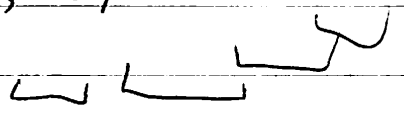
⑥ If no conflict, choose all; class w/ largest duration 

(8) x earliest start time

y second

{ if x, y disjoint, choose x, recurse
{ if x contains y, discard x, recurse
o.w. discard y, recurse

S, $x \notin S$.



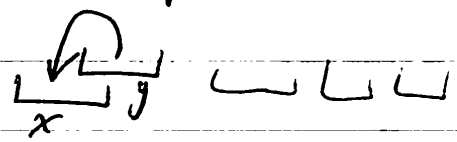
x, y disjoint $\Rightarrow$ no conflicts w/ x

$\Rightarrow$ add x. $\Rightarrow S \leftarrow S+x$

(8) 2 S, $x \in S$.

$S - x + y$ also optimum

3



S, $y \in S$

$S - y + x$ also optimum since x ends before y.

(9) If x contains any other course, discard. recurse.

o.w. choose y ends last. recurse.

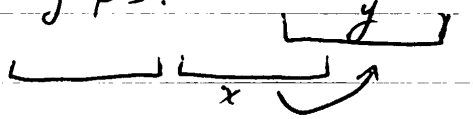
Claim 1: $s[x] < s[y]$, $e[x] > e[y]$.

$\exists$ optimum S, $x \notin S$.

suppose $x \in S \longrightarrow S + y - x$

Claim 2: $\exists$ optimum S, $y \in S$.

suppose $y \notin S$.



$S + y - x$

y does not contain x.

Thurs 9 Nov. lecture

Minimum Spanning Tree — Greedy

cheapest spanning graph — our MST

G is connected ↔ has spanning tree

n-1 edges in spanning tree

Safe edges form a tree

Cycle property

Prim's Algorithm

Kruskal's Algorithm

- pick edge of lowest cost and add if it does not form a cycle with existing edges

Borůvka's Algorithm

Reverse Delete Algorithm

When edge costs are not distinct, order by weight, then indices

Bottleneck shortest path problems

Prim's algorithm is similar to Dijkstra where you fiddle w/ edge prices

└ O(m log m+n)

Kruskal's algorithm

O(m log m) sorting

Union - Find

└ α(n,m) Inverse Ackerman function ≤ 4 (log, log*)

Fri 17 Nov Lab

M vs. $\langle M \rangle$

↓ encoding of M

a string of 0s & 1s.

Proof of undecidability.

Reduce a known undecidable language U.

(e.g., A_{TM}, HALT, E_{TM}, EQ_{TM}, ...)

↓ $\{\langle M, w \rangle \mid M$ is a TM and always halts$\}$

$\{\langle M, w \rangle \mid M$ is a TM that accepts $w\}$ an string w

Proof by contradiction

Assume L decidable ⇒ Decider of L

Decider of U ( )

Construct a TM/DFA. M.

Run Decider of L( )

if accept, ( )

if reject, ( )

$L = \{\langle M \rangle \mid M$ is a TM that accepts the string "ILLINI"$\}$

Decider of U($\langle M, w \rangle$)

Construct a TM/DFA. M'.
$M' = \{$ on what ever input simulate M on w. accept

$L(M') = \begin{cases} \Sigma^* & \text{if } M \text{ halts on } w \\ \emptyset & \text{if } M \text{ doesn't halt} \end{cases}$

Run Decider of L($\langle M' \rangle$)

if accept, (accept) if reject, (reject)

2. $L = \{\langle M \rangle \mid M$ is a TM that accepts exactly 3 strings$\}$

$L' = HALT_{TM}$.

