CS 577

WK1

Fibonacci numbers

LEO VARPO of PISA

1202 Book

ABOUT EFFICIENT CALCULATION

DEFINE BY RECURSION

$F_0 = 0$ $\quad F_n = F_{n-1} + F_{n-2}$

$F_1 = 1$

0 1 1 2 3 5 8 13 21 ....

MODELS POPULATION GROWTH

F_n Increases

GROWS EXPONENTIALLY

$F_n = F_{n-1} + F_{n-2} \geq 2F_{n-2}$

F_n Doubles (at least) EVERY 2 STEPS

ONE CAN SHOW (EXERCISE)

$F_n \sim \text{const} \left(\frac{1+\sqrt{5}}{2}\right)^n$

1.618 (GOLDEN RATIO)

Computational problem:

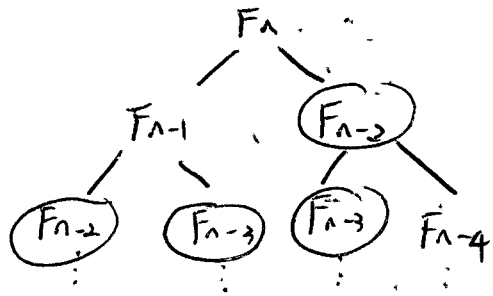
GIVEN n, Find F_n

ALG #1

RECURSIVE PROGRAM

```
Fib(n)          /* ASSUME n>=0 */
  if n=0 return 0
  if n=1 return 1
  return (Fib(n-1) + Fib(n-2))
```

RECURSION TREE



How many NODES?

T_n NODES

$T_0 = T_1 = 1$ $\quad T_n = 1 + T_{n-1} + T_{n-2}$

$M_n = T_n + 1 = (T_{n-1} + 1) + (T_{n-2} + 1)$

$n =$	0	1	2	3	4	...
$M_{n-1} + M_{n-2} =$ M_n	2	2	4	6	10	...
$F_n =$	1	1	2	3	5	

So $T_n + 1 = 2F_n$

$T_n = 2F_n - 1 \sim \left(\frac{1+\sqrt{5}}{2}\right)^n$

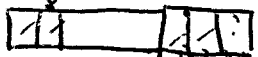
ALG #2

PUT $F_0, F_1, \ldots F_n$ IN ARRAY

F[0] = 1 F[1] = 1

for i = 2 to n

F[i] = F[i-1] + F[i-2]



USES $O(n)$ OPN's (APPLY LookUp, ADD, STORE)

MUCH LESS THAN

$(1.618\ldots)^n$

ALG # $2\frac{1}{2}$ ONLY STORE F_{n-1}, F_{n-2}

before before = 1

before = 1

for $i = 2$ to n /* $n \geq 2$ */

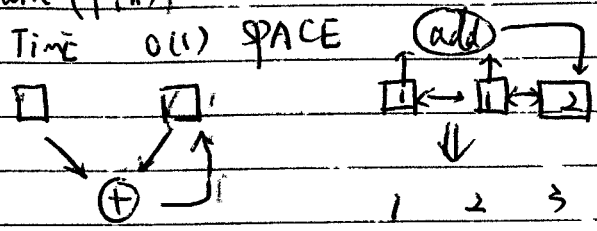
next Fib = before + before before

before before = before

before = next Fib

return (Fib)

$O(n)$ Time $O(1)$ SPACE



WK1 Friday

Finish Fibonacci

ASYMPTOTIC NOTATION

$F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$

ALG #1 STRAIGHT RECURSION

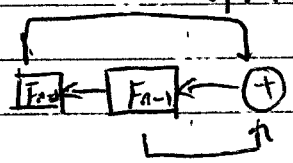
RUN TIME is EXPONENTAL in n.

ALG #2 S THRU ARRAY

$F_0, F_1, \ldots F_{n-2}, F_{n-1}, F_n$

$O(N)$ OPNS.

VARIATION: UPDATE LAST 2 VALUES



SHIFT REGISTER

LINEAR UPDATE RULE.

$F_n = 1 \cdot F_{n-1} + 1 \cdot F_{n-2}$

$F_{n-1} = 1 \cdot F_{n-1} + 0 \cdot F_{n-2}$

ABBREVIATE AS

$$\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}}_{\text{COEFFS}} \begin{pmatrix} F_{n-1} \\ F_{n-2} \end{pmatrix}$$

2×2 MATRIX

$$\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{n-1} \\ F_{n-2} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{n-2} \\ F_{n-3} \end{pmatrix}$$

$$= \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_{n-2} \\ F_{n-3} \end{pmatrix}$$

Product of $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$

THE RULE FOR COMBINING 2 COEFF ARRAYS (MATRIX MULT)

IS ASSOCIATIVE $((AB)C = A(BC))$

NOT COMMUTATIVE $(AB \neq BA)$ IN GENERAL

IN THIS "ALGEBRA" WE CAN SAY

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$$

ALGO#3.

$M = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ COMPUTE M^n BY THE REC

$$P = M^n = \begin{cases} (M^{n/2})^2 & n \text{ EVEN} \\ (M^{\frac{n-1}{2}})^2 M & n \end{cases}$$

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = P \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$$

EACH REC CALL.

HALVES n (OR BETTER)

INVOLVES $O(1)$ OF OPNS ON NUMBERS

$\Rightarrow O(\log N)$ RECURSIVE CALLS

$\Rightarrow O(\log N)$ number operations

REPEATED SQUARING

[

ALG #2: $O(N)$ ADDITION's of #'s of F_n

ALG #3: $O(\log N)$ MULTIPLICATIONS of #'s of F_n
+ADDITIONS

UPSHOT:

OF SINGLE DIGIT OPNS
FOR BOTH ALGO is $O\left(\left(\#\text{ of DIGITS in } F_n\right)^2\right)$
$= O(n^2)$

IF I DIAGONALIZE M

I can show

$$F_n = \frac{1}{\sqrt{5}}(\alpha^n - \beta^n)$$

$$\alpha, \beta = \frac{1 \pm \sqrt{5}}{2}$$

ASYMPTOTIC NOTATION

$f(n), g(n)$ ultimately > 0

$f(n) = O(g(n))$ MEANS FOR SOME $C > 0$
ALL SUFF LG IN
$f(n) \leq C g(n)$

$f(n) = \Omega(g(n))$ .... $f(n) \geq C g(n)$

$f(n) = \Theta(g(n))$ MEANS $f(n) = O(g(n))$
AND $f(n) = \Omega(g(n))$

VARIATIONS ON DEFINITION of LIMIT

Ex: $f = n^2 + 3n + 2$

$g = 3n^2 - 100$

$$\frac{f}{g} = \frac{n^2+3n+2}{3n^2-100} = \frac{1+3/n+2/n^2}{3-100/n^2}$$

$\to 1/3$

so $f = \Theta(g)$ AND $g = \Theta(f)$

$f(x) = \log x \quad g(x) = x$

Ex 2. $\frac{\log x}{x} = \frac{\frac{d}{dx}\log x}{\frac{d}{dx}x} = \frac{\frac{1}{x}}{1} = \frac{1}{x} \to 0$ $\quad x \to \infty$

L'HOPITAL'S RULE

$\Rightarrow \lim_{x \to \infty} \frac{\log x}{x} = 0.$

so $\log x = O(n)$

$\log(n) \neq \Omega(n)$

$\log(n) \neq \Theta(n)$

WK2 Mon

OUT 1/27 (WED)

DUE 2/3

Problem Sessions

Mondays 7:15 - 9:15 1240 CS

Tuesday 5:15 - 7:15 1240 CS

Office minutes TODAY 1:30-2:00

DIVIDE ANG CONQUER

ALGS

KET CHAPS

1) Break into several parts

2) recusively solve subprob.

3) combine these solns to get decired solutions

D/C

CAN substantially improve naive ALG's

Example: Sorting

GIVEN NUMERICAL KEYS $x_1, \dots x_n$

GOAL: REARRANGE INTO $x_1 \le x_2 \le \dots \le x_n$

ENOUGH TO RANK PAIRS " is $x_i \le x_j$"?

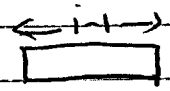
$$\#\{(i,j) \mid 1 \le i \le j \le n\} = \binom{n}{2} = \frac{n(n-1)}{2}$$

INSERTION

WILL SYSTEMATICALLY COMPARE PAIRS

START w/ x_1 (List of size 1) FOR $i = 2, \dots n$

ADD x_i TO THE CURRENT LIST



WORST CASE:

x_i compared TO i-1 PRED. KEYS

$\Rightarrow$ # compare's is $\le \binom{n}{2} = \frac{n(n-1)}{2} = \theta(n^2)$

WE SHOULD USE TRANSITIVITY

IF $x_i \le x_j$, $x_j \le x_k$ THEN $x_i \le x_k$

MERGE SORT TO SORT $x_1, \dots, x_n$

1) let $L = x_1, \dots, x_{\lfloor \frac{n}{2} \rfloor}$

let $R = x_{\lfloor \frac{n}{2} \rfloor + 1}, \dots x_n$

2) SORT L recursively

SORT R

3) REPEATEDLY COMPARE / MERGE

1ST ELTS OF L, R

OUTPUT SMALLEST

Ex.

MERGEING ① 3, 5 (L)
② 7, 8 (R)

OUTPUT — DON'T NEED 1 VS 7 OR 1 VS 8

TO COUNT # of comp's.
USE A RECURRENCE relation WHOSE STRUCTURE MIRRORS THE ALGS

$T(n)$ = WORST-CASE # OF COMP's DONE BY MERGESORT

FOR NOW, $n = 2^k$

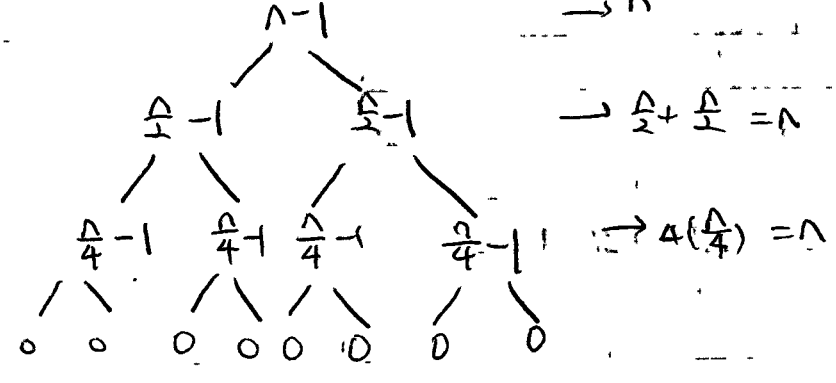
$$T(n) = \begin{cases} 0 & n = 1 \\ \overbrace{T(\frac{n}{2}) + T(\lceil \frac{n}{2} \rceil)}^{2T(\frac{n}{2})} + n-1 & \\ \text{(FOR L)} \quad \text{(FOR R)} \quad \text{(for MERGE)} & \end{cases}$$

IF $n = 2^k$

USE RECURSION TREE

TO SOLVE: PUT MERGE COST ON EACH NODE

Ex: $n = 8 = 2^3$



of -1's is $n-1 = 2^k - 1$ (HERE $2^3 - 1 = 7$)

SUMMING FRACTIONS GIVES 1 AT EACH LEVEL.

$k = \log_2 N$ LEVELS ($k=3$)

$\Rightarrow$ LEVELS SUM TO $n \log_2 n$

So $T(n) = n \log_2 n - (n-1)$ IF $n = 2^k$

IF $n \neq 2^k$, REPLACE BY NEXT HIGHER POWER of 2,

SAY n', $n' \leq 2n$

ADD $+\infty$ DUMMY KEYS

$T(n) \leq T(n') = n' \log_2 n' - n' + 1 \leq 2n(\log_2 n + 1) - 2n + 1$

$= 2n \log_2 n + O(n)$

$\Theta(n \log n)$

ONE CAN SHOW (ALL n)

$T(n) = n \log_2 n + O(n)$

WK 2 Wed

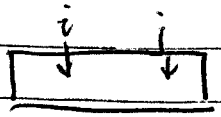
HWK #1

WILL BE ON COURSE WEB PG DUE IN 1 WEEK (WED) 2 PROBLEMS

TODAY: INVERSIONS (KT 5.3)

DEFN: LET $x_1, x_2, \ldots x_n$ BE DISTINCT KEYS

AN INVERSION IS A PAIR $i<j$ with $x_i > x_j$



Ex: 

7 8 HAS ONE INVERSION

1 2 7 8 (SORTED)

IN SOCIAL SCIENCE, WE CAN COMPARE RANKINGS BY COUNTING INVERSIONS.

EG: ISSUES A, B, C, D ← most important

order #1 A C B D ← reference order

order #2 B A C D ← modified order

3 crossings

∴ 3 inversions

KENDALL TAU TEST

n KEYS

$0 \leq$ # OF INVERSIONS $\leq \binom{n}{2} \sim \frac{n^2}{2}$

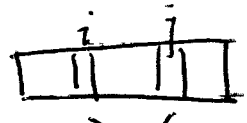
SORTED ↑ REVERSED SORTED

$x_1 > x_2 > \cdots x_n$

IN A RANDOM ORDERING THE AVG (EXPECTED) # of INVERSIONS is $\frac{1}{2}\binom{n}{2} \sim \frac{n^2}{4}$

Proof: USE INDICATED VARIABLES

$$z_{ij} = \begin{cases} 1 & \text{IF } x_i > x_j \\ 0 & \text{if not} \end{cases}$$



$$E[\#\text{ of INV's}] = E\left[\sum_{1 \leq i \leq j \leq n} z_{ij}\right]$$

$$= \sum_{1 \leq i \leq j \leq n} E[z_{ij}] = \sum_{1 \leq i \leq j \leq n} \frac{1}{2}$$

$$= \frac{1}{2}\binom{n}{2} \sim \frac{n^2}{4}$$

⇒ AND SORTING METHOD THAT WORKS BY SWAPPING ADJACENT PAIRS HAS $\theta(n^2)$ AVG RUN TIME

HOW DO WE COUNT INVERSIONS?

(1) TEST ALL PAIRS $\theta(n^2)$

b) DIVIDE AND CONQUER

"PIGGYBACK" ON MERGESORT

MERGESORT FOR $x_1, \ldots x_n$

SORT L (FIRST $\frac{n}{2}$ KEYS)

SORT R (LAST $\frac{n}{2}$ KEYS)

MERGE L W/ R

so # of INV's IS

← # of INV's IN L

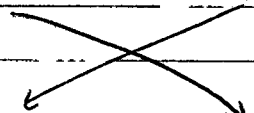
← + # of INV's IN R

← + # of REMOVE BY MERGING

BASE CASE: # INV's = 0 if $n=1$

CONSIDER AN EXAMPLE:

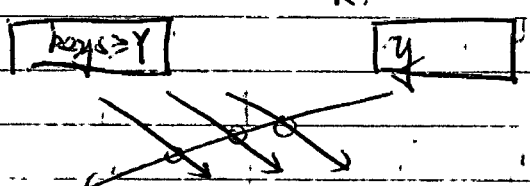
L: 1 2 6 R: 3 7 9 (SORTED)



⇒ 1 2 3 6 7 9

REMOVING A KEY FROM R FIXES SOME ~~IN~~VERSIONS (IF $|L| > 0$)

IN GENERAL:



$|L|$ CROSSINGS WE REMOVE $|L|$ INV'S

L R

MERGE $X_1 \ldots X_m$ $Y_1 \ldots Y_n$

COUNT = 0 while L, R BOTH NONEMPTY /* $m, n > 0$ */

x = 1ST ELT OF L

y = 1ST ELT OF R

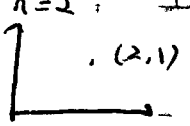
if $x_i < y$ { REMOVE x FROM L /* $m = m-1$ */
ADD IT TO OUTPUT LIST
else /* $y < x$ */ { count = count + + |L| /* add m */
REMOVE y FROM R
ADD TO OUTPUT LIST
ADD REMAINING NONEMPTY L (OR R) TO OUTPUT

ANALYSIS: WE ADD $O(1)$ WORK PER COMPARISON + $O(1)$ work for each internal node of recursion TREE

combine L, R
↙ ↘ | MUST ADD $\Rightarrow$ EXTRA WORK IS $O(n \log n)$
L R | 3 #s
HERE

WK 2 F_{ri}
(re) seatings

TODAY:
MORE DIVIDE + CONQUER
CLOSTEST PAIRS (K&T 5.4)
(A Computational GEOMETRY PROBLEM)
"SPACE"?
$\mathbb{R}^n$ = ALL REAL n-TUPLES
$P = (x_1, \ldots, x_n)$
$Q = (y_1, \ldots, y_n)$



distance between P, Q
is defined to
$d(P,Q) = \sqrt{(x_1-y_1)^2 + \cdots (x_n-y_n)^2}$
$n=1$ | $d(x,y) = |x-y| = \sqrt{(x-y)^2}$
$n=2$ PYTHAGORAS

$d^2 = \sqrt{a^2+b^2}$

GOAL : Given $P_1, \ldots, P_n$ ← #pts
not dimension

WANT TO FIND P_i, P_j $(i \neq j)$ TO MINIMIZE $d(P_i, P_j)$

ONLY $\binom{n}{2}$ pairs ... could TRY ALL PAIRS $\theta(n^2)$

ASSUME

① ALL COMP'S ARE EXACT WE CAN DO $+, -, *, \div, \sqrt{}$ EXACTLY

② GENERAL POSITION" ALL COORDS ARE DISTINCT.

DIM $n=1$

$x_1 < x_2 < x_3 \ldots$ ←

SORT BY X-VALUE $O(n \log n)$

SWEEP THRU LINCREASING i

x_{i-1} x_i

TEST IF x_i, x_{i-1} is smaller than BEST PREV VALUE

DIM

$n=2$: GOAL is AN $O(n \log n)$ ALG WORST CASE

DIVIDE & CONQUER

SORT BY X-COORD USE A VERTICAL LINE L TO DIVIDE INTO 2 SETS Q, R

PT W/ MEDIAN X-COORD

$n=10$ PTS

Q $\frac{n}{2}$ PTS | R $\frac{n}{2}$ PTS

A CLOSEST PAIR IS EITHER IN Q ... IN R, CROSSES Q, R

DO RECURSIVELY

LET δ = min distance FOUND SO FAR

LEMA 1:
ANY CROSSING PAIR W/ A PT THAT IS DIST δ FROM L CAN'T BE BETTER THAN WHAT WE'VE SEEN.
PF (EXERCISE)

BETTER

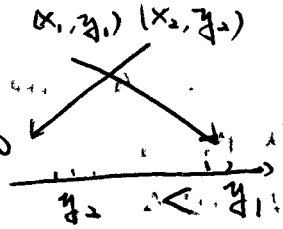
SEARCH FOR A CROSSING PAIR CAN BE DEFINED TO THE STRIP $\{P : d(P, L) \leq \delta\}$

PROBLEM?

→ TOO MANY PTS HERE?

S_y = PTS SORTED BY Y-COORD

x List ... $p < Q$:

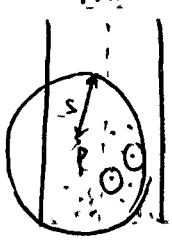


USE x List AS AN INDEX INTO S_Y

BIG IDEA Q

LEMMA 2

IF $P \in$ STRIP, ANY Q AT DIST $< \delta$, MUST BE 5 is Away FROM P IN S_y / IN STRIP



IF USES SPHERE PACKING PUT A DISK OF RADIUS $\delta/4$ AROUND ALL OTHER PTS, EACH DISK HAS AREA $\pi \frac{\delta^2}{4}$

∴ CAN ONLY PUT $O(1)$ SUCH PTS ($\leq \delta$) INTO DISK AROUND P W/ RADIUS $3/2\,\delta$ W/ NO OVERLAPS

WK3 Mon

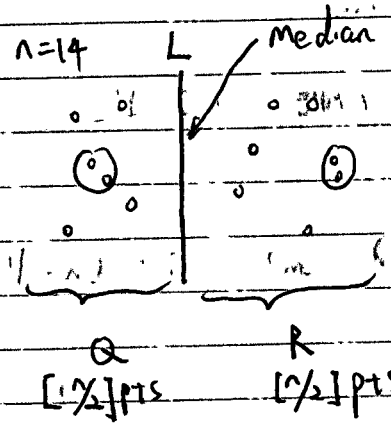
FINISH CLOSEST PAIRS

KT 5.4

INTEGER MULTIPLICATION 5.5

DIVIDE & CONQUER ALG FOR CLOSEST PAIRS IN $\mathbb{R}^2$

n PTS $P_1, \ldots, P_n$ USE A CENTRAL LINE L TO DIVIDE



IF $n \leq 3$ TEST ALL PAIRS

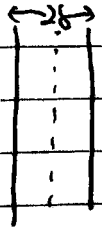
o/w 1) FIND a CLOSEST PAIR IN Q } RECURSION
2) IN R }

δ = BEST DIST SO FAR

3) FIND BEST "CROSSING" PAIR (USE δ)

RETURN BEST PAIR FROM 1)–3) AND BEST DIST.

SEARCH AMONG CROSSING PAIRS USES ONLY PTS IN S = STRIP OF WIDTH 2δ W/L DOWN CENTER.



EACH SUBPROB GETS 2 SORTED LISTS.

a) X-List : SORTED BY X-COORD

b) Y-List : SORTED BY Y-COORD

S_y = PTS IN S, SORTED BY y-COORD.

WE CAN MAKE THE x-List, y-List AND S_y IN $O(n^2)$ time, where $n = |Q| + |R|$.

KEY FACT (5, 10)^ K&T

IF q, r is A CROSSING PAIR IN S ($q \in Q$, $r \in R$) AND $\delta(q, r) < \delta$

THE q, r ARE AT DIST $O(\delta)$ WITH IN S_y

K&T SHOW "$O(\delta)$" ≤ 15.