Decider of L' $(\langle M, w \rangle)$

Construct a TM M':

$M' =$ "Simulate M on w
if $x = 01, 10, 110$
accept
else reject"

$L(M') = \{01, 10, 110\}$

Run Decider of L $(\langle M' \rangle)$
if accept, (accept)
if reject, (reject)

3. $L = \{\langle M \rangle \mid M$ is a TM that accepts at least one palintrome$\}$

$M =$ "on input x if x is of form $0^n 1^* 0^n$ then accept. else reject

$M =$ " if x is 1 accept. o.w. reject

$M =$ on input x accept.

$M' =$ {Simulate M on w.
on input x, accept

Wed
29 Nov Lab

1 Black box algorithm:

Input: A CNF formula φ with n binary variables

$x_1, \ldots, x_n$

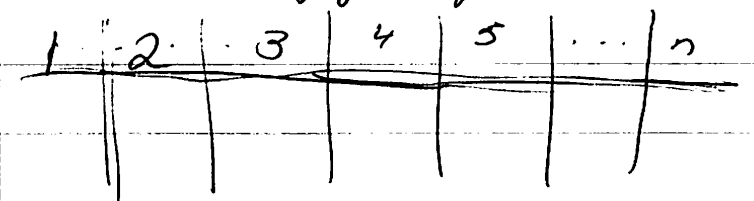
$(x_i \vee \neg x_j \vee \ldots) \wedge (\vee \vee) \wedge (\vee \vee) \ldots$

Output: True if φ is satisfiable

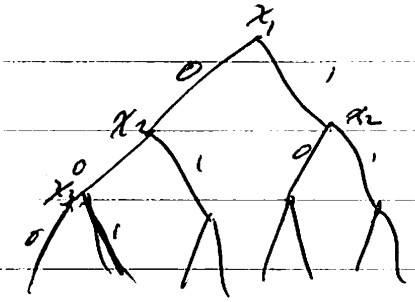
false o.w.

Input: —

Output: A satisfying assignment if φ has one. None o.w.



Hint: how to find a satisfying assignment by Brute force?



$2^n \to 2n$

Given CNF φ

$\varphi_{x_i=1}$: deleting all classes of φ that contains x_i

deleting literal $\neg x_i$ from any clause containing it

$\varphi_{x_i=0}$: similar

Satisfying Assignment (φ):

if SAT(φ) is false
    return none
else:
    for i from 1 to n
        if SAT($\varphi_{x_i=1}$):
            $\varphi \leftarrow \varphi_{x_i=1}$
            A[i] = 1
        else:
            $\varphi \leftarrow \varphi_{x_i=0}$
            A[i] = 0;
    return A[1:n]

2. Independent set.

$S \subseteq V$:

$\forall u, v \in S \quad (u,v) \notin E$

BlackBox: IndSet k?

Input: $G=(V,E)$, k.

Output: if G has indp set $\geq k$ or not

(A) Input: $G=(V,E)$

Output: max size of ind set of G $\quad 1 \leq ? \leq n$

Binary Search: MaxIndSet(G)

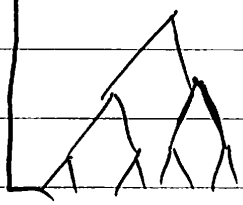
(B) Output: The max size independent set

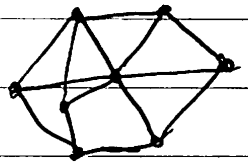
$k =$ ~~Check~~ MaxIndSet(G)

1 2 3 4 5 ... n

0/1 0/1 0/1 0/1 0/1 .