STEP 3) (COMBINE) CAN MAKE ONE PASS THRU S_y

LIST MANGEMENT CAN BE DONE IN $O(n)$ Time ($n = |Q| + |R|$)

LET $T(n)$ BE RUNTIME

$$T(n) = \begin{cases} \leq 3 & \text{IF } n \leq 3 \\ \leq \underbrace{T(\lceil n/2 \rceil)}_{1)} + \underbrace{T(\lfloor n/2 \rfloor)}_{2)} + \underbrace{O(n)}_{3)} \end{cases}$$

JUST LIKE MERGE SORT

$T(n) = O(n \log n)$

BEATS TESTING ALL $\Theta(n^2)$ PAIRS

INTEGER MULT

"SCHOOL" ALG TO MULT n-DIGIT #S DOES $\Theta(n^2)$ SINGLE DIGIT OPNS

```
  1 2 3
x 4 5 6
-------
  7 3 8
 6 1 5
4 9 2
-------
5 6 0 8 8
```

n ROWS OF ≤ n+1 DIGITS

KARATSUBA'S ALG (1962)

x, y = n-BIT NUMBERS

$0 \le x, y < 2^n$

$xy = (x_1 2^{n/2} + x_0)(y_1 2^{n/2} + y_0)$

$0 \le x_i, y_i < 2^{n/2}$ ← HALF LENGTH

$xy = x_1 y_1 2^n + (x_1 y_0 + x_0 y_1) 2^{n/2} + x_0 y_0$

FORM:

$A = x_1 y_1 \quad B = x_0 y_0 \quad D = (x_1 + x_0)(y_1 + y_0)$

$= x_1 y_1 + x_0 y_1 + x_0 y_0 + x_1 y_0$

$C = D - A - B = x_1 y_0 + x_0 y_1$

SHIFT TO FORM

$A 2^n$

$+ B$

$+ C \cdot 2^{n/2}$

xy

ANALYSIS ASSUME:

$n = 2^k$

LET $T(n)$ = # of SINGLE-BIT OPN'S

$T(n) = 3T(\frac{n}{2}) + O(n) \quad n > 1$

↑ 3 HALF LENGTH x's ↑ SHIFTING ADD, SUBTRACT

$T(1)=1$

TRY $T(n)=n^\alpha$, WE know $\alpha \geq 1$

ON $T(n) = 3T(\frac{n}{2})+n$

$n=1$ DOES NOT WORK

$n' = 3\frac{n'}{2}+n' = \frac{5}{2}n$ $\neq$

TRY $n^\alpha = 3(\frac{n}{2})^\alpha + n$

$n^\alpha \sim \frac{3n^\alpha}{2^\alpha}$

$\Rightarrow 2^\alpha = 3$

$\alpha = \log_2 3 = 1.59 < 2$

EXERCISE FOR YOU:

PROVE BY STRONG INDUCTION ON n THAT

$T(n) = 3T(\frac{n}{2}) + O(n)$

$\Rightarrow T(n) = O(n^{\log_2 3})$

WK3 Wed

MATRIX MULT

S. BASE, computer ALGORITHMS §7.3.4

MATRIX = AN ARRAY OF COEFFS FOR A LINEAR FUNCTION

$y_1 = 3x_1 + 4x_2$

$y_2 = 6x_1 - x_2$

$\begin{pmatrix} 3 & 4 \\ 6 & -1 \end{pmatrix}$

DATA STRUCTURE FOR THE 1 FN

MATRIX MULT: IS FUNCTION COMPOSITION WHERE THE LINEAR FUNCTIONS ARE GIVEN BY MATRICES.

EG: $z_1 = a_{11}y_1 + a_{12}y_2$

(1) $z_2 = a_{21}y_1 + a_{22}y_2$

(2) $y_1 = b_{11}x_1 + b_{12}x_2$

$y_2 = b_{21}x_1 + b_{22}x_2$

PLUG (1) INTO (2):

$$z_1 = a_{11}(b_{11}x_1 + b_{12}x_2) + a_{12}(b_{21}x_1 + b_{22}x_2)$$

$$z_2 = \underbrace{(a_{11}b_{11} + a_{12}b_{21})}_{c_{11}} x_1 + \underbrace{(a_{11}b_{12} + a_{12}b_{22})}_{c_{12}} x_2$$

$$c_{11} = \sum_k a_{1k} b_{k1}$$

IN GENERAL A, B $n \times n$ (a_{ij}) (b_{ij})

$C = A \cdot B$

(c_{ij}) $\quad c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$ (MATCH)

TO FIND ALL c_{ij}'s $1 \le i, j \le n$,

WE DO $n^2 * \Theta(n) = \Theta(n^3)$ OP'S

1960's IT WAS PROVED THAT SOLVING AN $n \times n$ SYSTEM $Ax = b$ HAS THE SAME COMPLEXITY (UP TO CONST FACTORS) AS MATRIX MULT

THE USUAL ALG FOR SOLVING $Ax = b$ IS GAUSSIAN ELIMINATION. AND COSTS $O(n^3)$

STRASSEN (1969)

FOUND A DIVIDE + CONQUER m.m. ALG w// COST $\Theta(n^{\log_2 7})$

$2.81 \cong \log_2 7 < 3$

IDEA: "NON-COMMUTATIVE ALG."

WE CAN MULT 2x2 MATRICES USING 7 *'s

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$x_1 = (a_{11} + a_{12})(b_{11} + b_{22})$

$x_2 = (a_{21} + a_{22})\, b_{11}$

$x_3 = a_{11}(b_{12} - b_{22})$

$x_4 = a_{22}(b_{21} - b_{11})$

$x_5 = (a_{11} + a_{22})\, b_{22}$

$x_6 = (a_{21} - a_{11})(b_{11} + b_{12})$

$x_7 = (a_{12} - a_{22})(b_{21} + b_{22})$

7 *'s, 10 ±'s

$C_{11} = x_1 + x_4 - x_5 + x_7$ 0 *'s

$C_{12} = x_3 + x_5$ 8 ±'s

$C_{21} = x_2 + x_4$

$C_{22} = x_1 + x_3 - x_2 + x_6$

ALG

THIS DOES NOT ASSUME MULT IS COMMUTAIVE ($xy = yx$)

WE CAN USE IT ON HALF SIZE MATRICES

$$n/2 \left(\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array} \right) = \left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array} \right) \left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array} \right)$$

$n/2$

USE IT RECURSIVELY

$n = 2^k$

BASE CASE $n=1$

1 * 0 ±

$M(n) =$

OF MULT'S OF NUMBERS USED BY THE $n \times n$ ALG

$$M(n) = \begin{cases} 7M\left(\frac{n}{2}\right) & n = 2^k > 1 \\ 1 & n = 1 \end{cases}$$

$A(n)$ = # OF $\pm$'s

$$A(n) = \begin{cases} 7A(\frac{n}{2}) + 18(\frac{n}{2})^2 & n>1 \\ 0, & n=1 \end{cases}$$

$$\Rightarrow M(n) = 7M(\tfrac{n}{2}) = 7 \cdot 7M(\tfrac{n}{4}) = 7^2 M(\tfrac{n}{2^2})$$
$$= 7^3 M(\tfrac{n}{2^3}) = 7^k M(\tfrac{n}{2^k}) = 7^k = 7^{\log_2 n}$$
$$= n^{\log_2 7}$$

LINEAR

$$A(n) - 7A(\tfrac{n}{2}) = \tfrac{1}{4}n^2$$

L

$$L(n^{\log_2 7}) = 0$$

$$L(n^2) = n^2 + \tfrac{7}{4}n^2 = \tfrac{11}{4}n^2$$

WE CAN MAKE A SOL'N

$$A(n) = c_1 n^{\log_2 7} + c_2 n^2$$

BY MATCHING INITIAL COND'S $A(1)=0$, $A(2)=18$

$$c_1 = 6 \quad c_2 = -6$$

$$\Rightarrow A(n) = \Theta(n^{\log_2 7})$$

WK3 Fri

HW2 PROBLEM 1

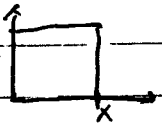
↑ AIM FOR $O(2^{4k}n)$ RUN TIME

USE WEB PG VERSION

TODAY FAST FOURIER TRANSFORM (FFT)

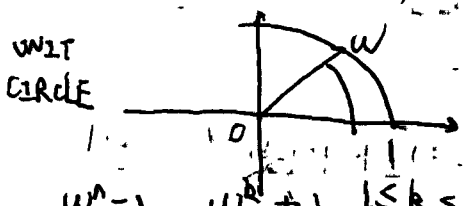
POLYNOMIAL MULTIPLICATION:

$$\omega = e^{\frac{2\pi i}{n}}$$



$x+iy$ $\quad i=\sqrt{-1}$

$= \cos\left(\frac{2\pi}{n}\right) + i\sin\left(\frac{2\pi}{n}\right)$



$\omega^n = 1, \quad \omega^k \neq 1 \quad 1 \leq k \leq n-1$

PRIMITIVE nTH ROOT OF UNITY

DEFN: LET x BE A SEQ OF n NUMBERS
(WRITTEN AS COLUMN VECTOR)
ITS (DISCRETE) FOURIER TRANSFORM IS $y = Fx$
WHERE $F = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ & & \cdots & & \end{pmatrix} = (\omega^{ij}) \quad 0 \leq i,j \leq n-1$

Ex $n=4 \qquad \frac{2\pi}{4} i = e^{\frac{\pi}{2} i} = i$

$$F = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

x

THE "constant" OR "DC" COMPONENT OF THE "SIGNAL x"

F is invertible (WE CAN RECOVER x FROM Y)
ONE CAN SHOW

$$F\bar{F} = \begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & n \end{pmatrix} = nI$$

$\bar{F}$ = CONJUGATE (REPLACE i BY $-i$)

$$\Rightarrow \frac{1}{F} = \frac{1}{n}\bar{F}$$

F I MATRIX W/ A DIFFERENT CHOICE OF ω

$$\omega = e^{-\frac{2\pi i}{n}}$$

THE USUAL ALG FOR FX USES $\theta(n^2)$ OPN'S $(+, -, *, -)$

THE FFT USES $O(n \log n)$ OPN'S

POLYNOMIALS: (ONE VARIABLE)

z, $4z+1$, $3z^3 + 2z - 5$,

IF f, g are PDLYS, $g \neq 0$

$f \bmod g$ = REMINDER UPON DIVISION OF f BY g

$$z^2 - 2 \overline{)z^3 + 3z^2 + 1} \quad \text{quotient } z+3$$

$$z^3 - 2z$$

$$3z^2 + 2z + 1$$

$$3z^2 - 6$$

$$2z + 7$$

$f \bmod g$

$$\deg(f \bmod g) < \deg(g)$$

FOR BINOMIALS

IF $g = z^d - a$

WE CAN OBTAIN $f \bmod g$ AS FOLLOWS:

REPEATEDLY REPLACE (w/ f) x^d By a

$z^3 + 3z^2 + 1$ $\quad z^2 \to 2$

$2z + 3 \cdot 2 + 1$

$2z + 7$

THM $\begin{pmatrix} y_0 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{n-1} \\ & & & \\ 1 & \omega^{n-1} & & \omega^{(n-1)^2} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \\ x_{n-1} \end{pmatrix}$

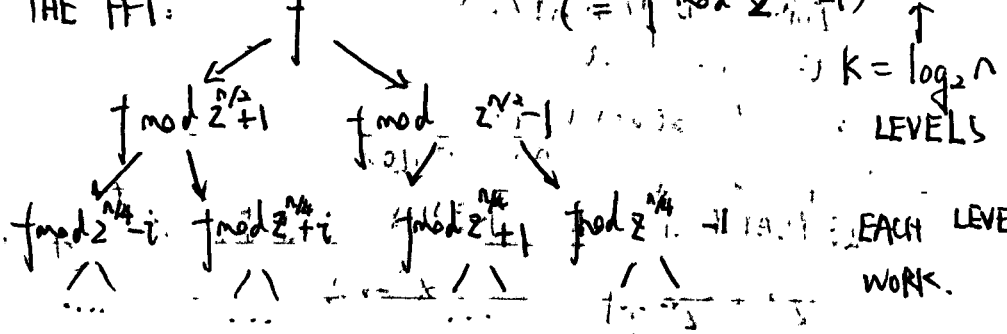
AND $f = x_{n-1} z^{n-1} + \cdots x_1 z + x_0$

THEN $f(\omega^i) = f \bmod z - \omega^i$, $i = 0, \ldots, n-1$

LET $n = 2^k$

$$z^n - 1$$
$$= (z^{n/2}+1)(z^{n/2}-1)$$
$$= (z^{n/4}-i)(z^{n/4}+i)(z^{n/4}+1)(z^{n/4}-1)$$
$$= \text{etc.}$$
$$= \prod_{j=0}^{n-1} (z - \omega^j)$$

THE FFT:



LEAVES HAVE $f(\omega^i) = f \bmod z - \omega^i$, $i = 0, \ldots n-1$

IN SOME ORDER

# of opn's is $\log n \times O(n)$

$= O(n \log n)$

WK 4 Mon

TODAY POLYNOMIAL MULTIPLICATION (AKA CONVOLUTION)

REVIEW OF GRAPH - THEORY

BASKETBALL TOURNAMENT $T(n) = 2T(\frac{n}{2}) + 1$

POLYNOMIAL MULT (KAT'S)

$f = \sum_{i=0}^{n} a_i z^i$, $g = \sum_{j=0}^{n} b_j z^j$

$fg = \sum_{k=0}^{2n+1} \underbrace{\left(\sum_{i+j=k} a_i b_j\right)}_{c_k} z^k = \sum_k c_k z^k$

CONVOLUTION $\begin{matrix} a_0, \dots, a_{n-1} \\ b_0, \dots, b_{n-1} \end{matrix} \to c_0, \dots, c_{2n-2}$

FFT (WORST TIME) $O(n \log n)$ OPN'S

INPUTS n COEFFS OF $f(z) = a_0 + a_1 z_1 + \dots a_{n-1} z^{n-1}$

OUTPUT n VALUES OF f AT w^k, $k = 0, \dots n-1$

$w = e^{\frac{2\pi i}{n}} = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$

WE CAN GO FROM VALUES TO COEFFS WITH (ALMOST) THE SAME ALGS!

$O(n \log n)$ POLY MULT

1) EVALUATE
2) MULTIPLY VALUES
3) INTERPOLATE

$n = 2^k$ $w = e^{\frac{2\pi i}{n}}$

COMPUTE $f(1), f(w), \dots f(w^{n-1})$ $O(n \log n)$

$g(1), g(w), \dots g(w^{n-1})$ $O(n \log n)$

$f(1)g(1), f(w)g(w), \dots, f(w^{n-1})g(w^{n-1})$ $O(n)$

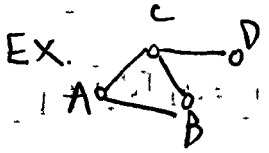
$h(1) \quad h(w) \quad h(w^{n-1})$ $O(n \log n)$

$O(n \log n)$

3) GET COEFFS OF λ FROM THESE VALUES

$O(n \log n)$

GRAPHS = VERTICES + EDGES
(FINITE) 2-SUBSETS OF VERTEX SET

EX. 

$V = \{A, B, C, D\}$

$E = \{\{AC\}, \{A, B\}, \{BC\}, \{CD\}\}$

MODEL RELATIONSHIPS BTWN TWO THINGS
ALLOW VISUAL/GEOMETRIC INSIGHTS

MOST EFFICIENT GPH ALGS USE NEIGHBOR LISTS TO REPRESENT GRAPH

A : C, B
B : A, C
C : A, D, B
D : C

TOTAL # OF NEIGHBOR LIST EG
$= 2 + 2 + 3 + 1 = 8$

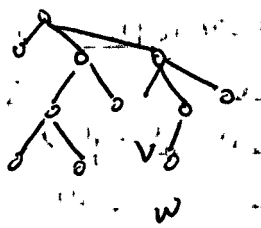
ALWAYS $\Theta(2 \cdot$ (# OF EDGES))

GRAPH SEARCH (AKA TRAVERSAL) IS AN EFFICIENT WAY TO EXAMINE ALL VERTICES (SOMETIMES, ALL EDGES OF A GRAPH)

START VERTEX $\sim \{v, w\}$

$R = \{s\}$ WHILE $\exists$ EDGES $\{v, w\}$ WITH $v \in R$, $w \notin R$
ADD w to R

WE CAN IMAGINE THIS GROWS A TREE



DEFN G IS CONNECTED IF AT THE END OF GPH SEARCH, R=V.
(HOLDS IFF EACH PAIR OF VERTICES IS LINKED BY A PATH

IF SO THE TREE CONSTRUCTED IS CALLED A SPINNING TREE

2 COMMON RULES FOR DECIDING WHICH "NEW" EDGE v-w TO USE.

1) DEPTH - FIRST ALWAYS EXPLORE FROM MOST RECENTLY DISCOVERED VERTEX } USE RECURSION OR A STACK

2) BREADTH FIRST
EXPLORE VERTICES AT DIST 1 FROM S
THEN DIST 2, 3, 4, ... } USE A WORK QUEUE (FIFO)

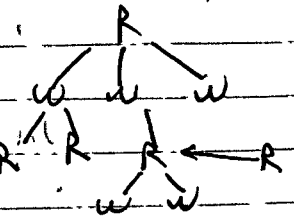
BOTH CAN BE CODED TO USE $O(m+n)$ OPN's ($m = |E|$, $n = |V|$)

BIPARTITE GRAPH

DEFN G IS BIPARTITE IF WE CAN COLOR VERTICES (R OR W), S.T. NO EDGE HAS ITS ENDS THE SAME COLOR

↑ TO TEST THIS
RUN GRAPH SEARCH (ALONG TREE) COLOR ACCORDING TO DIST FROM S

IF ALL NON TREE EDGES ARE BICHROMATIC
G IS BIPARTITE



IF ∃ A NON-TREE EDGE IS MONOCHROMATIC, WE HAVE AN ODD LENGTH CYCLE
⇒ NO VALID COLORING
⇒ G IS NOT BIPARTITE

WK 4 WED

TODAY DIRECTED GRAPHS

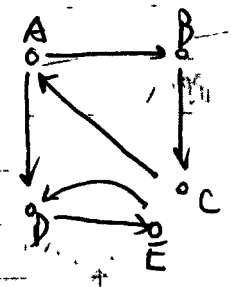
UPCOMING SHORTEST PATHS
MIN SPANNING TREE
INTERVAL PROBS

DIRECTED GRAPHS

EDGES ARE ORIENTED

$v \to w \neq w \to v$

NO SELF LOOPS



CYCLE
A → B → C → A

ADJACENCY LISTS
FOR EACH VERTEX v, LIST WHICH ST. $v \to w$ IS AN EDGE

EACH EDGE OCCURS ONCE.

USED IN SCHEDULING
VERTICES = JOBS
A → B MEANS A MUST PRECEDE B

USUALLY, NO (DIRECTED) CYCLES
(IF ∃ A CYCLE, CONSTRAINTS AREN'T FEASIBLE)

DIRECTED ACYCLIC GRAPHS
DAG

A TOPOLOGICAL ORDERING OF G's VERTICES IS A LIST

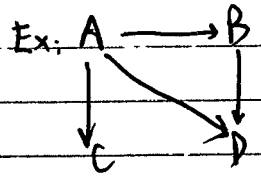
$v_1, v_2, \ldots, v_n$ $V = \{v_i, i = 1, \ldots, n\}$

S.T. IF $v_i \to v_j$ THEN $i < j$

(E.G. A FEASIBLE SCHEDULE FOR JOBS)

THM: EVERY DAG HAS A TOPOLOGICAL ORDERING

WE CAN FIND IT IN LINEAR TIME! $O(m+n)$ $n = |V|$ $m = |E|$

Ex: 

3 TOPOLOGICAL ORDERINGS A, C, B, D

A B C D

A B D C

PF OF THM.

USE INDUCTION ON $n = |V|$

$n = 1$ NO EDGES, OK

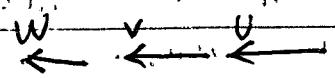
$n > 1$ EVERY DAG HAS A SOURCE (NO EDGES INTO IT)

PUT v FIRST IN THE LIST

REMOVE IT AND ALL ITS (OUTGOING) EDGES FROM G.

RECURSIVELY PROCESS REMAINING GPH

TO SHOW THIS IMAGINE FOLLOWING EDGE BACKWARDS



MUST REPEAT A VERTEX

(TO AVOID SEARCHING FOR THE NEXT SOURCE:

MAINTAIN IN-DEGREE (# OF VERTICES POINTING TO IT)

FOR EACH VERTEX.

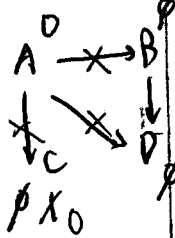
ALG:

COMPUTE (ALL IN-DEGREES

(SCAN ADJACENT LISTS)

PUT ALL IN DEG 0 VERTICES ON A WORK QUEUE Z
A SET

WHILE $Z \neq \emptyset$
  REMOVE v FROM Z
  FOR EACH EDGE $v \to w$, DECREMENT w's IN-DEGREE
  IF THIS VANISH, ADD w to Z



$Z = \{A\}$ OUTPUT LIST A

$Z = \{B, C\}$ B

$Z = \{D, C\}$

$Z = \{C\}$ D

$Z = \emptyset$ C

DONE!

THE ALG, PROCESS EACH EDGE $v \to w$ TWICE $O(m)$
(ONCE TO RAISE w's IN-DEG
ONCE TO LOWER IT) $O(n)$
EACH VERTEX IS PUT ON Z
ONCE & REMOVE ONCE. TOTAL WORK: $O(m+n)$

WK4 Fri
TOPIC DU JOUR:
SHORTEST PATHS K&T 4.4
(HEAPS K&T 2.5)

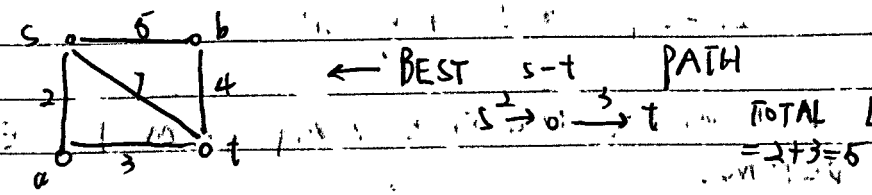
VERTICES

$G = (V, E)$ CONNECTED GPH

EDGES

EACH EDGE HAS A LENGTH l_e.

(TODAY: $l_e \geq 0$)



← BEST s-t PATH

$s \xrightarrow{2} a \xrightarrow{3} t$ TOTAL LENGTH $= 2+3=5$

GIVEN $s, t \in V$, FIND A PATH FROM s TO t OF MIN TOTAL LENGTH.

WE ARE MINIMIZING OVER AN INFINITE SET.

WE CAN RESTRICT TO SIMPLE PATHS (NO REPEATED VERTICES)

(FINITE SET)

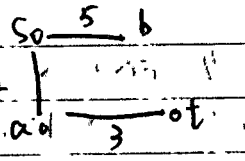
CAN REMOVE THIS IT HAS LENGTH ≥ 0

LENGTH OF PATH CAN'T INCREASE.

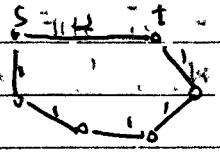
ALL KNOWN ALG's TO FIND A SHORTEST s-t PATH COMPUTE SHORTEST PATHS (FROM s) TO ALL $v \in V$

USUALLY DONE BY MAKING A TREE.

E.G.



SIMPLE "GREEDY" STRATEGIES DON'T WORK:



DIJKSTRA (1959)

MAKE SURE THAT

ALL VERTICES IN THE CURRENT TREE ARE LABLED BY THE

INVARIANT MIN DIST TO ROOT S

INITIALLY T = S + NO EDGES

EACH VERTEX ONE EDGE AWAY FROM T GETS A TENTATIVE DISTANCE.

REPEAT: DIJKSTRA'S RULE CHOOSE A VERTEX V W/ MIN TENTATIVE DIST AND ADD IT TO TREE.

SCAN V'S NEIBORS & UPDATE (REDUCE) TENTATIVE DIST'S IF NECESSARY

UNTIL T HAS $n = |V|$ VERTICES

FRINGE = NON-VERTICES THAT CONNECT TO T BY ONE EDGE.

MAINTAIN A LIST OF ALL FRINGE VERTICES

FOR EACH, TENTATIVE DIST

CANDIDATE CONNECTION TO T.

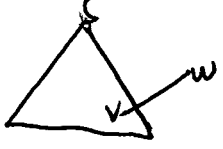
CORRECTNESS PF

SHOW BY INDUCTION ON t (= # VERTICES IN T)

THAT ALL DIST'S TO VERTICES IN T ARE CORRECT.

$t=0$ [0]

$t \to t+1$



LET w BE THE NEW VERTEX, ITS CADIDATE IS ✓

LET P BE ANY OTHER PATH TO W

OTHER PATH TO W. EXISTS THE TREE FIRST VIA

$v' - w'$

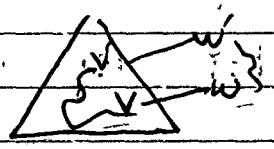
$\in T \quad \notin T$

LENGTH (P) $\geq$ LENGTH OF TREE + $(v'-w')$

PATH TO v'

$\geq w'$s TENTATIVE DIST

so ACCEPTABLE TO ADD W..



W1<5 Mon

MIN COST SPANNING TREE

D ALG: BUILDS A TREE T OF SHORTEST PATHS FROM s (START VERTEX)

T GROWS (FROM THE ROOT s) ALL VET HAVE A DIST FROM s (CORRECT)

ALL NEARBY VERTICES HAVE A TENTATIVE DIST (MIGHT BE TOO LARGE) + A TENTATIVE PARENT.

TREE GROWING STEP:

CHOOSE $v \in V$ w/ SMALLEST TENTATIVE DIST, TENTATIVE PARENT v

DELETE MIN

ADD $v \to w$ TO T

IF w's NEW DIST IS BETTER

SCAN w's NBRS, UPDATE THEIR TENTATIVE DIST, TENTATIVE PARENT

DECREASE KEY AT w.

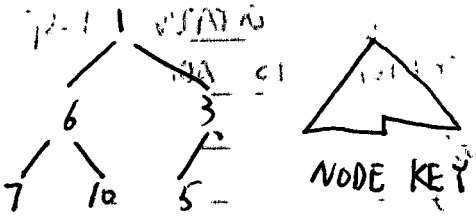
CORRECTNESS LAST TIME

(CRUCIAL THAT DIST ≥ 0)

Efficient Implementation.

AKA

USES A PRIORITY QUEUE (HEAP)
A "SET" DATA STRUCTURE
ELT's HAVE A NUMERICAL KEY
WE CAN BUILD A HEAP w/ n ELTs $O(n)$
DELETE MIN $O(\log n)$
DECREASE KEY $O(\log n)$
USUALLY STORED IN A BINARY TREE:



NODE KEY ≥ KEY (CHILD)

SEE KLT 12.5

TO IMPLEMENT DIJKSTRA
PUT ALL VERTICES INTO A HEAP
(KEY = TENTATIVE DIST)
ANALYSIS: 1. CREATE HEAP $O(n)$ ← DIST = $l_{s,v}$, IF v
$n-1$ DELETE MIN $O(n \log n)$ CONNECTS to s
$\leq m$ DECREASE KEY $O(m \log n)$ + ∞ O/W
$m = |E|$ CONNECTED
$n = |V| \leq m$ $m \geq n-1$

RUNTIME
FOR DIJKSTRA IS $O(m \log n)$ (ALMOST LINEAR)

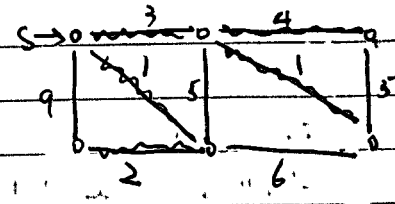
1930's MINI-COST SPANNING TREE
(JARNIK) GIVEN A GRAPH (V, E) (CONNECTED)
EACH EDGE $e \in E$ HAS A cost c_e
GOAL: FIND A SPANNING TREE ($n-1$ VERTICES)

(1957) PRIM'S ALG
GROWS A TREE

TREE GROWING STEP — AMONG ALL EDGES LEADING OUT OF T, CHOOSE ONE W/ MIN COST

"GREEDY STEP"

+ADD IT TO TREE

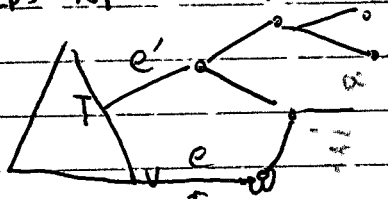


TOTAL COST: $3+1+2+4+1 = 11$

INVARIANT.

T CAN BE EXTENDED TO A M.S.T. T'

HOLDS AT START. CONSIDER ADDING OLD EDGE.



GREEDY EDGE CHOICE

BY INDUCTION

T EXTENDS TO SOME M.S.T T'

IF $e \in T'$, OK.

SUPPOSE $e \notin T'$

SOME OTHER EDGE $e' \in T'$

LEADS OUT OF T

$T'+e$ HAS A UNIQUE CIRCLE

AN EXCHANGE ARGUMENT.

$\hat{T} = T' - e' + e$ IS A SPANNING TREE (CONNECTED, ALL VERTICES)

COST $(T' - e' + e) = \text{COST}(T') + \text{cost}(e - e')$

$= \text{cost}(T') + (C_e - C_{e'})$

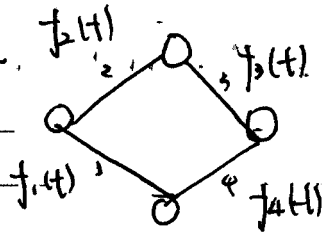
≤ 0 (BY GREEDY CHOICE)

$\leq \text{cost}(T')$

$\Rightarrow$ (BY MINIMALITY OF T') $\hat{T}$ IS A M.S.T. EXTENDING $T'+e$

$\square$

#2



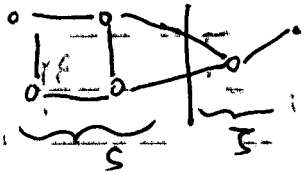
WK5 WED

G = connected grp of EGDE COSTS

Goal: FIND A TREE LINKING ALL VERTICES OF MIN TOTAL COST.

CUTS:

DEFN: A CUT IS A DIVISION OF V INTO 2 DISJOINT NONEMPTY SETS S, $\bar{S} = V - S$



MOST (ALL?) MST ALG'S USE A COMMON FRAMEWORK

$E = \{\text{ACCEPTED}\} \cup \{\text{REJECTED}\} \cup \{\text{UNCLASSIFIED}\}$

APPLY THESE RULES (IN ANY ORDER) UNTIL A MST IS DETERMINED

(n-1 ACCEPTED, OR m-n+1 REJECTED)

CUT RULE: FIND A CUT CROSSED BY NO ACCEPTED EDGE CHOOSE A MIN-COST CROSSING EDGE AND ACCEPT IT.

CYCLE RULE: FIND A CYCLE w/ NO REJECTED EDGES. REJECT A MAX-COST CYCLE EDGE

INVARIANT

AT ALL TIMES, $\exists$ A MST CONTAINING ALL ACCEPTED EDGES

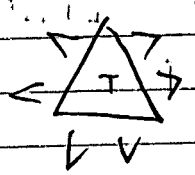
AND
NO REJECTED EGDES

PF: EXCHANGE ARGUMENTS (K&T PP MAY 148).

SOME MST ALG'S

① PRIM'S TREE GROWING)

ALGO: USES THE CUT RULE



② REVERSE-DELETE:

USE THE CYCLE RULE ONLY.

③ KRUSKAL (1956)

SORT EDGES (CHEAPEST FIRST)

T = ∅ / * FOREST */

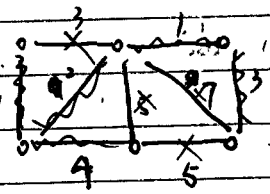
WHILE T HAS<n VERTICES

e = NEXT EDGE

IF (T+e HAS A #CYCLE, REJECT e

ELSE ACCEPT e (T=T+e)

Ex:

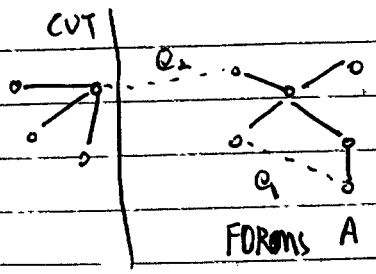


ACCEPTED

REJECTED

TOTAL COST = 1+1+2+3+4 = 11

K's ALG EXPLOITS BOTH RULES



FORMS A CYCLE w/ CHEAPER OR NO WORSE EDGES

OK TO REJECT e_1

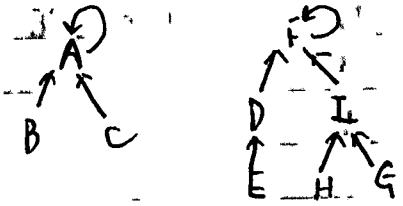
e_2 CROSSES A CUT

NO BETTER EDGE DOES

OK TO ACCEPT e_2

AN EFFICIENT KARUSKAL IMPL NEEDS FAST CYCLES TESTING
(NO BFS/DFS)
USE TREES

EACH COMPONENT OF T HAS ITS VERTICES IN A TREE w/ LINKS POINTING TOWARD ROOT



x—y MAKES A CYCLE IFF r = ROOT OF x's TREE
= ROOT OF y's TREE = s

ALONG CHASING PATH, WE MAKE ALL PATH VERTICES POINT TO ROOT (PATH COMPARESSION)

IF r≠s, JOIN TREES MAKE THE "SMALLER" TREE POINT TO THE OTHER ONE
ANALYSIS COMITTED) $n \leq (\frac{2}{2}) \leq n^2$
⇒ K's ALG USES = O(mlogm)
= O(mlogn)
STEPS

"UNION FIND"

WK5 Fri
Interval Scheduling

JOBS $J_1, J_2, \ldots J_n$
Each has A starting time and A finishing time
(START ≤ FINISH)

A SET OF JOBS IS COMPATIBLE IF NO TWO OVERLAP

(ARE COMPATIBLE)

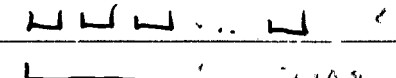
GOAL: FIND A LARGEST SET OF COMPATIBLE JOBS

(2^n SUBSETS, WE'D LIKE A POLY (IN n) ALG.)

MANY SIMPLE HEURISTICS DON'T WORK

E.G.

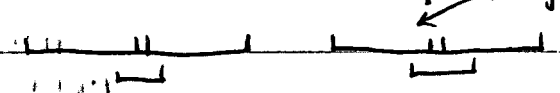
1) EARLIEST START TIME



WE PICK 1 JOB, OPT IS n-1 JOBS

PERFORMANCE RATIO $= \frac{1}{n-1} \to 0$

2) PICK SHORTEST JOBS FIRST



WE PICK $\frac{n}{3}$ PERF RATIO $= \frac{1}{2}$

OPT $= \frac{2n}{3}$

A RULE THAT DOES WORK:

GREEDY: SORT BY FINISH TIME

(@EARLIEST FIRST)

GO THRU IT'S ALL JOBS IN ORDER

IF IT'S COMPATIBLE W/ ALL JOBS SO FAR TAKEN, USE IT.

ELSE GO ON TO NEXT JOB

SIMILAR TO KRUSKAL MST

EACH JOB IS ACCEPTED OR REJECTED (NO BACKTRACKING)

FEASIBILITY:

FOLLOWS FROM 1ST INSTR IN LOOP

LEMMA (GREEDY STAYS AHEAD)

SUPPOSE GREEDY PICKS r "JOBS" (MAYBE MORE)

LET O BE ANY OTHER RULE

PICK r JOBS

FINISH TIME FOR GREEDY'S rth JOB $\leq$ SAME FOR O'S rth JOB

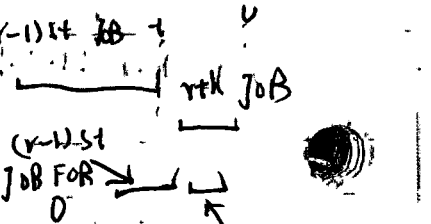
PF: INDUCTION ON r

$r=0$, NO JOBS SO OK FOR GREEDY

$r>0$

ASSUME TRUE FOR $r-1$,

PROVE FOR r.



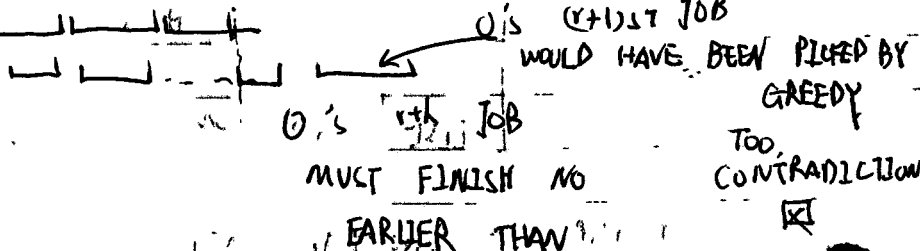
STMT: HOLDS FOR rth JOB AS WELL

THM (OPTIMALITY)

GREEDY PICKS AT LEAST AS MANY AS ANY OTHER RULE

PF. SUPPOSE GREEDY PICKS r JOBS

O PICKS $> r$ JOBS GREEDY



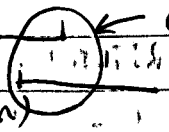
☒

RUN TIME:

$O(n \log n)$ TO SORT

ACCEPT / REJECT TEST IS $O(1)$

GREEDY



LOOP IS $O(n)$

TOTAL : $O(n \log n)$

WK6 MON

DYNAMIC PROGRAMMING

WEIGHTED INTERVAL SCHEDULING (KST 6.1-6.2)

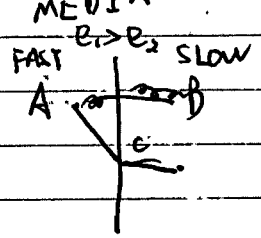
DYNAMIC PROGRAMMING

ROOTED IN PHYSICAL "LEAST EFFECT"

1950's BELLMAN ADAPTED THESE IDEAS TO OPTIMIZATION

FERMAT'S LEAST-TIME PRINCIPLE: A LIGHT RAY FROM A TO B WILL ALWAYS TAKE THE PATH REQUIRING LEAST TIME

2 MEDIA



LIGHT BENDS (REFRACTION)

CALCULUS $\Rightarrow$ FIND THE CROSSING PT C

BELLMAN'S PRINCIPLE OF OPTIMALITY

EVERY PART OF AN OPTIMAL SOL'N IS ALSO OPTIMAL

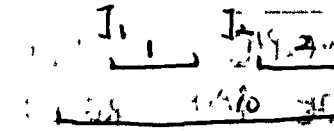
DYNAMIC PRINCIPLE WILL BE APPLICABLE WHEN THIS HOLDS.

EX. IN A) EDGE (DIST ≥ 0)

A SHORTEST PATH S TO T,
IF v IS AN INTERMEDIATE VERTEX,
THE PATH FROM S TO v IS ALSO SHORTEST

SUGGESTS AN ALGORITHMIC STRATEGY, SOLVE ALL SUBPROBLEMS, USE THE ONES YOU NEED.
E.G: A SINGLE-SOURCE S.P. ALG FINDS BEST PATHS FROM S TO ALL V

WEIGHTED INTERVAL SCHEDULING
JOBS $J_1, \ldots, J_n$, EACH HAS A START TIME AND FINISH TIME ($\geq$)
EACH JOB HAS A VALUE (>0) v_k = VALUE OF J_k
GOAL: FIND A SET OF COMPATIBLE (NON-OVERLAPPING) JOBS OF MAX TOTAL VALUE.

EX 

THE GREEDY RULE (LAST TIME) PICKS J_1, J_3 (VALUE 1+2=3), BUT J_2 IS THE BEST

DP SOL'N
SORT JOBS BY FINISH TIMES (EARLIEST FIRST) $J_1, \ldots, J_n$

CONSIDER USING J_n OR NOT

2 POSSIBILITIES
(J_n NOT USED)
PICK A BEST SET FROM $J_1, \ldots, J_{n-1}$
(J_n USED)
PICK A BEST SET FROM THE JOBS $J_i, \ldots, J_p$ ← USED HERE

THAT FINISH NO LATER THAN J_n's START TIME.
PICK THE BEST ALTERNATIVE

DP WILL BE APPLICABLE WHEN THIS HOLDS

DEFN: $P(k)$ = { LARGEST $j<k$ S.T. J_j FINISHES BEFORE (OR EQUAL) J_k STARTS
0 IF NO SUCH EXISTS

$P(0)=0$

WE CAN FIND ALL $P(k)$, $0 \le k \le n$, IN TIME $O(n \log n)$
(BINARY SEARCH)

ALG TO FIND VALUE OF OPT(k) $P_1 \dots P_k$, $k=1,\dots,n$

$O(n \log n)$ 1) SORT JOBS BY FINISH TIME

2) COMPUTE $P(k)$ $0 \le k \le n$

$O(n)$ 3) FOR $k=2$ TO n DO

$$OPT(k) = \max \{ OPT(k-1),\ v_k + OPT(P(k_i)) \}$$

↳ TOTAL $O(n \log n)$

4) [RECURSIVELY FIND A BEST FOR $J_1, \dots J_n$]

IF $OPT(n) = OPT(n-1)$ /* J_n out */
RECURSIVELY SOLVE w/ $n=n-1$

ELSE /* J_n IN k */
RECURSIVELY SOLVE w/ $n = P(n)$
ADD J_n TO SCHEDULE
BASE CASE IF $n=0$ SHED $= \emptyset$

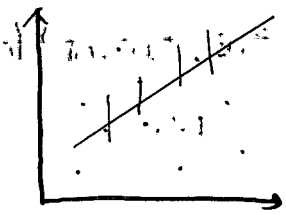
WK 6 / WED

TODAY: SEGMENTED LEAST SQUARES

K&T 6.3

LEAST SQUARE

* DATA PTS (x_i, y_i) $i = 1, \dots, n$



FIND A "GOOD" APPROXIMATION: LINE

$y = ax + b$

LINE OF BEST FIT

$$E_{1,n} = \sum_{i=1}^{n} (y_i - (ax_i + b))^2$$

$$= \text{using DATA points} \; \|y - (ax_i + b)\|_2^2$$

$\omega / \; 1 \le i \le n$ (L_2 DIST)

A BEST LINE CAN BE FOUND BY SOLVING

$$\frac{\partial E}{\partial a} = 0 \qquad \frac{\partial E}{\partial b} = 0$$

GET LINEAR EQN's FOR a, b

SOLVE THEM:

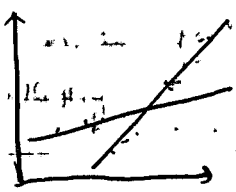
BEST a, b: $a = \dfrac{n(x,y) - S_x S_y}{n S_x^2 - (S_x)^2}$

$$b = \frac{S_y - a S_x}{n} \qquad \text{K\&T p. 262}$$

$S_x = \sum x_i$, $S_{x^2} = \sum x_i^2$

$(x, y) = \sum x_i y_i$

WE MIGHT NEED DIFFERENT PREDICTORS FOR DIFFERENT x-RANGES.