~~Remove~~ Remove v_i: check MaxIndSet$(G) \stackrel{?}{=} k$. $\forall i$





Hint: If v is not in any max independent set,

what is MAXINDSET$(G-v)$? $= k$

MAXINDSET*(G)

$S \leftarrow \phi$

for v is any vertex of G ~~of~~ V in only max ind set

if MAXINDSET$(G-v) = k$: Max$(G - v) < k$

$G = G - v$,

else:

$G \leftarrow G - v - N(v)$

$S \leftarrow S \cup \{v\}$ $\quad k = k-1$

Return S

DP prob on MT: hardest avg 4/20 pts
35% 10th

3 Coloring

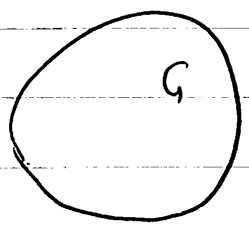
$G = (V, E)$

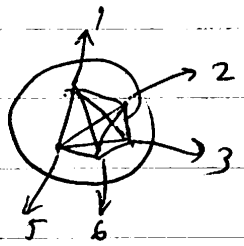
$C = \{1, \ldots, k\}$

$\forall (u,v) \in E, \quad C(u) \neq C(v)$

Alg: Input: G, k

Output: Is there a coloring of G using $\leq k$ colors?





clique

Thursday November 16 Lecture

TM = Turing machine = program

Turing-Church hypothesis – all computers are equivalent

Undecidable – program always stops, accepts/rejects

given pgrm M, input w, does M accept w?

$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$

decider

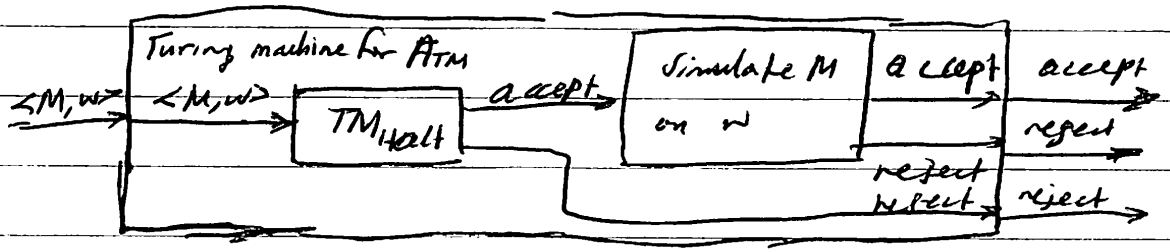
Rice Thm.

decider oracle ORAC black box

$A_{Halt} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ stops on } w\}$

encoding of the program

$A_{TM} \Rightarrow A_{Halt}$



Emptiness

$E_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$

$L(M_w)$

Equality

$EQ_{TM} = \{\langle M, N\rangle \mid M \text{ and } N \text{ are TM's and } L(M) = L(N)\}$.

Regularity

Consider M_w

Rice's Theorem

Asking non-trivial property of programs

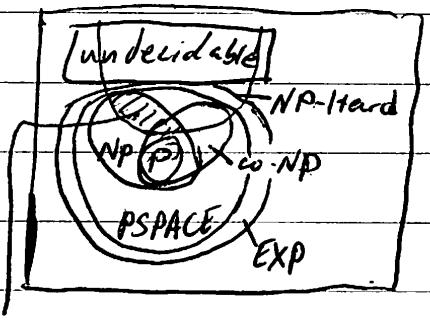
by default do not return a language w/ the property

Any interesting question we cannot solve quickly/efficiently w/ a computer

Tuesday 5 December Lecture

Def A decision problem is in NP if it has a polynomial time certifier for all the YES instances.

Def " co-NP "NO"



Why are NP & co-NP in PSPACE?

NP-complete

Theorem (Cook-Levin) : SAT is NP-complete.

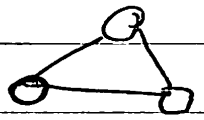
CNF formula

Reducing 3-SAT to Independent Set (3-SAT $\leq_p$ Independent Set)

$x_i = 0$

$x_i = 1$

Clause: $x_1 + x_2 + \overline{x_3}$



Gadgets $k = n + m$ size of independent set $2n, 3m$

# of clauses $\leq m$

connect each literal w/ its complement.