DISCRETE PROBLEM.

HOW MANY LINES?

WHAT ARE THE BREAK PTS?

INTRODUCE A PENALTY FN

$$\text{PENALTY} = C\left[\#\text{ OF LINES}\right] + \sum_{\text{SEGMENTS}} (E \text{ FOR SEGMT})$$

$C > 0$

GOAL: FIND A PARTITION OF A MIN PENALTY

HOW MANY CHOICE?

$n=7$

$x_1 < x_2 < x_3 \nless x_4 < x_5 \nless x_6 < x_7$

SMT SMT SMT

$n=2$ $x_1 < x_2$ 2 WAYS

$x_1 \nless x_2$

$n-1$ places to divide

$$\binom{n-1}{0} + \binom{n-1}{1} + \cdots \binom{n-1}{n} = 2^{n-1}$$

A POLY TIME DP ALG

IMAGINE CHOOSING THE LAST SGMT:

$x_1 < x_2 \cdots < x_{i-1} < x_i < \ldots x_n$

SGMT $i \ldots n$

PRINCIPLE OF OPTIMALITY

IF $i \ldots n$ IS THE LAST SEGMT IN A BEST PARTITION, THE PTS $1 \ldots i-1$ THEMSELVES PARTITIONED OPTIMALLY

$\text{OPT}(I) =$ MIN PENALTY OVER PARTITION OF $1 \ldots I$

INCREMENTAL COST OF j more dom

$$\text{OPT}(J) = \begin{cases} 0 & J=0 \\ \min_{1 \le j \le J} \{ \text{OPT}(j-1) + C + E_{ij} \} & \text{PENALTY FOR MODELING DATA PTS ABOVE } x_1, \ldots x_{j-1} \end{cases}$$

CHOOSE LAST BREAK PTS

Last Elmt or x_{j-1}

ALG:

For $j = 0, \dots, n$.

USE RECURIVE TO compute OPT(j)

ANALYSIS

① WE CAN COMPUTE ALL E_{ij} AND EXPLAIN $S_{x_i}, S_{x_2} \dots$ IN $O(n^2)$ STEPS.

E.G: $S_{ij} = \sum_{1 \le k \le j} x_k$

```
for i = 1 TO n
    S_i,i = x_i
for j = i+1 TO n
    S_i,j = S_i,j-1 + x_j
```

USE DP

② TO FIND OPT(j)

WE MINIMIZE A OVER j QUANTITIES

OPT(j) USE O(j) work

TOTAL TO GET OPT(0), ... OPT(n)

$\sum_{j=1}^{n} \theta(j) = O(n^2)$

①+② IS $O(n^2)$ work.

WK6 Fri

SUBSET SUM PROBLEM (KET 6.4) KNAPSACK (MAYBE)

COVERING (MINIMIZE)

MUST CHOOSE "ENOUGH" COSTLY OBJECTS

Ex: MIN-COST SPAINNING TREE

VS. PACKING

WANT TO CHOOSE PROFITABLE OBJECTS

BUT NOT "TOO MANY"

Ex: SUBSET SUM (TODAY)

GIVEN WEIGHTS $w_1, w_2, \ldots w_n$ POSITIVE INTEGERS

CAPACITY W

GOAL: CHOOSE A SET $S \subseteq \{1, \ldots, n\}$ so that $\sum_{i \in S} w_i$ is as Big as possible, s.t. $\sum_{i \in S} w_i \leq W$.

SAME AS $\max \sum_{i=1}^{n} x_i w_i$ s.t. $x_i \in \{0,1\}$ ($0 \leq x_i \leq 1$)

$\sum x_i w_i \leq W$

AN INTEGER - PROGRAMMING Problem.

INTERESTING BECAUSE x_i's ARE DISCRETE (0 OR 1)

THE RELAXATION, $0 \leq x_i \leq 1$, FRACTIONS ALLOWED

EASY TO SOLVE GREEDILY

POLITICAL EXAMPLE

ELECTORAL COLLEGE

100
+43
+3(DC)

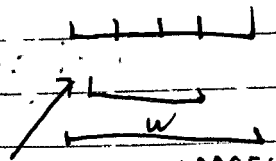
$W = \frac{538}{2} = 269$

WE CAN CONTAIN $\sum_{i=1}^{50} w_i = 269$

IFF A TIE IS POSSIBLE

GREEDY WON'T OBVIOUSLY WORK

Ex:



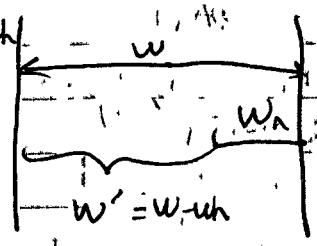
PICKING LARGEST WT. FIRST FAILS

USE DYNAMIC PROGRAMMING

CONSIDER LAST WT w_n

IF WE DON'T USE IT, WE MUST SOLVE A PROBLEM W/ $w_1, \ldots, w_n$ SAME CAP W

IF WE USE IT, SOLVE W/ $w_1, \ldots, w_{n-1}$ CAPACITY $W' = W - w_n$



(n, w)

$(n-1, w)$ $(n-1, w')$

CHOOSE BEST OF THE 2 ALTERNATIVES

CAN USE $w_1, \ldots, w_i$ CAP w

$$OPT(i, w) = \begin{cases} OPT(i-1, w) & \text{IF } w_i > w \\ \text{ELSE} \quad \max \text{ of } \begin{cases} OPT(i-1, w) \\ OPT(i-1, w - w_i) + w_i \end{cases} \end{cases}$$

BOUNDARY CONDITIONS

$OPT(0, w) = 0$, ALL w

$OPT(i, 0) = 0$, ALL i

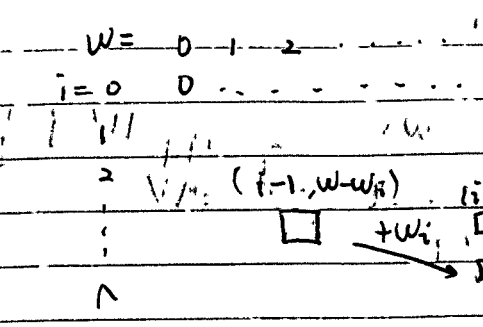
THIS REFERS TO $OPT(i, w)$, $0 \le i \le n$, $0 \le w \le W$

SHOULD SOLVE

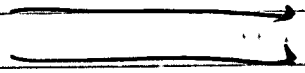
$(n+1)(W+1) = O(nW)$

SUBPROBLEM

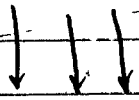
FILL IN A 2-D ARRAY



GO ROW BY ROW



OR COL BY COL



OPT IS COMPUTE, WE CAN BACK-TRACE TO SET THE OUT : CHOICE

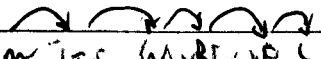
RUNTIME is O(n W)

NOT POLY IN BIT LENGTH

$W = 2^{\log W}$

PSEUDO-POLY TIME

FINISH KNAPSACK PROBLEM



JAM: MEET IN THE MIDDLE

THE "LEAP"

REVIEW:

Subset sum

given weights $w_1, \ldots, w_n$ capacity W } POS INT'S

$$\max \sum_{i \in S} w_i \text{ s.t. } \sum_{i \in S} \leq W$$

$$S \subseteq \{1, \ldots, n\}$$

DP SOLN

FOR ALL $i \leq n$

FOR ALL $w \leq W$

Find BEST SUM $\leq w$

USING WTS $w_1, \ldots, w_i$

Ex: WTS, 1, 2, (3)

CAP: 4

	w = 0	1	2	3	4
i=0	0	0	0	0	0
1	0	1	1	1	1
2	0	1	2	3	3
3	0	1	2	3	4

copy; $+w_1$; copy; copy; copy

$= \max\{1+3, 2\}$

$= 4$

TIGHT

OPT = 4

RECOVER A SET OF OPT VALUE BY BACK-TRACING

soln is $S = \{3, 1\}$

Q: WHAT IF WTS ARE REAL?

NO KNOWN P-TIME ($in n$) ALG TO SOLVE SUBSET SUM OVER $\mathbb{R}$, USING $+, \leq$

REASON: SUBSET SUM IS NP-COMPLETE.

DP SOL'N (FOR INTEGER VERSION) TAKES $O(nW)$ STEPS

$W = 2^{\log_2 W} \neq \text{poly}(\log W)$

KNAPSACK

WEIGHTS $w_1, \dots w_n > 0$

VALUES $v_1, \dots, v_n > 0$

CAPACITY $W > 0$

GOAL:

$$\max_{S \subseteq \{1,\dots,n\}} \sum_{i \in S} v_i \quad \text{s.t.} \sum_{i \in S} w_i \leq W$$

(SUBSET SUM: $w_i = v_i$ $\forall i$)

Fractional version:

$$\max \sum_{i=1}^{n} x_i v_i \quad \text{s.t.} \sum_{i=1}^{n} x_i w_i \leq W$$

$$0 \leq x_i \leq 1$$

HAS A GREEDY SOLN

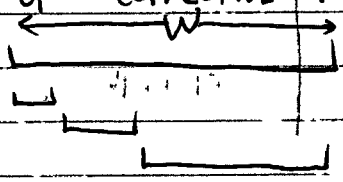
(DANTRIG 1957) ✓ $O(n \log n)$

SORT BY UNIT PRICE $= \frac{\text{value}}{\text{wt}}$ (LARGEST FIRST)

WHILE CAPACITY REMAINS.

① TAKE AS MUCH OF NEXT ITEM AS WILL FIT.

PF: OF CORRECTNESS.



IF A PRICER ITEM WASN'T USED, WE CAN SWAP.

↑ INCREASE TOTAL VALUE.

EXAMPLE

$i =$	1	2	3
$w_i =$	1	2	3
$v_i =$	2	5	1

$W = 4$

$\frac{v_i}{w_i}$ 2 5/2 1/3

USE ALL OF ITEM 1

	wt.	value
	2	5
	2	3
1/3	3	$7\frac{1}{3}$
	W = 4	

INTEGER KNAPSACK

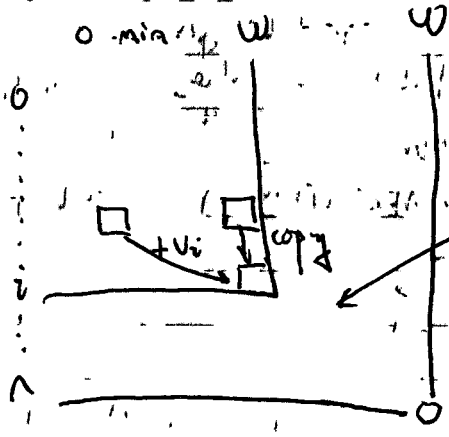
SAME, BUT $x_i \in \{0, 1\}$ $\forall$ $v_i, w_i, W > 0$ INTEGERS

(ITEMS INVISIBLE)

THESE ARE MAXMIZING PROB'S SO) RELAXATION (IMPORTANT)

OPT FOR INTEGRAL KNAPSACK $\leq$ OPT FOR FRACTIONAL KNAPSACK

APPALE SOLVES ← THIS IN $O(nW)$ STEPS



FOR w_i: 2 3

v_i: 2 5

$W = 4$

OPT value $= 7 \leq 7\frac{1}{3}$

WKT WED

TODAY: "STRUCTURAL" DYNAMIC PROGRAMMING

VERTEX COVER ON TREES

INDUCTIVE DEFN'S

E.G. 0 is a NAT NO. The successor of ANY NAT NO IS also a NAT NO. (NOTHING ELSE IS)

WE CAN PROVE FACTS ABOUT IN BY INDUCTION.

COURSE-OF-VALUES INDUCTION

INDUCTION: FORM OF PF MIRROR DEFN.

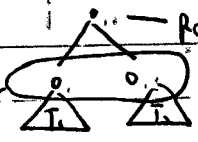
TREES: IN GRH THEORY, A TREE IS A CONNECTED GPH $G=(V,E)$ WITH $|E| = |V| - 1$ (m, n)

INDUCTIVE DEFN: OF ROOTED TREES.

(ONE VERTEX) is A TREE.

IF $T_1, \ldots, T_n$ ($n \geq 1$) ARE TREES.

So is



ROOT of T_1, T_2

WE CAN USE THIS DEFN IN AN INDUCTION PF

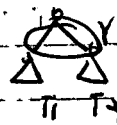
(STRUCTURAL INDUCTION)

THM: EVERY TREE w/ n VERTICES HAS $n-1$ EDGES.

BASE CASE: HAS $n-1$

$n=0 = 1-1$ ✓

INDUCTIVE STEP: CONSIDER



T_i HAS n_i VERTICES, $n_i - 1$ EDGES

$i = (n_j \geq 1)$

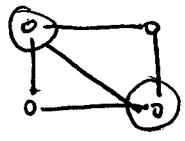
(BY INDUCTION) # of EDGES in T is $r + \left(\sum_{i=1}^{r} n_i - 1\right) = \left[\left(\sum_{i=1}^{r} n_j + 1\right)\right] - 1$

OF VERTICES IN T

EX FOR YOU:
PROVE ALL TREES ARE CONNECTED

DEFN IF $G = (V, E)$ IS A GRAPH, A VERTEX COVER IS A SET $S \subseteq V$ S.T. EACH EDGE HAS AT LEAST ONE END IN S

E.G A ART GALLERY

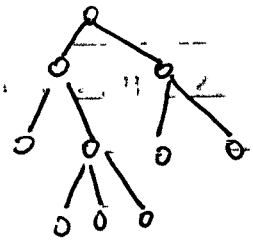


HALLS = EDGES
S = GUARDS
(ON VERTICES)

GOAL: FIND A VC OF MIN SIZE
HERE $|S| = 2$ IS MINIMAL

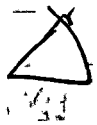
CURRENTLY, VC HAS NO KNOWN POLY (in m, n) - TIME ALG.

WE CAN SOLVE VC IN P-TIME FOR TREES
(USE DYNAMIC PROGRAMMING ON TREES)



IN A ROOTED TREE, EVERY VERTEX SPECIFIES A SUBTREE.
OUR DP ALG SOLVES VC FOR EVERY SUCH SUBTREE (BOTTOM-UP).

CONSIDER T =



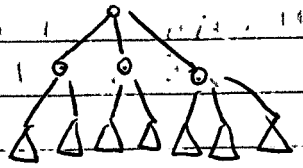
SO THE BEST VC EITHER USES r (ROOT), OR NOT.

IF YES
BEST S IS

S_v $v \cup \bigcup S_w$

w IS A CHILD of v $\quad S_w$ = AN OPTIMAL COVER FOR w's subtree.

IF $v \notin S$



MUST USE ALL CHILDREN OF v, OPTIMALLY COVER EACH TREE ROOTED AT A GRANDCHILD

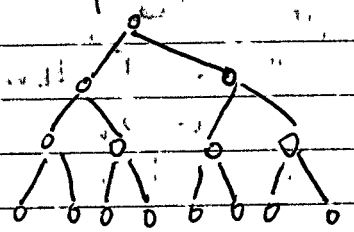
IN THIS CASE:

$$S_v = \{ w \mid w \text{ is a child of } v \} \cup \bigcup_{x_i} S_x$$

x_i x_i IS A GRANDCHILD OF v

FOR AN OPTIMUM COVER, v's subtree = smallest S_v FOR THESE

2 options ($v \in S_v$, $v \notin S_v$)



$|OPT| = 5$

RUN TIME is $O(n)$

STORE (FOR EACH VERTEX)

A BIT INDICATING IF it's IN COVER FOR THAT subtree

cost to cover its subtree

EACH EDGE is USED $\leq$ TWICE

$\Rightarrow O(m) = O(n)$

WK7 Fri

TODAY: LONGEST INCREASING SUBSEQUENCE

GIVEN A SEQ of n #'s

n=7 3 1 4 2 5 9 6

Goal: FIND AN INCREASING SUBSEQ of max length

Note: skip ALLOWED, E.G. 1 < 2

BRUTE FORCE:

2^n sub seq's

Better IDEA:

Solve A shortest path problem.

MAKE A DAG

VERTICES $s, x_1, \dots, x_n, t$

(INPUT SEQ)

EDGES:

	LENGTH
$s \to x_i$ (ALL i)	0
$x_j \to t$ (ALL j)	0
$x_i \to x_j$ (ALL $i<j$ AND $x_i < x_j$)	-1

ANY s–t path of length $i+2$ HAS AN INCR SUBSEQ OF LENGTH k

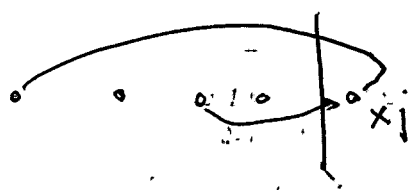
$s \to x_{i_1} \to x_{i_2} \to \dots \to x_{i_k} \to t$

reduction: Shortest path ($s=t$) = longest incr subseq

SINCE GPH is ACYCLIC, DYNAMIC PROGRAMMING WILL WORK

GO THRU VERTICES IN ORDER $s, x_1, x_2, \dots, x_n, t$ (x_0, x_{n+1})

dist (s) = 0,



$$dist(x_j) = \min_{\substack{i<j \\ x_j < x_i}} \{ dist(x_i) \}$$ length of $x_i \to x_j$: -1

$L_3 = 2$

3 1 4 2 5 9 6

CLEAN DP FOR LIS

Computed L_j = Length of a ~~longer~~ longest INCR SUBSEQ ENDING AT x_j

$$L_j = \begin{cases} 1 & \text{IF } j=1 \\ \max_{i<j,\ x_i \le x_j} L_i + 1 \end{cases}$$

EVAL 1 FOR $j = 1, n$

RUN TIME $O(n^2)$

3 1 4 2 < 5 < 9 6

L_i 1 1 2 2 3 (4) 4

|LIS| = 4

BACK TRACING FINS A LIS IN $O(n)$ TIME

1 2 5 9

~~SOLITAIRE~~ (PATIENCE)

RULES: ① A CARD MAYBE placed ON TOP OF a larger one

② CAN START A NEW PILE

GOAL: MAK SIMPLIEST # OF PILE

GREEDY STRATEGY.

CHOOSE LEFTMOST PILE (IF POSSIBLE)

START A NEW ONE (IF NOT POSSIBLE)

3 4 5 9

(LIS) 1 2 6

TAM (HAMMERSLEY)

MAX INCR = MIN # OF PILES

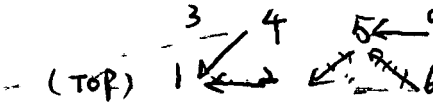
SUBSEQ LENGTH

(A STRONG DUALITY THM)

THE GREEDY STRATEGY FINDS BOTH

TO MAKE A LIS:

AT PLACEMENT TIME MAKE EACH CARD POINT TO top of each pile.



START w/ TOP CARD IN last pile, chase pointers.

A LIS is $1 < 2 < 5 < 6$

THIS CAN BE DONE IN TIME $O(n \log n)$

IDEA: BINARY SEARCH FOR CORRECT PILE.

WK 8, Mon

EDIT DISTANCE
SEQUENCE ALIGNMENT } KT 6.6

WHEN ARE 2 STRINGS SIMILAR? HOW SIMILAR?

Ex LOOKING FOR COGNATES IN WRITTEN TEXT

GUILLAME LE BÂTARD
WILLIAM THE CONQUEROR

IDEA: count disagreement IF x, y SAME LENGTH ← HAMMING DISTANCE

INTRODUCE 3 EDIT OPN'S

1. CHANGE A SYMBOL INTO ANOTHER
2. INSERT A SYMBOL
3. DELETE "

DEFN: THE EDIT DISTANCE BTWN x, y = MIN # OF OPN'S ↓ needed to TURN x into y.

FOR OUR EX.

CHANGE G TO W
DELETE U } dist (x, y) ≤ 3 (optimal)
DELETE E

$d(x,y)$ = DIST FROM x TO y $\leq \max\{|x|, |y|\}$

SATISFIES:

$d(x,y) \geq 0$, $=0$ only if $x=y$

$d(x,y) = d(y,x)$

$d(x,y) \leq d(x,z) + d(z,y)$ TRIANGLE INEQ

NOT | OBVIOUS HOW TO DETERMINE $d(x,y)$

USE DYNAMIC PROGRAMMING

PENALTIES FOR EACH EDIT OPN

K$T OSD δ FOR INSERT / DELETE

$\alpha_{a,b}$ FOR CHANGE a to b

TODAY:

ALL PENALTIES ARE 1.

DP IDEA:

COMPUTE EDIT DISTS BTWN PREFIXES OF x, y

CONSIDER $x_1, x_2, \ldots x_i$ into $y_1, \ldots y_j$

FIND A LEAST-COST TRANSF BTWN THESE

$OPT(i,j)$ = MIN COST FOR $x_1, \ldots x_i$ TO $y_1, \ldots y_j$

FOCUS ON FINAL SYMBOLS

WE COULD:

A) CHANGE x_i TO y_j (IF NEEDED)

$x_1, \ldots, x_{i-1} \rightarrow y_1, \ldots y_{j-1}$

B) $x_1, \ldots x_i \rightarrow y_1, \ldots y_{j-1}$, APPEND y_j

C) DELETE x_i, $x_1, \ldots, x_{i-1}$ into $y_1, \ldots y_j$

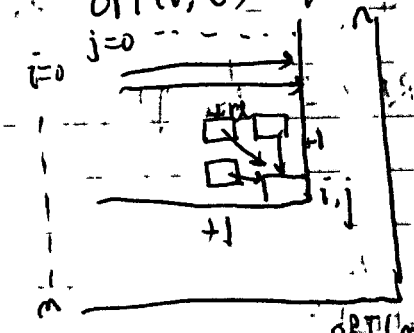
$\alpha = \begin{cases} 0 & \text{IF } x_i = y_j \\ 1 & \text{IF } x_i \neq y_j \end{cases}$

$$OPT(i,j) = \min \begin{cases} \alpha + OPT(i-1, j-1) & \text{FOR A)} \\ OPT(i, j-1) + 1 & \text{B)} \\ OPT(i-1, j) + 1 & \text{C)} \end{cases}$$

BOUNDARY CONDITIONS:

$OPT(0, j) = j$

$OPT(i, 0) = i$



$OPT(m, n) = dist(x, y)$

		T	R	U	M	P
	0	1	2	3	4	5
C	1					
R	2					
U	3					
Z	4	4	3	2	2	3

dist 3.