Directed Hamiltonian Cycle

Input: $G = (V, E)$, directed

Eulerian cycle

$2^n = 16$ Hamiltonian cycles : each can either left → right

Add clauses & connections to graph

Directed → undirected $v \Rightarrow v_1 - v - v_3$

Wed 6 December Lab

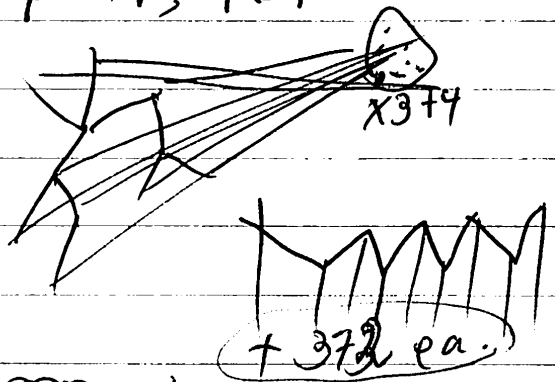
- Hamiltonian $G=(V,E)$ NP-hard
- Hamiltonian path NP-hard start & end diff.

Given $G=(V,E)$

1. does G contain a simple path that visits all but 374 vertices?
2. does G have a spanning tree where $\deg_T(v) \leq 374 \; \forall v \in V$?
3. does G have a spanning tree where ≤ 374 leaves?

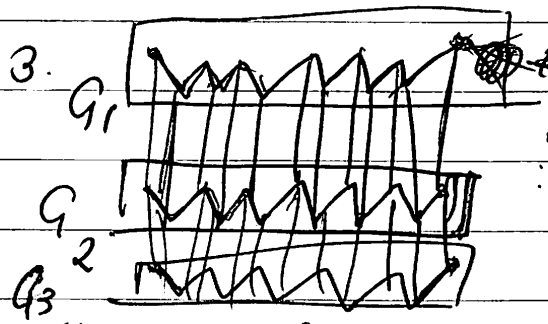
1. $|V| \leq 374$ vertices
   else $|V| \geq 374$ vertices:
   brute force choose all combinations of 374 vertices to remove → any yes → yes

2. Let H be a graph.
   $E(H) = E(G)$. $V(H) = V(G) \cup$ 374 new vertices
   Ham path $G \iff$ all but 374 H
   $p \to p'$; $p \leftarrow p'$



$\deg(v) \; \forall v \in G$
≥ 2
$+372$
≤ 374
each new node
deg = 1
original node connected to 373 + 0, 1

3. $\#\text{con}(G) \iff T(G)$

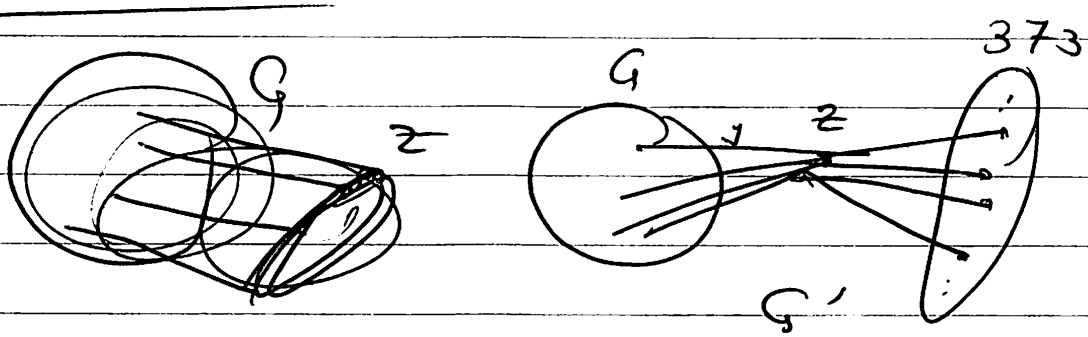
3. 