TRUMP

CRUZ

ALIGNMENT

WK8 WED

TODAY: EDIT DIST

SEQ ALIGNMENT

LONGEST COMMON SUBSEQ

SHORTEST PATHS (ARBITRARY LENGTHS)

EDIT DIST

$X = x_1, x_2, \ldots x_m$

$y = y_1, y_2, \ldots y_n$

dist (x, y) = min # of EDITOPN IT (CHANGE INSERT, DELETE, ONE SYMBOL)

To change x to y

Dist = 4

WE DON'T HAVE P, R, C, E

IN KING, SO 4 is optimal.

O(mn) DP ALG:

Compute, FOR $0 \le i \le m$, $0 \le j \le n$

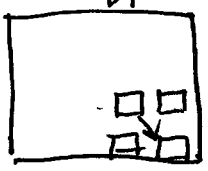
dist $(x_1, \ldots x_i, y_1, \ldots y_j)$

$$= \min \begin{cases} \text{dis}(x_1, \ldots x_{i-1}, y_1, \ldots, y_{j-1}) + \alpha, & \alpha = 0 \text{ IF } x_i = y_j \\ & 1 \text{ IF } x_i \neq y_j \\ \text{dis}(x_1, \ldots x_i, y_1 \ldots y_j) + 1 \\ \text{dis}(x_1, \ldots x_i, y_1 \ldots y_{j-1}) + 1 \end{cases}$$

WE CAN FIND

THE OPTIMAL EDIT

SEED BY BACK-TRACING

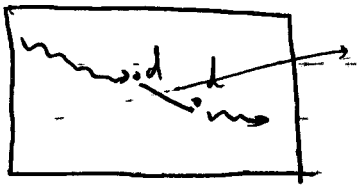


USE THE PREDECESSOR THAT ATTAINING THE MIN

DEFN: IN AN EDIT SEQ THAT CHANGES $x_1, \ldots x_m$ TO $y_1, \ldots y_n$, UNCHANGED SYMBOLS ARE CALLED ALIGNED.

PRINCE
KING

2 ALIGNED SYMBOLS



steps that don't change dist, correspond to ALIGN PAIRS

NOTE: A MINIMAL EDIT SEQ NEED NOT MAXIMIZE THE # OF ALIGNED PAIRS.

A B C D X R
(4)

E X S T U V
(4)

ALIGNS 1 PAIR
(MAX PROCESSING)
cost = 8

BUT:

A B C D X R
↓ ↓ ↓ ↓ ↓ ↓
E X S T U V

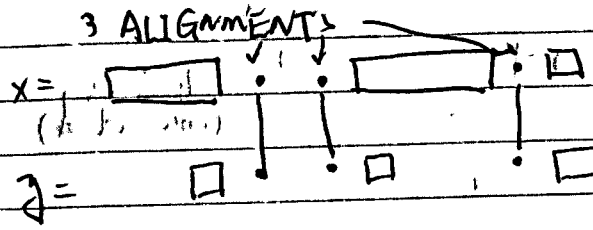
costs 6 (optimal)

TO MAXIMIZE # OF ALIGNED PAIRS,
CHANGE PENALTY FUNCTION:

SYMBOL CHANGE = +1

INSERT / DELETE = +1

3 ALIGNMENTS



IF WE HAVE k ALIGNMENTS, # OF DELETES = $|x| - k$

$\therefore$ # OF INSERTS = $|y| - k$

$\therefore$ EDIT COST

$= |x| + |y| - 2k$

TO FIND max # OF ALIGNED PAIRS WE MINIMIZE EDIT COST

A LONGEST COMMON SUBSEQUENCE IS JUST THE SUBSEQ OF ALIGNED PAIRS FOUND BY THE EDIT DIST ALG W/ MODIFIED PENALTIES

TO FIND A LONGEST INCREASING SUBSEQ IN $x = x_1, x_2, \ldots x_n$

SORT x (INCREASING), TO MAKE $y = y_1 \ldots y_n$

FIND A LONGEST COMMON SUBSEQ BETWEEN x AND y

x: 3 1 4 1 2

y: 1 2 3 4

BEST BECAUSE ANY LENGTH 4 SUBSEQ HAS A DECREASE

SHORTEST PATHS

$G = |V, E|$ $|V| = n$ $|E| = m$

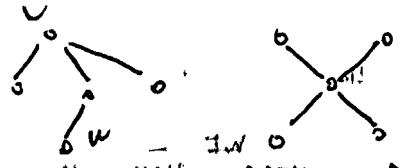
EACH EDGE HAS A LENGTH $\in \mathbb{R}$

WE CAN SOLVE:

1) LENGTHS ≥ 0 DIJKSTRA $O(m \log n)$

2) DIRECTED ACYCLIC GRH — $O(n)$ IF G is (connected)

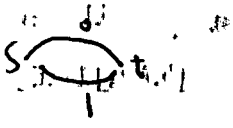
3) ACYCLIC, UNDIRECTED GRH



SO $\forall u, w$ $\exists$ ONE PATH OR NONE

USE BFS/DFS $O(m+n)$

SHORTEST PATHS NEED NOT EXIST. s.t. COSTS 0



s.t.s.t

s.t.s.t

WK 8 Fri FINISH SHORTEST PATHS

$G = (V, E)$ DIRECTED GPH

SOURCE $s \in V$ CAN REACH all vertices $v \in V$

EDGE LENGTHS $l_{v,w} \in \mathbb{R}$

CONVENTION: $l_{x,x} = 0$

THM: Shortest PATHS (from s) exist iff G Has no negative cycles. PF: exercise

Bellman's EQN's

ASSUME No NEG CYCLES

d_v = min length of AN s-v path

$d_v = \min_{w \in V} \{ d_w + l_{w,v} \}$, ALL $v \in V$

expresses A principle of optimality in A min-length s-v path

s ⋯ w → v

The s-w part is itself mini-length $= d$

FOR ANY VECTOR $\{d_v\}_{v \in V}$ $d_v \in \mathbb{R} \cup \{+\infty\}$

LET $F(d)_v = \min_{w \in V} \{ d_w + l_{w,v} \}$

BELLMAN EQN's SAY THE "CORRECT" DIST VECTOR IS A FIXED PT OF F

F: $F: X \to X$

a fixed PT is a sol'n to $F(x) = x$.

IF $F(x) = \frac{1}{2}(x + \frac{a}{x})$ $a > 0$

$\sqrt{a}$ is A FIXED PT OF F

PF $\frac{1}{2}(\sqrt{a} + \frac{a}{\sqrt{a}}) = \sqrt{a}$

very effective as a method to compute $\sqrt{a}$.

x_0 = initial guess

REPEAT $x_k = \frac{1}{2}(x_{k-1} + \frac{a}{x_{k-1}})$ } Fix point iteration (very common)

until $|x_k - x_{k-1}| < \varepsilon$

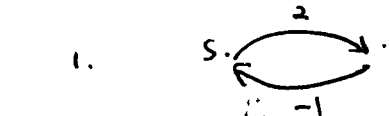
Ballman-ford ALG:

TRY TO SOLVE B's EQNs BY FIXPT ITERATION

$d^{(0)} = (0, \infty, \infty, \ldots, \infty)$

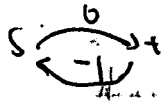
$d^{k+1} = F(d^k)$, $k \geq 0$

$(d^k)_v = \min_{w \in V} \{ d_w^{k-1} + l_{w,v} \}$ $k = 1, 2, 3$

ex: 1. 

$d^{(0)}$	$= 0$	$+\infty$
$d^{(1)}$	$= 0$	2
$d^{(2)}$	$= 0$	2

} stabilize

Ex 2: 

$d^{(0)}$	$= 0$	$+\infty$
$d^{(1)}$	$= 0$	6
$d^{(2)}$	$= -1$	0
$d^{(3)}$	$= -1$	-1

↓ GOES FOREVER

THM (CORRECTNESS OF B-F)

IF $d^{(n)} = d^{(n-1)}$: shortest paths exist, $d^{(n-1)}$ = dist from s to v

IF $d^{(n)} \neq d^{(n-1)}$, $\exists$ A NEG CYCLE

IF $|V| = n$, $|E| = m$ cost per update is $O(n)$

TOTAL $\underline{\underline{O(mn)}}$

CAN FIND MIN LENGTH PATHS OR A NEG CYCLE BY BACK TRACING

LEMMA: $\forall k \geq 0$

$d_v^{(k)}$ = min length of A N s-v sequence (can repeat vertices) using $\leq$ k "Hops" (edge or stay put)

PF: INDUCTION ON k

$s \rightsquigarrow w \rightarrow v$ } EXPAND

$\exists\ d_w^{(k-1)}$

PF OF THM

IF d^{n-1} IS A FIXED PT, $d^{(k)} = d^{(n-1)}$, $\forall k \geq n-1$

so $d_v^{(n-1)}$ = length of best path to v.

$\Rightarrow$ No NEG

IF NOT, $d^{(n)} \neq d^{(n-1)}$

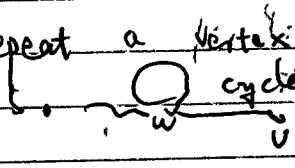
$d_v^{(n)}$ can't increase

$\exists$ $v \in V$ s.t. $d_v^{(n)} < d_v^{(n-1)}$

SEQ GUARANTEE BY USES n EDGES.

LEMMA

& must repeat a vertex w



IF WE OMIT THE CYCLE THRU, v GETS LONGER.

$\therefore$ length of cycle $C \leq 0$.

SP's Don't Exist ☒

WK 9 MON

RANDOMIZED ALG'S chapter 13 of K&T

TODAY: collision resolution (K&T 13.1)

TRADITIONAL ALG'S ARE DETERMINISTIC.

OUTPUT (IF ANY) FUNCTIONALLY DEPEN'S ON INPUT

BUT PROCEDURES

Do use RANDOM CHOICES.

- GAMBLING
- POLITICAL POLLING
- TAX AUDITING

1970's: RANDOMIZATION IN CS

WHY?! ELEGANT RANDOMEED ALG'S

(EG: PRIME TESTING)
NON DETERMINISM (EG P NP)

PROBABILITY:

S = SAMPLE SPACE (PRIMITIVE OUTCOMES)

EG: DRAW OF 1 (card)

P_x (FOR) $x \in S$) = PROB

x is observed IN ONE TRIAL

$0 \leq P_x \leq 1$

FINITE OR COUNTABLE

A NUMERICAL FN on S is called a random variable

Its Expectation is $\sum_{i \in S} f(x) P_x = E[f]$

A RANDOMIZED ALG IS A COMPUTATIONAL PROCEDURE THAT CAN MAKE RANDOM CHOICES (EG COIN FLIPS)

COLLISION RESOLUTION

n PROCESSES WANT ACCESS TO A PIECE OF DATA

COULD

WHO GETS IT?

$P = P_{rob}$ [P_i makes A REQUEST]

$0 \leq P_i \leq 1$

$S = \{0, 1\}^n$ (0 = NO RES, 1 = REQ)

$P_R[x_1 x_2 \ldots x_n] = P^r (1-P)^{n-r}$

r 1's

n-r 0's

EG. n=2

		ACCESS
0	0	$(1-P)^2$
1	0	$P(1-P)$ P_1
0	1	$(1-P)P$ P_2
1	1	P^2

RANDOM $\leq n$ possible requests
[≤ 1 FROM EACH PROCESSED]

COLLISION > 1 PROC REQUESTS

MECHANISM GRANT REQUEST IFF EXACTLY ONE PROC DEMANDS IF

$\forall i$

$Pr[P_i \text{ GAINS ACCESS}] = p(1-p)^{n-1}$ ← KIT claim MAXIMIZED

LET $p = \frac{1}{n}$ ⇒ $p = \frac{1}{n}$

$Pr[P_i \text{ gains access}] = \frac{1}{n}(1-\frac{1}{n})^{n-1}$

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

AS $n \to \infty$ $\frac{n-1}{n} \to 1$

$(1-\frac{1}{n})^n \to e^{-1} \to \frac{1}{n}e^{-1} = \Theta(\frac{1}{n})$

SUPPOSE REQ's ARE MADE IN ROUNDS

How Long most P_i wait to gain access?

# of rounds

WAITING FOR 1ST SUCCESS.

(H IN SEQ OF COIN FLIPS)

EG: TTHT HTHHH

LET $\pi = Pr[H]$, $1-\pi = Pr(T)$

$Pr[(k-1)T\text{'s THEN } 1H] = (1-\pi)^{k-1}\pi$

WAIT = 6

$E[WAIT] = \sum_{k=1}^{\infty} k(1-\pi)^{k-1}\pi$ $\quad Pr[WAIT = k]$

$x = 1-\pi$

$= \frac{\pi}{(1-(1-\pi))^2} = \frac{\pi}{\pi^2} = \frac{1}{\pi}$

RECALL $1 + x + x^2 + x^3 + \cdots \quad \frac{1}{1-x}$

$\frac{d}{dx}(\frac{1}{1-x}) = \frac{1}{(1-x)^2}$

$1 + 2x + 3x^2 + \cdots$

$\sum_{k=1} k x^{k-1}$

IF PR [P_i succeeds] $= \Theta(\frac{1}{n})$
(once)

E [WAIT FOR P_i] $= \Theta(n)$

TODAY: CONTENTION Resolution } FROM performance ANALYSIS ← CS(47
K&T 13.

PROCESS P_i, $i = 1, \ldots, n$
want to use a piece of memory
exclusive access ⇒ 1 PROC is GRANTED ACCESS.
round-robin allocation
GRANT ACCESS 1-TO:
$P_1, P_2, P_3, \ldots, P_n, P_1, P_2, \ldots$
WAIT FOR ACCESS $\leq n$ (means clocks)

RANDOMIZED ALLOCATION
EACH PROC TRIES TO GAIN ACCESS W.Pr. $\frac{1}{n}$
(simultaneously, independently)

Last time:
For each i
E [TIME until i can access] $\sim e = 2.71828\ldots$

The "price of ANARCHY" is a const-factor slowdown
(no clock)

Starvation

$\Pr[\exists\ i$ P_i gains access$] = \Pr[\bigcup_{i=1}^{n} \{P_i \text{ gains access}\}]$ DISJOINT

$\leq \sum_{i=1}^{n} \Pr[P_i \text{ gains access}]$

$= \sum_{i=1}^{n} \frac{1}{n}(1-\frac{1}{n})^{n-1} = \frac{n}{n}(1-\frac{1}{n})^{n-1} \approx e^{-1} \simeq 0.367$

FOCUS ON TIMES OF SUCCESSFUL REQ's (FOR SOMEBODY)

PROC #'s = SEQ of RANDOM DRAWS (w/ REPLACEMENT) FROM $(1, 2, \ldots, n)$

W = 1st # TIME AT WHICH WE'VE DRAWN ≥1 COPY OF EACH #

E.G. $n = 3$

$2, 3, 3, 2, 3, 1, \ldots$

$W = 6$

coupon collection.

WHAT IS $E[W]$?

$W = \sum_{i=1}^{n} W_i$ $\quad W_i$ = # of draws used to go from $(i-1)$ different coupon to i

I'm waiting to get one $n-(i-1)$ "useful" coupons

π_i = PR[get a new coupon time] $= \frac{n-i+1}{n}$

LAST TIME $E[W_i] = E[\text{WAIT FOR 1st success, if } Pr[\text{success}] = \pi_i]$

$= \frac{1}{\pi_i} = \frac{n}{n-i+1}$

$E[W] = E\left[\sum_{i=1}^{n} W_i\right] = \sum_{i=1}^{n} E[W_i]$ ✓ linearity of expectation

$= \sum_{i=1}^{n} \frac{n}{n-i+1} = n\left(\frac{1}{n} + \frac{1}{n-1} + \cdots + 1\right)$

H_n ; $\sim n \ln n$

ONE CAN SHOW

$H_n \sim \ln n$ (compare w/ $\int_1^n \frac{dx}{x} = \ln n$)

ON AVG, $\sim \frac{1}{2}$ of the rounds will grant access to ANY ONE.

LEADS TO

$E[\#$ Rounds FOR ALL PROC's to access memory $] \sim n \ln n$

NEXT PROVE A TAIL BOUND

$PR[\#$ Rounds need FOR ALL Proc's to gain access $] > 2[1+\varepsilon] n \ln n] \leq \frac{1}{n}$

QuickSort

$x_1, \dots x_n$ DISTINCT

CHOOSE ONE INPUTS x AS THE SPLITTER

GROUP KEYS SO THAT

$L = x_1, \dots, x_{i-1} < x$

$x = x_i$

$R = x_{i+1}, \dots x_n > x$

} Compare all other keys to x

$n-1$ comp's

Recursive by sort L, R

Ex: $n = 7$

$=x$

7 2 5 1 4 3 6

2 1 4 3 7 6

2 1 3 4

1 3

CAN BE DONE IN PLACE

depth = 3

perfect splitting. For $n = 7$ has depth = 2

A WORST-CASE # of comps is sorted keys $(\binom{n}{2})$

$x = \min\{x_i\}$

$L = \emptyset$ $|R| = n-1$

Total # of comp's
is $(n-1) + (n-2) + \cdots + 2 + 1 = \frac{n(n-1)}{2} = \binom{n}{2}$

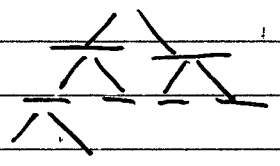
SAME AS COMPARING ALL PAIRS

RANDOMIZED QUICKSORT

CHOOSE x randomly from $x_1, \ldots, x_n$

$P_r[x = i\text{th key}] = \frac{1}{n}$

INTUITION: SPLITS WILL TEND TO BE EVEN.



depth $\approx \log n$

GOAL: # of comp's is a random vble compute its expectation.

~~Compact~~ conditional expectation.

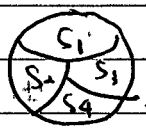
sample space $S = S_1 \cup S_2 \cup \cdots \cup S_n$ (DISJOINT)

f = RANDOM VBLE

$$E[f] = \sum_{x \in S} f(x) P_x = \sum_{i=1}^{n} \sum_{x \in S_i} f(x) P_x$$

$$= \sum_{i=1}^{n} \left(\sum_{x \in S_i} f(x) \frac{P_x}{P_r[S_i]} \right) P_r[S_i]$$

$E[f \mid S_i]$



PROB of x, Given C_i

Exp of f: Given S_i

very useful

S_i = event that x_i (top level splitter) has rank i $1 \le i \le n$

$P_i[S_i] = \frac{1}{n}$ min = rank 1

max = rank n

$E_n = E[\#\text{ comp's ON } n \text{ KEYS}]$

$E_n = \sum_{i=1}^{n} E[\#\text{ Comp's} \mid \text{rank } i] \cdot P_r[\text{rank} = i]$

If rank $= i$, $|L| = i-1$

$|R| = n-i$

So $E_n = \sum_{i=1}^{n} \left((n-1) + E_{i-1} + E_{n-i} \right) \frac{1}{n}$

↑ linearity of expectation.

$$\sum_{i=1}^{n} (n-1) + \frac{1}{n} \left[\begin{matrix} E_0 + E_1 + \cdots + E_{n-1} \\ E_{n-1} + E_{n-2} + \cdots E_0 \end{matrix} \right] \quad E_0 = E_1 = 0$$

$$E_n = (n-1) + \frac{2}{n} \sum_{i=0}^{n-1} E_i \,, \quad E_0 = E_1 = 0$$

SOLVE BY DIFFERENCING

$$n E_n = n(n-1) + 2 \sum_{i=0}^{n-1} E_i$$

$$(n-1) E_{n-1} = (n-1)(n-2) + 2 \sum_{i=0}^{n-2} E_i$$

$$n E_n - (n-1) E_{n-1} = 2(n-1) + 2 E_{n-1}$$

$$\Rightarrow n E_n = (n+1) E_{n-1} + 2(n-1)$$

$$\frac{E_n}{n+1} = \frac{E_{n-1}}{n} + \frac{2(n-1)}{n(n+1)}$$

so $\frac{E_n}{n+1} = \sum_{i=1}^{n} \frac{2(i-1)}{i(i+1)}$

NOTE THAT ith TERM $\sim \frac{2}{i}$ (As $i \to \infty$)

$$\therefore \frac{E_n}{n+1} \sim \sum_{i=1}^{n} \frac{2}{i} = 2H_n \sim 2 \ln n$$

$$\Rightarrow E_n \sim 2n \ln n$$

$$\doteq 1.4 n \log_2 n = O(n \log n)$$

PRICE OF IMPERFECT SPLITTING

WK11 MON

TODAY: RANDOMIZED SORTING (FRI)
SELECTING BY RANK (TODAY) } KT 13.5

NEXT: STRING MATCHING (see course web page)

Quick sort (CAC HOARE)

~1962

$x_1, \ldots, x_n$ DISTINCT KEYS CHOOSE ONE KEY x (SPLITTER) GROUP INTO

L_i keys $< x$

$x = X_{i:}$

R_i : keys $> x$

return sorted $(L, x, \text{sort}(R))$

Base case: $n = 0, 1$ Do nothing return x

THM:

IF x is chosen uniformly at random from $x_1, \ldots, x_n$

EXPECTATED: # of key comp's $\sim 2n \ln n = 1.4 (\log n)^n$

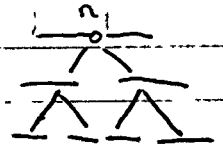
(PROVED LAST TIME)

INTUITIVE REASON:

x tends to be close to middle of the sorted list

so

DEPTH should be low



Selection BY RANK

DEFN: THE ORDERED statistics of $x_1, \ldots, x_n$ are $x_{(1)} = \min x_i$

$x_{(2)}$ = 2nd smallest x_i

BEFORE SORTING

IF A KEY IS kth from bottom $(= x_{(k)})$ $x_{(n)} = \max x_i$

its rank is k

After sorting

Selection problem: Given $x_{(1)}, \ldots, x_{(n)}$, $1 \le k \le n$

FIND $x_{(k)}$

Example: THE median key is $x_{(\lceil \frac{n}{2} \rceil)}$

with k find $x_{(k)}$.

cost: $O(n \log(n))$ key comp's

(MERGE sort)

~1975 LINEAR TIME = $(O(n))$
Selection algs:
Intricate DIVIDE & CONQUER

TODAY: RANDOMIZED $O(n)$ expected # of comp's quicksort's
THE IDEA is to go down one branch of recursion tree.

which branch?

Suppose input splits as $x_{(1)}, \dots x_{(i-1)}, x_{(i)}, x_{(i+1)}, \dots x_n$
L i R

$1 \le k \le n$

$$x_{(k)} = \begin{cases} k\text{th smallest of } L, & \text{IF } k < i \\ x_{(i)}, & \text{if } k = i \\ (k-i)\text{th smallest of } R, & \text{if } k > i \end{cases}$$

BASE CASE: $k = n = 1$
Return $x_{(1)}$

RANDOMIZED QUICK SELECT
LET $\{1, \dots, n\}$ ← choose i uniformly from $\{1, \dots, n\}$

SPLIT USING $x = x_i$
Go DOWN BRANCH THAT HAS $x_{(k)}$

THM: Quickselect uses $O(n)$ key comp's in expectation.

PF: PROB size
Strictly decreases w/ each rec call
($\Rightarrow$ ALL terminates)

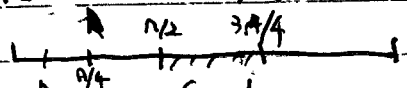
LET $c(\frac{3}{4})^{j+1} \le$
THE jth PHASE BE LEVELS FOR WHICH PROB SIZE
$\le n(\frac{3}{4})^j$
THE jth phase ENDS IF
NEW PROB $\le 3/4$ CURRENT PROBLEM

STRATEGY: PROVE THERE'S A "GOOD" CHANCE of this.

call x good

IF $|\text{NEXT PROB}| \leq 3/4 \cdot |\text{CURR PROB}|$

CONSIDER $k \leq n/2$



$\geq 1/4$ of x's ARE GOOD

$k \geq \frac{n}{2}$ SIMILAR

b/c IN ALL CASES

$\Pr(\text{Good splitter}) = \Pr(\text{PHASE ENDS}) \geq 1/4$ (LAST CLAIM $\geq 1/2$)

$E(\#\text{ of levels in a phase}) \leq 4$

(waiting time FOR FIRST "H" IF $\Pr["H"] \geq 1/4$ head, $\Pr["T"] \leq 1/4$)

$$E(\#\text{ comp's}) = \sum_{i=0}^{\wedge} E(\#\text{ comp's in } i^{th}\text{ phase})$$

$$\leq \sum_{j \geq 0} \left[\frac{3^j n}{4^j} \right] \leq n \cdot \left[\#\text{ of levels in } j\text{th phase} \right]$$

$$= 4n \cdot \frac{1}{1 - 3/4} = 4n = O(n)$$

WK 11 WED

RABIN KARP STRING MATCHING (see "course web pg")

String matching

Text t, eg Presidential

Pattern p ↑ p must be contionions

Does p occur in t (where?) Dental ✗ is not a substring

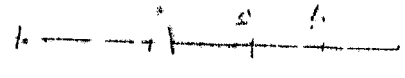
As a substring

Suppose: $|p| = m$, $|t| \leq n$

We could test for p at all possible places:

checking one place: $\leq m$ comps

checking all places $\leq n \cdot m$



Why not stop early?

worst case is not improved

$t = A, A, A \ldots A$

$p = A, A A B$

AA A B

Rabin - Karp (~1970!)

Randomized ALG

INFALLIBLY DETECTS MATCHES

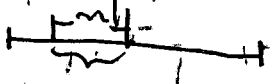
one-sided error

A PROB (small) of (FALSE POSITIVE)

IDEA: compute A FINGER PRINT (HASH VALUE)

For p.

For each position, compute the F.P. For that position



SAY "MATCH" IF FP's AGREE

"NO MATCH" IF THEY DON'T

R-K USES POLYNOMIAL HASHING

This CAN EVALUATE F.P. FOR EACH SUCCESSOR position w/ $O(1)$ work.

$\Rightarrow$ Total search = $O(m+n)$

THE FINGERPRINT:

choose a prime q

A base b, $1 < q < b$

SYMBOLS ARE NUMBERS, E.G. ASCII TEXT, $0 \leq c \leq 128$

THE FINGERPRINT OF $s_0, \ldots, s_{n-1}$

IS

$F(s_0, \ldots s_{n-1}) = (s_0 + s_1 b + s_2 b^2 + \cdots + s_{n-1} b^{n-1}) \bmod q$

WE CAN FIND F IN $O(n)$ steps

HORNER'S RULE (AKA SYNTHETIC DIV'N)

$1 + 2b + 3b^2 + 4b^3$

$= 1 + b(2 + 3b + 4b^2) = 1 + b(2 + b(3 + 4b))$

n +'s

n *'s

UPDATING

$s_0 + s_1 b + \cdots + s_n b^n = [s_0 + s_1 b + \cdots + s_{n-1} b^{n-1}] + s_n b^n$

$= F(s_0, s_1, \ldots s_{n-1})$

$= s_0 + b[s_1 + s_2 b + \cdots s_n b^{n-1}]$

$F(s_1, \ldots s^n)$

ALGEBRA $\Rightarrow$

$b^{-1}[F(s_0, \ldots s_{n-1}) + s_n b^n - s_0] = F(s_1, \ldots, s_n)$

↑ use this to update

ALL ARITHMETIC IS mod q

THM: If q is prime, $0 < b < q$, There is a b' s.t. $bb' \equiv 1 \pmod{q}$

PF: (constructive), use Euclid's ALG

SOLVE, $bx + qy = 1$ in $\mathbb{Z}$

get $x = b'$ mod q

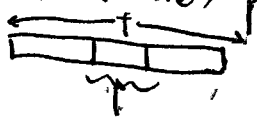
Wk 11 Fri

FINISH RABIN-KARP STRING MATCHING

n character pattern p

n character text t $(n \geq m)$

Determine IF (where) p occurs in t "(No skips in t)



character = integers in $0 \ldots q-1$

q is prime $1 < b < q$

base

FINGERPRINT OF $s_0 s_1 \ldots s_{n-1} = s_n$ is $F(s) = (s_0 + s_1 b + \ldots + s_{n-1} b^{n-1}) \bmod q$

overlapping strings have related fingerprints

$s_0 s_1 \ldots s_{n-1} s_n$

$F(s_1 \ldots s_n) = b^{-1}(F(s_0 \ldots s_{n-1}) + b^n s_n - s_0)$

$b \cdot b^{-1} \equiv 1 \pmod q$

Ex. A B C D E F G

1 2 3 4 5 6 7

$q = 11$, $b = 2$ $\quad b^{-1}$?

$2x \equiv 1 \pmod{11}$

$b^{-1} = x = 6$

$p = A\ B\ C \qquad F(p) = A + 2B + 2^2 C$

$= 1 + 2 \cdot 2 + 2^2 \cdot 3 \equiv 17 \equiv 6$

$t = A\ B\ A\ B\ A\ F$

$F(A\ B\ A) = 1 + 2 \cdot 2 + 1 \cdot 2^2 = 9$ No match

$F(B\ A\ B) = 6(9 + 8B - A)$

$= 6(2 + 16 - 1) = 6 \cdot 7 \equiv 1$ No match

$F(A\ B\ A) = 9$ (check)

$F(B\ A\ F) = 6(9 + 9F - A) = 6(9 + 48 - 1) = 6$ FALSE MATCH

To make false matches unlikely, q s/b large

How large

We want $N/2 \le q \le N$ (N chosen later)

USE:

THE PRIME NUMBER THM

$\pi(x)$ = # PRIMES $\le x$

(so $\pi(4) = \#\{2, 3, 5, 7\} = 4$)

As $x \to \infty$, $\pi(x) \sim \frac{x}{\ln x}$

($\sim$ MEANS $\lim_{x \to \infty}$ (RATIO) $= 1$)

NO. OF ELIGIBLE q

$\pi(N) - \pi(N/2) \sim \frac{N}{\ln N} - \frac{N}{2\ln(N/2)}$

$\le \#S'$

CONSIDER $S = s_0 s_1 \cdots s_{m-1}$ $\quad S' = s_0' s_1' \cdots s_{m-1}'$

$f = s_0 + s_1 b + \cdots + s_{m-1} b^{m-1}$

$f' = s_0' + s_1' b + \cdots + s_{m-1}' b^{m-1}$ $\Big\}$ $\in \mathbb{Z}$

$|f - f'| \le b^m$

of q, $q \mid (f - f')$

$1 \le \le \log_{N/2}(b^m) \sim \frac{m \ln b}{\ln N}$

suppose $q_1 \cdots q_r < b^m$ $\quad q_i \ge N/2$

$(\frac{N}{2})^r < b^m$ $\quad q_1 \cdots q_r < \log_{N/2}(b^m)$

PR[q makes S, S' a false match]

$\sim \frac{m \ln b}{\ln N} \cdot \frac{2 \ln N}{N} \sim \frac{2m \ln b}{N}$

Goal: expected # of false matches

- IS TO BE $O(1)$

CHOOSE N so that $\frac{2m \ln b}{N} \le 1$

i.e. $N \ge 2m \ln b$

LINEARITY OF EXPECTATION

$$E\left[\#\text{ of false matches}\right] = \sum_{i=1}^{n} Pr[\text{FALSE MATCH AT pos } i]$$

$$\leq \frac{2mn}{N} \leq 1$$

FINDING q

VERY EFFICIENT RANDOMIZED PRIME TESTS $O(\log q)$ $\times$ mod q

A SIMPLER TEST WILL DO.

REPEAT: [$q \leftarrow R[Np, N]$; TEST $|q|$ FOR PRIMALITY BY TRIAL DIVN] UNTIL q is prime.

cost to test q is $\sqrt{q} \leq \sqrt{N}$

E[work for this] $= O(\sqrt{N} \log N)$

IF $b=2$ $\leq O((mn)\log(mn))$

WK 12 Mon

TODAY: NETWORK FLOW KT 7.1

A USEFUL "PHYSICAL" MODEL

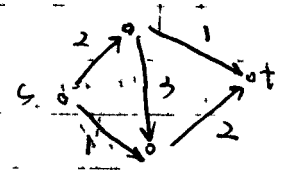
$G=(V,E)$ DIRECTED GPH

2 SPECIAL VERTICES

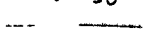
s (THE SOURCE) t (the sink)

EACH EDGE e has a capacity $c_e \geq 0$

Ex.



IMAGINE A FLUID FLOWING FROM s to T.

constraints Let $f(e)$ = FLOW THRU e 

1) FLOW $\leq$ CAPACITY $0 \leq f(e) \leq c_e$ $\forall$ all $e \in E$

2) MASS Conservation

IF $v \neq s,t$ FLOW INTO v = FLOW OUT OF v

KIRCHOFF'S CURRENT LAW

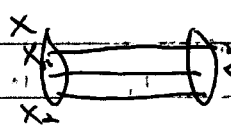
$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$$

If $f: E \to \mathbb{R}_{\geq 0}$ satisfies 1) & 2)
It's called A FLOW.

EXAMPLE: A) PLUMBING (2 WAY PIPES $\rightleftarrows$)
B) ELECTRICAL CIRCUITS (DC)
C) TRAFFIC IN A ROAD NETWORK.

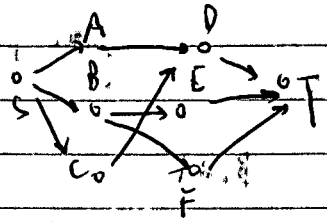
IN A) – C) FLOW is A IDEALIZATION.

D) BIPARTITE MATCHING (NON-OBVIOUS EXAMPLE)
$E \subseteq X \times Y$ Say $V = X \cup Y$ $(X \cap Y \neq \emptyset)$
A MATCHING Is A set of ~~pairs~~ EDGES showing NO END POINTS



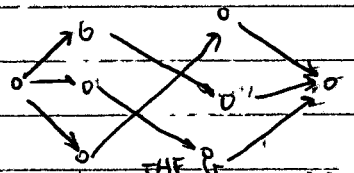
$\{(x_1, y_1), (x_2, y_2)\}$ is one
$\{(x_1, y_1), (x_1, y_2), (x_3, y_3)\}$ isn't

WE CAN MAKE A FLOW NTWK BY ADDING NEW VERTICES s, t,

G. 

EACH MATCHING DETERMINES A FLOW

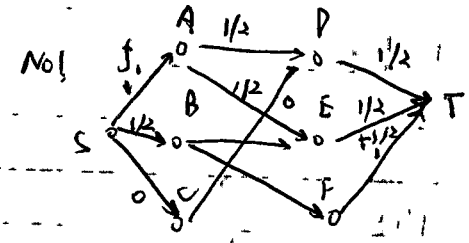
$M = \{(A, E), (B, F), (C, D)\}$



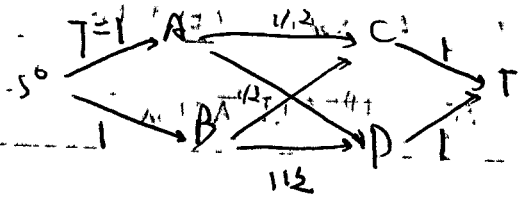
$f = 1$ ON THE $\overrightarrow{se}$ EDGE, 0 ELSE WHERE

SATISFIES 1) & 2)

DOES EVERY FLOW COME FROM A MATCHING?



HERE, FLOWS w/ $f = 0, 1$ on (s, x) AND (y, t) EDGES IS a fractional matching.



IF f IS A FLOW, ITS VALUE IS $v(f) = \sum_{e \text{ out of } s} f(e) = \sum_{e \text{ into } t} f(e)$

IF WE ASSUME NO EDGES INTO s

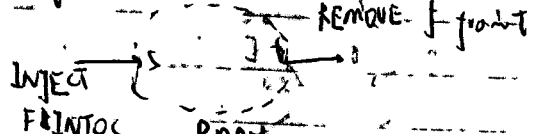
PROBLEM:

GIVEN A FLOW NTWK, FIND A FLOW OF MAXIMUM VALUE

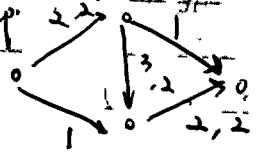
NET FLOW OUT of s = NET FLOW INTO t

ALWAYS

PHYSICALLY evident



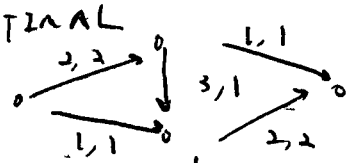
WE CAN TRY TO FIND A MAX FLOW BY SEVING FLOW ALONG PATHS.



VALUE = 2.

NO MORE PATHS CAN BE USED

NOT OPTIMAL

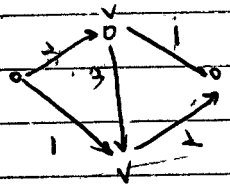


value = 3

WK 12 WED

TODAY: NETWORK FLOW
FORD-FULKERSON PROCEDURE K&T 7.1, 7.2

FLOW NTWS
DIGRAPH $G = (V, E)$
SOURCE SINK $s \in V, t \neq s \in V$
EACH EDGE HAS A CAPACITY $c_e \geq 0$
FLOW: REAL-VALUED FN f ON EDGES, s.t. 1) $0 \leq f(e) \leq c_e$
2) AT EACH $v \neq s, t$ flow into v = flow out of v
KIRCHOFF LAW.

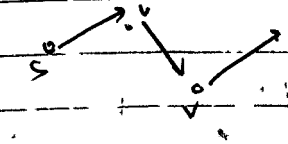


VALUE(f) $= \sum_{e \text{ out of } s} f(e) - \sum_{e \text{ into } s} f(e)$
NET FLOW OUT OF s
VALUE $f \leq \sum_{e \text{ out of } s} c_e$

GOAL: FIND A FLOW OF MAXIMUM VALUE.
(HAS TO EXIST.)
A KIND OF LINEAR PROGRAMMING (MAX. A LINEAR FN SUBJECT TO LINEAR $\leq$'s)
NOTE $a = b$
IF $a \leq b$, $b \geq a$

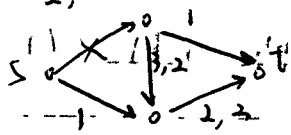
A SIMPLE GREEDY STRATEGY:
FIND AN s.t PATH P W. POSITIVE CAPACITY.
SEND AS MUCH FLOW AS P AS WE CAN
REPEAT

CAN STOP BEFORE MAXIUM FLOW IS REACHED. IN EX.



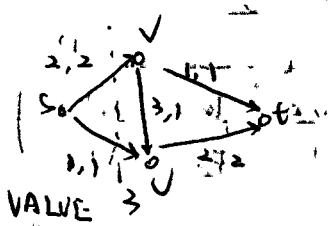
HAS CAPACITY 2.

cap, flow



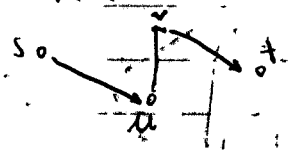
IF WE REMOVE THESE EDGES, NO MORE s-t PATHS, HALT W/ A FLOW OF VALUE 2, NOT OPTIMAL

$s \to v$ AND $v \to t$ HAVE NO MORE CAPACITY (SATURATED)

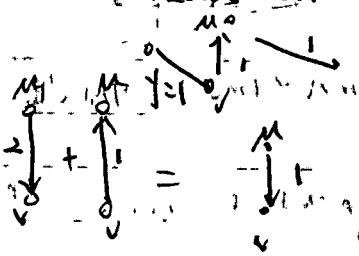


VALUE 3 (BEST POSSIBLE)

THE "INCORRECT" EDGES FROM A PATH OF SORTS.



IDEA: FLOWS CAN CANCEL IN PARTICULAR



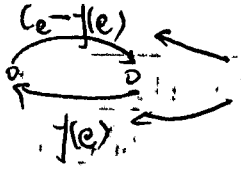
USING THE IMAGINARY PATH P' (W/ 1 UNIT OF FLOW) PRODUCES AN OPTIMAL (MAX-VALUE) FLOW

TO SYSTEMATICALLY FIND IMAGINARY PATHS LIKE P', WE DEFINE THE RESIDUAL GRAPH $G_f = (V, E_f, c_f)$ OF THE FLOW.

SAME VERTICES, SOURCE & SINK

EDGE $e: x \to y$, CARRIES FLOW $f(e) \le c_e$,

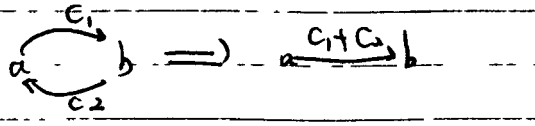
G_f WILL HAVE 2 EDGES



CAPACITIES

THIS could create multiple (double) edges BUT ALLOW this.

We will consolidate these

SO 

FORD-FULKERSON PROC.

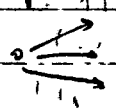
$f = 0$, $G_f = G$

WHILE $\exists$ AN S.T PATH P
OF POS CAPACITY $\Delta > 0$
SEND Δ UNITS of FLOW ALONG P (CHANGES f)
UPDATE G_f

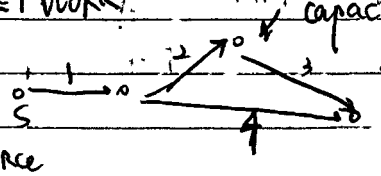
THIS WILL STOP IF ALL CAPACITIES ARE INTEGERS.

INVIRIVANT: ALL CAPACITIES OF G_f's $\in \mathbb{Z} \geq 0$
$\therefore$ VALUE INCREASES BY $\Delta \geq 1$

BUT ANY FLOW SATISFIES
VALUE $(f) \leq \sum_{e \text{ out of } S} f(e) \leq \sum_{e \text{ out of } S} c_e$



WK 2 Fri
TODAY: MAX-FLOW, MIN-CUT THM
(PATH AUG. ALG FOR MAX. FLOW)
NEXT MON: BIPARTITE MATCHING KT 7.5

FLOW NETWORK.
E.G. 

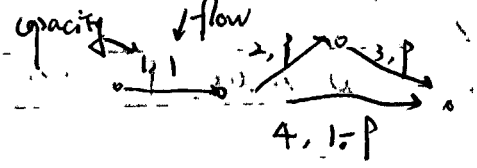
GOAL: FIND A FLOW OF MAX. VALUE (NET FLOW OUT OF S).

FLOWS SATISFY $0 \leq f(e) \leq$ capacity of e c_e.