~~1 leaf~~ / 2 leaves ≥ 2 leaves

$\frac{374}{2} = 187$ copies

$H = 187 \times G$

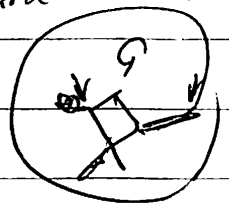
connect each G_i w/ G_{i+1} . ∅ s.t. ∅ vertices connected ∅ to ∅ counterpart.

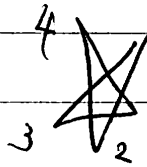


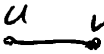
T - new edges ~~4~~

· 5 Coloring – NP-Hard careful ~~Hard~~ $\forall (u,v) \in E$

$\{0, 1, 2, 3, 4\}$





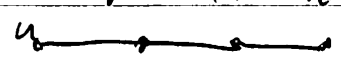


c(u) c(v) differ ≥ 2 (mod 5)

0

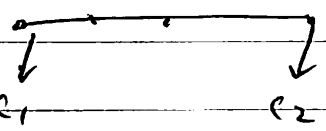
reduce ∅ 5 Coloring ∈ p Careful 5 coloring

Hint: ∀ path of len 3



$c(u) \neq c(v)$

$\forall\ c_1, c_2 \in \{0, 1, 2, 3, 4\}$



Fri 8 Dec Lab

To prove B is NP-complete

- B is in NP
- A (known NP-complete / NP-hard) reduces to B

{ ① certificate (format)
② polynomial (in terms of input size)
③ certifier to verify that a yes instance of B is indeed a yes instance

1. Semi-Independent Set

Input/Instance: A graph G, integer k

Q: is there a set $X \subseteq V(G)$ s.t. no pair of them is connected by an edge or a path of length 2?

Is in NP. ① Certificate: A set of vertices of size k.

② $k \leq n$ $k = O(n)$

③ BFS 2 layers for every u in S.

Certifier:
```
for u ∈ S
  BFS(u)
  check if S is in first two layers
```

2. Subset sum

Input/Instance: a set $S \in \mathbb{Z}^+$, t: integer

example: $S = \{1, 2, 3, 100, 200\}$, $t = \{6\}$

Q: is there a subset $X \subseteq S$, $\sum_{x \in X} x = t$.

① Certificate: subset of size k subset of ints that add to t

② $k \leq n$ (size of set) $k = O(n)$ $|X| \leq n$

③ Certifier:
```
for x in X:
  sum += x
return t == sum
```
Linear time $\subseteq$ poly time certifier

① Decision version ?

② Is NP-complete w.r.t.

2> Polynomial run-time → oracle

③ Assuming black box algorithm for decision version

How to solve the optimization problem in poly time.

1. Max Inner Spanning tree

Input/Instance: a graph G

Q: Compute the spanning tree T s.t. # of vertices in T of deg ≥ 2 is maximized

=1 is minimized

① Is there a ... w/ k # of vertices ... in G?

Reduce Hamiltonian Path to → Max Inner Spanning Tree.

②

G_1

G', $k=2$

$\rightarrow G$

③

```
for e ∈ E(G)
    Ĝ = G − e
    if BB_2(Ĝ) = k*,
        G ← G − e.
    o.w. continue.
```

black box 2

Checks all possible $k \leq n$, binary search

↓

known optimum is k^*

2. NO COVER

Input: $\mathcal{C}$ subsets of finite set S.

Q: compute max k s.t. $S_1, S_2, \ldots S_k \in \mathcal{C}$

s.t. $S \notin \bigcup_{i=1}^{k} S_i$

e.g. $S = \{1, 2, 3, 4, 5\}$, $\mathcal{C} = \{\{1\}, \{2\}, \{1, 3\}, \{1, 4, 5\}, \{2, 5\}\}$

① int k ⟹ $\exists \geq k$ subsets

② $\exists$ one element $x \in S$, $x \in \bigcup_{i=1}^{t} S_i$

Find $x \in S$ s.t. x in min # of subsets in $\mathbb{C}$

Poly time