∀ $v \neq s, t$

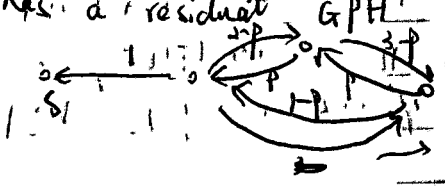
FLOW = FLOW
INTO v OUT OF v.

△ MAX-FLOW NEED NOT BE UNIQUE



ANY FLOW f has a residual GPH G_f

E.G.



EDGES INDICATE WHERE EDGE FLOW COULD BE CHANGED.

FORD + FULKERSON PROC. — AUGMENTING PATH
(SEEKS A MAX FLOW)

$f=0$, $G_f = G$ while G_f has a s.t. path P of capacity $\Delta > 0$

$f = f + (\Delta$ units ALONG $P)$

REVISE G_f.

EACH STEP INCREASES FLOW VALUE

IF IT STOPS, NO AUG PATH EXISTS!

THM: ALL $C_e \in \mathbb{Z}$ F-F STOPS

COR: ALSO IF $C_e \in \mathbb{Q}$

CUTS: A CUT IS A PARTITION OF V.

$V = CUT(S, T = \overline{S})$ with $s \in S$, $t \in T$

E.G. $S = \{s\}$ ITS CAPACITY IS

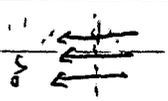
$T = S - s$

$\sum_{\substack{e: x \to y \\ x \in S \\ y \notin t}} c_e$ TOTAL "FORWARD" CAPACITY OUT OF S

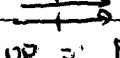
THM:

FOR ANY FLOW f, ANY CUT C, value(f) $\leq$ capacity(C)

A PHYSICAL ARGUMENT



THM (WEAK DUALITY MAX $\leq$ MIN)



VALUE = FLOW OUT OF S

= FLOW INTO S $\leq$ CAPACITY

MAX FLOW (MIN CUT!! THM (STRONG DUALITY))

VALUE = CAPACITY OF A MIN. CUT

OF MAX FLOW

PF: TFAE 1) f is MAXIMUM

2) $\exists$ AUG PATH

3) value(f) = capacity(C) FOR some cut C

1) $\Rightarrow$ 2). SAME AS

2) $\Rightarrow$ 1).

IF $\exists$ AUG PATH

WE CAN INCR. FLOW

2) $\Rightarrow$ 3) NO AUG $s \to t$

A SEARCH FOR IT IN G_f ($\exists$ cap edges) will BE Blocked.

LET S = vertices reachable from s

$T = V - S$ (Note $t \in T$)

IF A FWD EDGE $\Rightarrow$ (FROM s to t) has capacity, G_f

I WOULD MAKE S BIGGER, SUCH EDGES DON'T EXIST IN

CAPACITY OF (S,T) = FLOW VALUE.

3) ⇒ 1) USE WEAK DUALITY ☒

COR: IF F-F PROC TERMINATES
THE FLOW IS MAXIMAL

WK13 MON
TODAY: NTWK FLOW FORD/FULKERSON RUN TIME KT 13
BIPARTITE MATCHING KT 7.5

MAX FLOW / MIN CUT THM.

IN A FLOW NTWK:
WEAK DUALITY: VALUE OF ANY FLOW ≤ CAPACITY OF ANY CUT
EQUALITY HOLDS AT OPTIMALITY

STRONG DUALITY: F/F PROC REPEATEDLY FIND PATH OF POS CAPACITY
PUSH AS MUCH FLOW AS POSSIBLE THRU PATH
UPDATE RESIDUAL GRAPH
(INDICATES EDGES THAT COULD CARRY FLOW
†THEIR CAP'S)

MFMC THM:
INTEGRAL CAPACITIES
ALG will stop
AFTER ≤ C IMPROVEMENTS
C = ANY BOUND ON MIN CUT CAPACITY

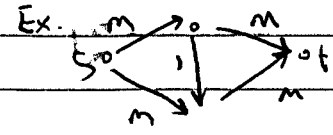
EXAMPLE (w/ IRRATIONAL CAPACITIES)
show that F-F CAN RUN FOREVER

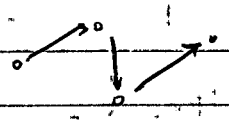
RUNTIME: (CAP's ∈ ℤ) is O(m)·C = O(mC)
steps to FIND A PATH
UPDATE RES GPH G_f

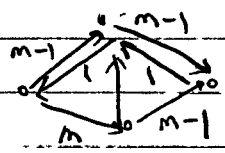
A Pseudo-polynomial

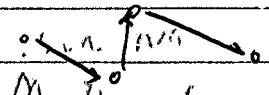
TIME BOUND

EXAMPLES. KNOWN OF NON-POLY RUNTIME (CAP's $\in \mathbb{Z}$)

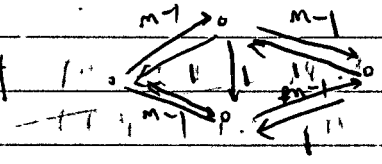
Ex. 

1 UNIT ALONG 

G_1 

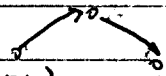
~~1 UNIT~~ 1 ALONG 

REPEAT (SAME PATHS)

NEXT: G_1 

USE AUG. STEPS

WE STOP IN 2 ROUNDS

IF 1st PATH IS 

EDMONDS + KARP (1972) SHOWED SHORTEST-PATH AUGUMENTATION (LEAST # EDGES) will stop after $O(mn)$ Rounds - (IF CAP's $\in \mathbb{Z}$)

TOTAL RUNTIME FOR E&K is POLYNOMIAL $O(mn^2)$

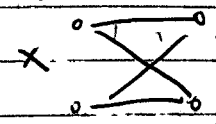
$= (V, E)$

BIPARTITE MATCHING. (G UNDIRECTED GPH)

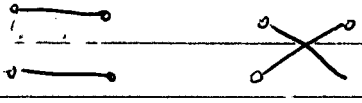
set $M \subseteq E$ SHARING NO ENDPT (EACH MATCHED VERTEX HAS A UNIQUE PARTNER)

$V = X \cup Y$ EVERY EDGE IS $\{x, y\}$ $x \in X$, $y \in Y$

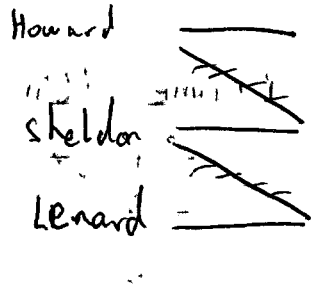
Ex.

$K_{2,2}$ 

HAS 2 PERFECT MATCHING



Ex2



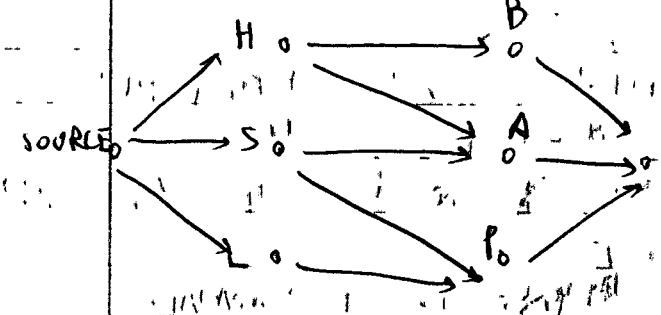
MAXIMAL AS A SET

$\nexists$ $M' \geq M$

↑ matching

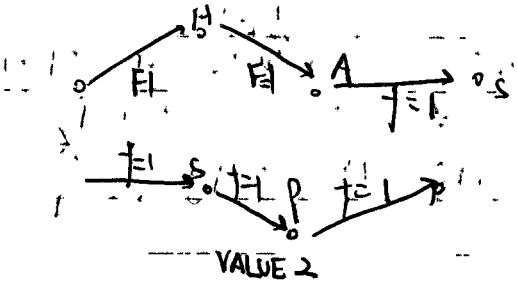
WE COULD TRY A "GREEDY" STRATEGY FOR FINDING A LARGEST MATCHING (CAN GET STUCK)

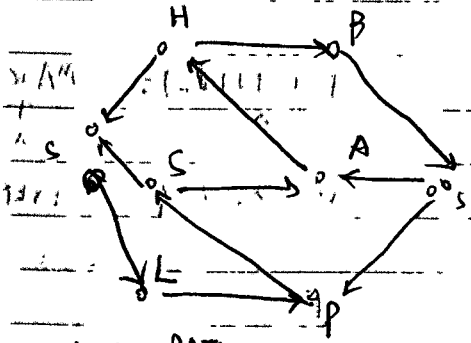
TRY NTWK FLOW



ALL CAPACITIES ARE 1

CAN USE 2 PATHS





AUG PATH

$\rightarrow L \rightarrow P \rightarrow S \rightarrow H \rightarrow B \rightarrow S$

7EGDES

Original RUNTIME:

GPH → G = (V, E) w/

$|V| = 2n$, $|E| = m$, $|X| = |Y| = n$

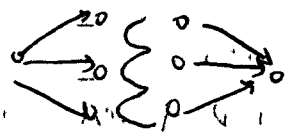
Derived F FLOW NTWK HAS $2n+2$ VERTICES
$m+2n$ EDGES

F-F PROC Does path search

cost per search is $O(m+2n+2n+2)$

$= O(m+n)$

# of FLOW IMPROVEMENTS IS $\leq n$



TOTAL IS $O(n+m)n) = O(mn)$

IF G was connected $= O(n^3)$

BASEBALL ELIMATION

SPORTS LEAGUE w/n TEAMS.

GAMES HAVE A WINNER + A LOSER (NO TIES)

CHAMPION IS THE TEAM W/ MOST WINS AT END OF SEASON

Q is my team ELIMINATED AS A CHAMPION.

ONE IDEA:

IF LEAGE LEADER WINS ~~already~~ faced already

my team wins

GAMES TO PLAY

my team can't win championship

Won't cover all situations.

	WINS SO FAR	YET TO PLAY	
NYY	92	2	BAL, TOR
BAL	91	3	NYY, TOR, BOS
TOR	91	3	NYY, BAL, BOS
BOS	90	2	BAL, TOR

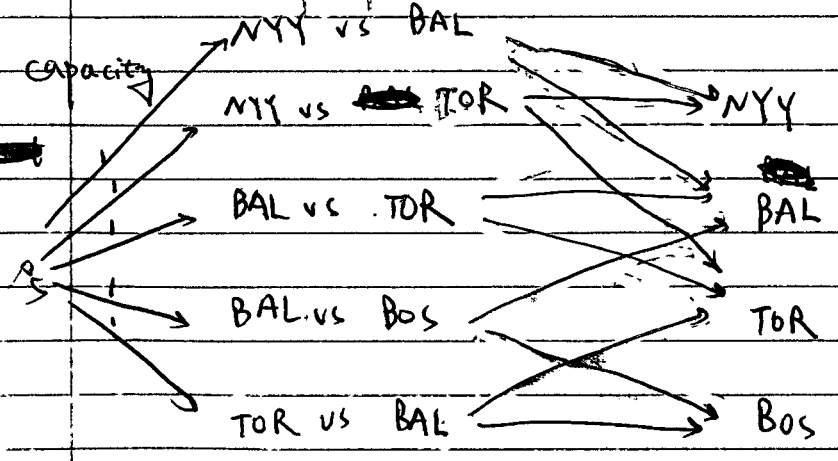
NYY could end w/ 92 (2 losses)
BOS could end w/ 92 (2 wins)

BOS is eliminated

PF: NYY loses to BAL, TOR
BAL, TOR, GET TO 92 WINS FROM THIS
WINNER OF BAL vs TOR GETS ≥ 93 wins

Problem: How to determine such conclusions systematically?

REDUCE TO A MAX-FLOW PROBLEM



what cap's

- Do tem → sink EDGES GET z = my team
  m = max possible wins FOR z (= 92 BOS)
  z =

KEY : IF z wins.
it can do so by winning all its REM. GAMES

$$\forall x \left[\underbrace{\#\text{ already won By } x}_{W_x \text{ (DATA)}} \right] + \left[\begin{matrix} x\text{'s} \\ \text{FUTURE} \\ \text{wins} \end{matrix} \right] \leq m$$

FLOW ON $x \to t$

$$\left[\begin{matrix} x\text{'s} \\ \text{future} \\ \text{wins} \end{matrix} \right] \leq m - w_x$$

No flow of value 5 subject to constraints
B.O.S. can't win

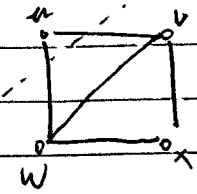
Wk 4 Fri

$G = (V, E)$ UNDIR. G.P.H

$n = |V|$, $m = |E|$

A CUT is A partition of V into two nonempty sets.

$V = A \cup B$

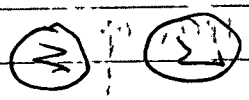


E.G: $\{u, v\}, \{w, x\}$ size 3 — $\{u\}, \{v, w, x\}$ (smallest)

G has $2^n - 2$ cuts size of (A, B) is the # of crossing edges

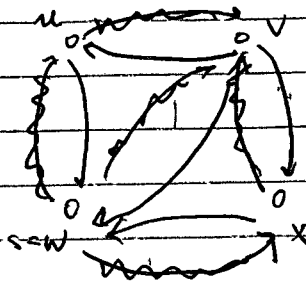
The size of a min cut measures G's "connectness"

NOTE: G is connected $\Leftrightarrow$ min cut size ≥ 1



we can find A global min-cut using $n-1$ MAX FLOW comp'utations $|V|$

ALG: ① EACH EDGE BECOMES 2 DIRECTED EDGES (UNIT CAPACITY)



① choose any $s \in V$

② For each $t \in V - \{s\}$
FIND A MAX s-t FLOW
GET A MIN s-t CUT
USE BEST CUT FOUND

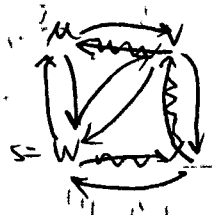
→ value 3 : cut is $\{u\}$ (3)

CAP 3 $\{w, v, x\}$

sink t cut capacity

v 3

w 2



KARGER (1992)

A RANDOMIZED MIN CUT ALG WORST w/ MULTIGRAPHS

(MULTIPLE EDGES, NO SELF LOOPS)

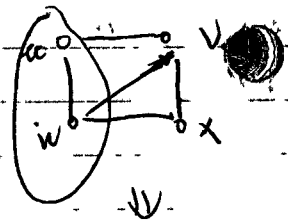
COLLAPSING STEP

PICK A RANDOM EDGE FROM G

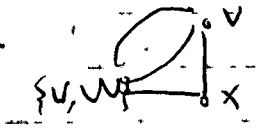
IDENTIFY ITS END PTS

REMOVE SELF LOOPS

collapse



EACH reduces |V| BY 1 REPEAT UNTIL |V|=2



$\{u,v,w\} = A$

$\{x\} = B$

OUTPUT THE CUT FOUND IS [illegible]

CANDIDATE MIN-CUT

RUN THIS N TIMES,

OUTPUT THIS N TIMES, output the best cut

(MIGHT NOT BE MINIMAL)

HAS PROB $\geq \frac{1}{\binom{n}{2}} \sim \frac{2}{n^2}$ of finding A min-cut

IF $N = \binom{n}{2}$, PR of not find a min-cut is

$\leq \left(1 - \frac{1}{\binom{n}{2}}\right)^{\binom{n}{2}} \leq e^{-1} = 0.367$

PF Let C be a min-cut of size k

each $v \in V$ has $\geq k$ nbrs

$$|E| = \frac{\sum \deg(v)}{2} \geq \frac{kn}{2}$$

$$\Pr\left[\text{some edge of } C \text{ is collapsed in } A\text{th step}\right] \leq \sum_{e \in C} \Pr[e \text{ is chosen}]$$

$$\leq \frac{k}{|E|} \leq \frac{k}{\frac{kn}{2}} = \frac{2}{n}$$

Given that C survives $\Pr\left[\text{survives 2ND step}\right] \geq \left(1 - \frac{2}{n-1}\right)$

So $\Pr\left[C \text{ survives to the end}\right] \geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right)\cdots$ ETC

$$\left(1 - \frac{2}{4}\right)\left(1 - \frac{2}{3}\right)$$

$$= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{2}{4} \cdot \frac{1}{3}$$

$$= \frac{2}{n(n-1)} = \binom{n}{2}^{-1}$$

WK14 WED

TODAY: ADVERSARY LOWER BOUNDS

BASE PP. 124 - 133 (ON OUR website)

SO FAR.

Lower Bounds ON ALG complexity by counting outcomes

THM: IF A BINARY DECISION TREE has L leaves, its height is $\geq \log_2 L$

Good: IF A PROB has many possible results

e.g. sorting n distinct keys

NEED $\geq \log_2(n!) = n\log_2 n - O(n)$ Comp's in worst case

It may not be informative.

ex. Finding max of $x_1, x_2, \ldots x_n$ (Distinct)

n outcomes

$\therefore \geq \log_2 n$ comp's

A better Bound

each $x_i \neq$ max must "lose" to another x_j ($x_i < x_j$) since any undefeated key could be largest

$\therefore \geq n-1$ comp's needed.

ALG to do this

$M = x_1$

$n-1$ comp's For $i = 2$ To n

[if $(x_i > M)$ then $M = x_i$

Ex 2: Finding max + min of n keys — cost $n-1$

obvious ALG's: Find max of n keys

Remove it — $n-2$

Find min of $n-1$ remaining keys $2n-3$

Is $2n-3$ optimal?

imagine a conversation

An ALG: A

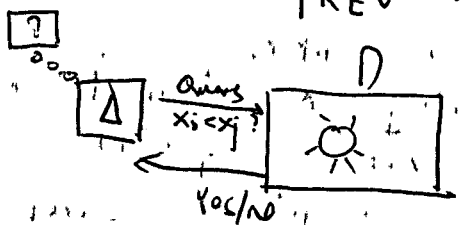
A "DEMON" D (the adversary)

A sends queries (ie "$x_i < x_j$")

TO D

D ANSWERS, must be consistently

PREV REPLIES



A wants to learn max w/ as FEW QUERIES As possible

D wants to maximize the # of queries

Group keys into 4 states } what is D's BEST STRATEGY

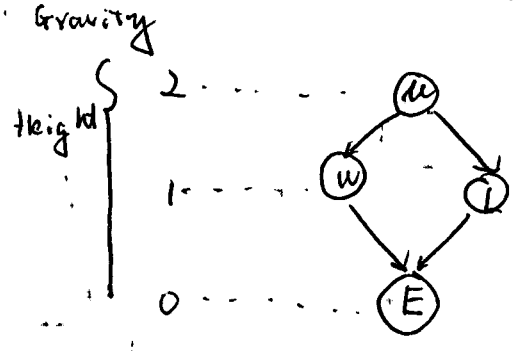
U - untested

W - won, never lost

L - lost, never won

E - ELIMINATED WON AND LOST

Introduce a physical model



keys: have unit mass

Potential energy = $\sum$ (mass) × (height) PE TO START: $2n$

(ALL keys u)

At end 1W, 1L, 2E

PE = 2

D's strategy:

IF A Query has 2 consistant Replies, choose one that minimize ΔPE

Ex: u vs L

if u wins $\Delta PE = 1$

if L wins $\Delta PE = 2$

Demon will pick this

D can force $\Delta PE \leq 1$ Except for u vs u Queries (PF exercise)

THERE ARE $\leq \frac{n}{2}$, u vs u queries

other queries must DROP PE $\geq n-2\frac{n}{2}$ to 2

must BE $\geq n-2$ other queries

WE GET A MUCH BETTER Lower Bound ANY ALG to FIND max, min must use $\geq \frac{3n}{2} - 2$ key comp's

n Even

A should Do all possible u vs. u queries.

($\frac{n}{2}$ OF THEM)

FINDS MAX of $\frac{n}{2}$ winners $+\frac{n}{2} - 1$

" MIN of $\frac{n}{2}$ Losers $+\frac{n}{2} - 1$

$\frac{3n}{2} - 2$

NAIVE (MAX THEN MIN)

Uses $n-1 + n-2 = 2n-3$

WKM Fri

577 So far

improvements to ALG's For problems we can solve already

Ex. sorting ^ keys

NAIVE ALG's $O(n^2)$ comp's

D+C $O(n \log n)$

Best possible.

As Better-better, ALG's WERE FOUND, some "HOLDOUTS" REMAINED, (& W/ (APPARENTLY) NO BETTER ALG THAN EXHAUSTIVE SEARCH.

Examples:

1) Boolean: satisfiability

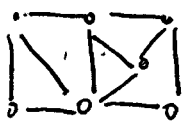
E.G: CAN WE MAKE

$(x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge (x \vee z \vee w) \wedge \ldots$

EQUAL 1 ?

2) HAMILTONIAN CIRCLE

EG:



is there A TOUR USING EDGES, VISITING EACH NODE ONCE

ALL SUCH "HOLDOUTS" SHARE A PROPERTY.

EVERY POSITIVE EXAMPLE HAS A SHORT, EASY TO CHECK CERTIFICATE

EACH NEG EXAMPLE HAS NO SUCH CERT.

PROBLEM	CERT	CHECKER
SATISFIABILITY	0-1 ASST'S TO VALUES	Bottom-up EVAL OF FMLA
HAM CIRCLE	LIST OF VERTICES $v_0, v_1, \ldots v_{n-1}, v_n = v_0$	

S.T. $\{v_i, v_{i+1}\} \in E$ ✓ VERIFY THESE

AND $V = \{v_1, \ldots v_{n-1}\}$

THE CLASS OF DECISION (YES-NO) PROBLEM'S w/ such certificates IS NOW CALLED NP

NP prob's could be solved if we could solve the "certificate search" problem.

WHAT COUNTS AS A COMP'N?

1930's CHURCH, GODEL, KLEENE, ...

ANY ALG THAT TERMINATES (w/ A correct answer)

1960's COBHAM, EDMONDS

ANY ALG RUNNING IN TIME $\leq$ POLYNOMIAL (INPUT LENGTH) } WORST-CASE

E.G.

1) SORTING, $x_1 \ldots x_n$

$O(n \log n)$ comp's

FOR large n, $n \log n \leq n^2$

2) BIPARTITE MATCHING

FORD-~~FULKERSON~~ Fulkerson ALG., $O(n^3)$, $n = |V|$

WHAT ABOUT TRIAL DIVN PRIME TESTING?

1) FOR $d = 2$ TO $\sqrt{n}$ ✓ FIBONACCI!

IF $d \mid n$ SAY "composite" AND STOP SAY "PRIME"

d is "evenly" divided.

WORST CASE ^ PRIME
ALG DOES $\sqrt{n} = 2^{\frac{1}{2}\log n}$ DIVN STEPS

BINARY LENGTH $n = \lfloor \log_2 n \rfloor + 1$

NOT POLY IN # OF BITS IN THE INPUT

MOST

FOR n PROBLEMS USING ONLY "SMALL" #s
NAIVE of 'N COUNTS
ARE SUFFICIENT

Ex: BIPARTITE MATCHING
ADJ MATRIX FOR G has n^2 BITS
NEIGHBORLIST " HAS $O(m \log n + n)$ BITS

THEP CWP

PF. THE CHECKER CAN IGNORE THE CERT AND JUST DECIDE IF IT HAS A YES OR NO INSTANCE, BY RUNNING THE POLY TIME ALG.

WK15 MON K&T
P AND NP 8.3
REDUCTIONS 8.1
NP-complete probs 8.4

P : DECISION PROBLEM SOLVABLE IN POLY TIME
E.g $O(n^2)$, $O(n^3)$

$\cap$

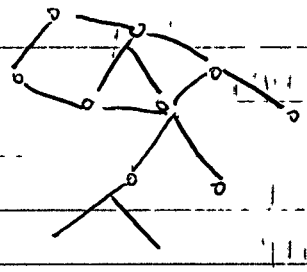
NP DECISION PROBLEM w/ short, easily checkable certificates

Ex1 Composite Numbers ($n=1$, in Binary $\approx \log n$ Bits)

certificate: Any m, $1<m<n$ s.t. $m \mid n$ ($n \bmod m \neq 0$)

Ex 2: BIPARITE GRAPHS

CERT: ODD CYCLE

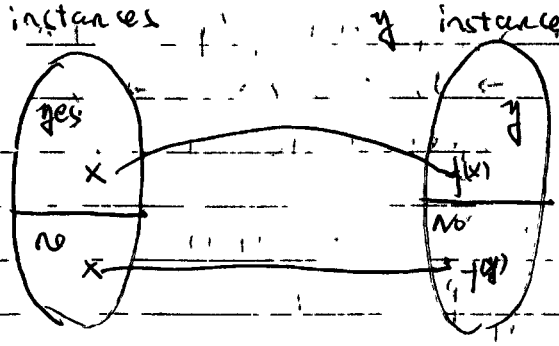


REDUCTIONS:

DEF IF X, Y ARE DECISION PROBLEMS, A REDUCTION FROM X TO Y is MAPPING f s.t.

IF x is A yes ins of X, $f(x)$ is A yes ins of Y

IF y is NO Y, $\forall y$ NO Y



Ex. X = QUAD. EQNS $ax^2+bx+c=0$ $a,b,c \in Q$, $a \neq 0$

Question: DOES EQN have a real solution?

Roots: $\frac{-b \pm \sqrt{b^2-4ac}}{2}$

LET $D = b^2-4ac$ YES, IFF $D \geq 0$

So y = rational # ≥ 0

Question: "Is $D \geq 0$"?

$f: (a, b, c) \longrightarrow b^2 - 4ac$

DEFN: IF x, y ARE DECISION PROBS

$x \leq_p y$ MEANS

THERE'S A RED FROM x to y

Computable IN POLY TIME

So QUAD EQNS w/ real solution $\leq_p$ NON-NEGATIVE NUMBERS

THM: IF $x \leq_p y$ AND $y \in P$ THEN $x \in P$

PF: RED ↑ ALG A ↑ θ

THE ALG FOR x is LET $y = f(x)$

ANSWER WHATEVER A SAYS ON y

COR. IF $x \leq_p y$ AND $x \notin P$ THEN $y \notin P$

PF: $\varphi \Rightarrow \theta$ SAME AS $\bar{\theta} \Rightarrow \bar{\varphi}$

(JUST LOGIC)

DEFN A DECISION PROBLEM Z is NP-COMPLETE IF:

1) $Z \in NP$

2) $x \leq_p Z$, FOR ALL $x \in NP$

INFORMAL IDEA: THERE ARE THE "HARDEST" PROBLEMS WITHIN NP.

1971 COOK (US)
LEVIN (USSR)

SHOWED Boolean SATISFIABILITY is NP-C,

1972: KARP
ABOUT 20(!) WELL KNOWN DISCRETE MATH PROBS ARE NP-C

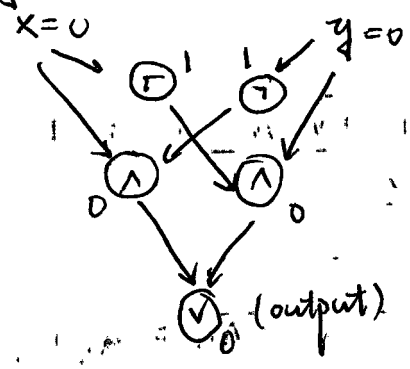
WK 15 WED
TODAY: VARIATIONS ON SATISFIABILITY
COOK - LEVIN THM (SAT IS NP-C).

DECISION PROBLEMS
P - HAS A POLY-TIME ALG
NP - HAS SHORT, EASY TO CHECK CERTIFICATES
$X \leq_p Y$ - X REDUCES TO Y IN POLY TIME VIA SOME f
f send {yes / no} INSTANCES TO {yes / no} INSTANCES of Y of X

X is NP-complete $X \in NP$
FOR EVERY $Z \in P$, $Z \leq_p X$

A Bodean CIRCUIT IS A DAG SUCH THAT:
1) EACH SOURCE (IN-DEG 0) IS LABELED BY A const (0, 1) OR value ($x=1, 2, \cdots$)
2) NODES w/ IN-DEG 1 LABELED NOT ($\neg$)
3) NODES w/ ID-DEG 2 BY AND (∧) OR OR (∨)
4) THERE is one sink (out-DEG, 0)
N REPESENTING THE OUTPUT w/

E.G $x\bar{y} \vee \bar{x}y$ (XOR)



EACH SUCH CKT REPRESENT A Boolean FN

we can evaluate it (FOR AN ASSIGNMENT OF ITS VBLES) IN POLY TIME

CKT SAT $\in$ NP

INSTANCE: A Boolean CIRCUIT

QUESTION: Does some ASSIGNMENT TO THE VBLES MAKE THE OUTPUT 1?

is satisfiable PF: use $x=0, y=1$

COOK-LEVIN THM:

CKT-SAT IS NP-C

PF (SKETCH): LET $Z \in NP$

SUFFICIENT TO SHOW $Z \leq$ CKT SAT

LET A (JUST AN ALG) BE A CERTIFICATE CHECKER FOR Z

SO (FOR SOME POLY $p(\cdot)$, E.G. $\wedge^4 + 10$) z IS A YES INSTANCE OF Z

IFF $\exists y,\ |y| \le p(|x|)\ [A(z,y) = 1]$

A: ALLEGED CERTIFICATE; z: INSTANCE

NOTE: z, y ARE STRINGS OF 1's & 0's, is REPRESENTING INSTANCES + CERTIFICATES.

CONSIDER AN (ENCODED) z - INSTANCE, z

THERE is A Boolean CKT C_z THAT REPRESENTS

$y_1 \to A(z, y_1)$

A Boolean FN OF $p(|x|)$ values

THE MAPPING $z \longmapsto C_z$

(INSTANCE) (CERTIFICATE CHECKER FOR z)

IS POLY-TIME COMPUTABLE

(DETAILS NEEDED)

z IS A "yes" INST IFF C_z is satisfiable.

DONE

(SEE K+T p 464 FOR A NICE EXAMPLE)

GOAL: SHOW MORE PROB's ARE NP-C,

STP PF: To show z is NP-C

WE SHOW ① $z \in NP$ (USUALLY) EASY

FOR A KNOWN NP PROB w,

$w \le_p z$

OBSERVE THAT

IF $x \le_p w$ FOR ALL $x \in NP$,

WE HAVE

$x \le_p w \le_p z$

COMPOSE THE REDUCING FNS TO SHOW $X \leq_P Z$

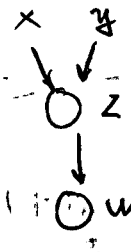
EX: Z = SATISFIABILITY OF Boolean FORMULAS

NEED TO SHOW: CKT-SAT $\leq$ FORMULA-SAT
(F-SAT)

THE REDUCTION

TAKE A CKT

INTRODUCE A NEW VBLE FOR EACH NODE W/ INPUTS (GATE)



THE FMLA IS

[FMLA FOR $z = x \vee y$]

[FMLA FOR $w = \neg z$]

$\wedge \psi$

WK

WK 15 Fri

More NP-C PROBLEMS

FORMULA SAT

INDEPENDENT SET

COOK-LEVIN THM (LAST TIME)

CKT-SAT IS NP COMPLETE
SATISFIABILITY OF Boolean circuits

Boolean Formulas
expressions using constants 0, 1
variables x, y, z...
operations ∧ (AND)
∨ (OR)
¬ (NOT)

DEFN A FMLA IS IN CONJUNCTIVE NORMAL FORM (CNF)
IF IT'S A OR OF CLAUSES
(AND OF VBLES OR THEIR NEGATIONS)
x, $\bar{x}$, ... LITERALS

CNF $(x \vee y) \wedge (\bar{x} \vee \bar{z} \vee w) \wedge (\bar{x} \vee \bar{y})$

$\bar{x}$ MEANS $\neg x$

$\neg(x \vee y)$ IS NOT CNF

Pone 240 352 IN FACT EVERY Boolean FN CAN BE REPRESENTED BY SOME CNF FORMULA

E.G
$x \neq y$, HAS THE CNF $(x \vee y) \wedge (\bar{x} \vee \bar{y})$

THE SATISFIABILITY OF Boolean formulas ($\overbrace{\text{F-SAT}}$) IS NP-COMPLETE

PF: 1) F-SAT $\in$ NP
(CERTIFICATES = SATISFYING ASS'TS)

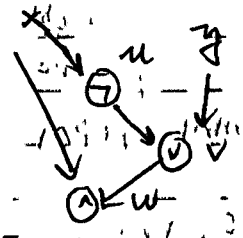
2) WE know that
FOR ANY NP PROBLEM X, $X \leq_p$ CKT-SAT } Cook-LEVIN

WE'll Show that CKT SAT $\leq_p$ F-SAT
So $\forall x \in NP$, $x \leq_p$ F-SAT. (COMPOSE THE REDUCTIONS)

THE REDUCTION

CRT. C)

E.G.



INTRODUCE ~~INTRODUCE~~ A NEW VBLE FOR EACH "OPERATION" NODE

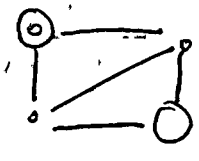
THE FMLA FOR C ASSERTS
THAT ① ALL OPN'S ARE DONE correctly AND ②
THE OUTPUT is T

E.G.

[CNF FOR $x \neq y$] $(x \vee w) \wedge (\bar{x} \vee \bar{w})$ } ①
$\wedge$ [CNF FOR $v = u$ OR y]
$\wedge$ [CNF FOR $w = x$ AND v]
$\wedge w$

THIS IS SAT IFF THE CKT IC WAS T

INDEPENDENT SET (IS)
$G = (V, E)$ UND GPH
A SET OF VERTICES IS INDEPENDENT
IF NO EDGE HAS BOTH ENDS IN S



THM: THE DECISION PROBLEM "DOES G HAVE AN IND SET W/ ≥ K VERTICES?" IS NP-C

A REDUCTION FROM CNF-SAT TO IS }

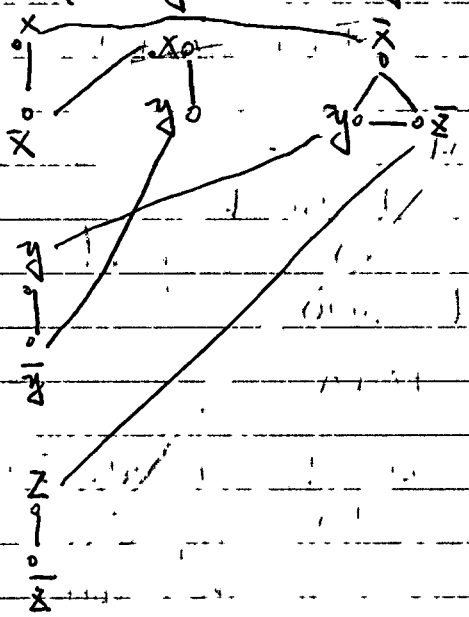
COR CNF-SAT IS NP-C

(THE CNF'S HAVE ≤3 LITERALS IN ANY CLAUSE)

THE KEY STEP IN PROVING IS IS NP-C

THE REDUCTION:

$(\bar{x} \vee \bar{y}) \wedge (\bar{x} \vee y \vee \bar{z}) \longrightarrow$



A LABELED COMPLETE GPH

FOR EACH CLAUSE

A LABELED EDGE FOR EACH VBLE.

TARGET $k = [\#\text{ OF CLAUSES}] + [\#\text{ OF VBLES}]$

S IS AN I.S. W/ k VERTICES IFF IT USES 1 VERTEX FROM EACH "GADGET" AND COVERS NO CROSSING EDGE

IFF THE VBLE-GADGET VERTICES IN THE SET

GIVE A SATISFYING ASST.

WK16 MON

X is NP-HARD MEANS

IF X HAS A P-TIME ALG

THEN P = NP

3-SAT: satisfiability with Exactly 3 literals in each clause

↑

NP-C

WE SHOWED

≤3-SAT (≤3 Literals in EACH CLAUSE)

is NP-C

THM: ≤3-SAT ≤p 3-SAT

ANY CLAUSE w/ 1 Literal,

E.G. $\bar{x} \wedge (\bar{x} \vee z \vee w) \wedge$ is forced.

$(\bar{x} \vee z \vee u)$

I can remove it from FMLA

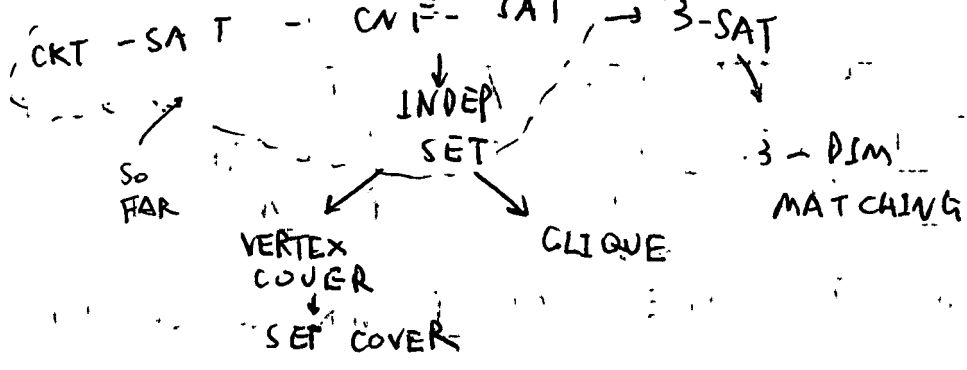
IF ANY CLAUSES w/2 Literals REMAIN, E.G.

$(x \vee \bar{y}) \longmapsto (x \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee \bar{z})$

z = NEW VBLE

I CAN REPLACE EACH

BY 2 CLAUSES w/ A NEW VBLE



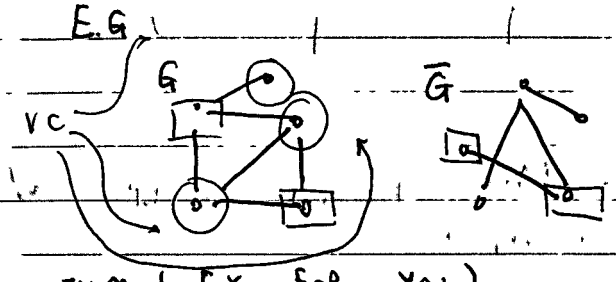
$X \to Y$ MEANS $X \leq_p Y$

3 GRAPH PROBLEMS

$G = (V, E)$, UNDIR GPH

$\overline{G} = (V, \overline{E})$ COMPLEMENT HAS $\{v, w\} \in \overline{G}$ IFF $\{v, w\} \notin E$

E.G.



THM (EX: FOR YOU)

$S \subseteq V$ is A VERTEX COVER of G : (EVERY EDGE TOUCHES S)

IFF $\overline{S} \subseteq V$ is INDEPENDENT IN G (NO EDGE HAS BOTH ENDS IN $\overline{S}$)

IFF $\overline{S} \subseteq V$ is A CLIQUE in $\overline{G}$ (ANY 2 VERTICES IN $\overline{S}$ ARE LINKED BY AN EDGE)

THM: THE FOLLOWING PROB'S ARE NP-C

① Does A GIVEN GPH G HAVE A VERTEX COVER OF SIZE $\leq k$?

② Does G have AN INDEP. SET with $\geq l (= n-k)$ VERTICES?

③ Does G HAVE A CLIQUE w/ $\geq l$ vertices?

SET COVER:

INSTANCE: $M = \{1, \ldots, n\}$

$G = \{S_1, S_2, \ldots S_m\}$

EACH $S_i \subseteq M$

Question: ARE THERE $\leq k$ S_i's

WHOSE UNION is M?

E.G. $k=2$

	1	2	3	4	5	
S_1:	0	1	0	1	0	} No
S_2:	1	1	0	0	0	
S_3:	1	1	1	0	0	
S_4:	0	0	0	1	0	
S_5:	1	0	1	0	1	} Yes

$\{2,4\}$

VERTEX COVER
IS THE SPECIAL CASE OF SET COVER w/ $|S_i| = 2$
∴ SET COVER is NP-C

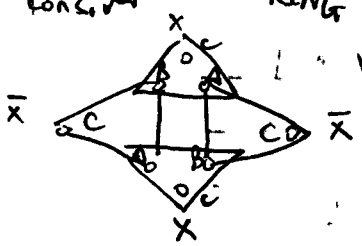
≡ Exactly 2 1's per row

3DM.
INSTANCE = 3 DISJOINT SETS A, B, C s.t. $|A| = |B| = |C| = n$
+ triples (a, b, c) w/ $a \in A, b \in B, c \in C$

Question: ARE there n triples using ALL OF A, B, C?

P.F. SHOW 3 SAT $\leq_p$ 3DM (KET P.481)
LET ϕ BE A 3-SET FMLA w/
n VBLES
k clauses
($\Rightarrow 3k$ literals APPEAR)

consider A RING



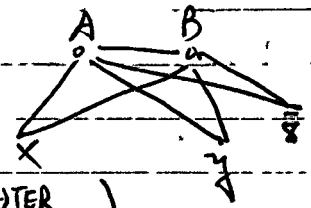
$k=2$

w/ $2k$ INNER VERTICES
& outer vertices
it's BISTABLE
(ONLY 2 WAYS TO MATCH INNER VERTICES)

MAKE SUCH A RING FOR EACH VBLE X

ITS
LABEL OUTER VERTICES $x, \bar{x}, x, \bar{x} \ldots$ (CYCLIC ORDER)

EACH CLAUSE SAY $x \vee y \vee \bar{z}$ HAS



WE HAVE THIS CONNECTED $\leq 3k$ OUTER A VERTICES IF WE PICK A "WITNESS" (OUTER) TO SATISFY EACH CLAUSE. THIS VERTICES MATCHES k OUTER VERTICES FOR X

WE have $kn-k$ unmatched outer vertices

ADD $k(n-1)$ "CLEAN-UP" COMPONENTS.

WK 16 WED

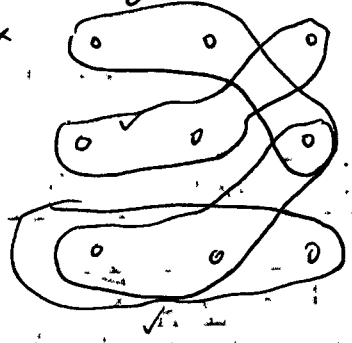
TODAY: INTEGER PROGRAMMING (IP)
VS LINEAR PROGRAMMING (LP)
NP HARD IP'S

DEF: X is NP - HARD
IF X (solvable in POLY-TIME
IMPLIES $P = NP$
IF X is A DECISION PROB, NP-HARDNESS FOLLOWS
FROM $\forall Z \in NP,\ Z \leq_p X$

3DM ("LAST TIME)
Given $A = \{\alpha_1, \ldots, \alpha_n\}$
$B = \{\beta_1, \ldots, \beta_n\}$
$C = \{\gamma_1 \ldots, \gamma_n\}$
t in Triples $(\alpha_i, \beta_j, \gamma_k)$
ARE THERE n of these
THAT [1]† EVERY ELT OF A
B
C

A B C

Ex



Yes

subset sum

INSTANCE $a_1 \ldots a_n$ } Pos INTEGERS

TARGET → b

Question: CAN WE ADD SOME of THE a_i's AND Get b?

KT 492-493

THM: SUBSET SUM (SS) is NP-C

(Because 3DM $\leq_p$ SS)

PF LET $d = n+1$ (BASE)

EACH TRIPLE PRODUCES ONE a_i E.G

BASE d →

α_1	α_2	α_3	β_1	β_2	β_3	γ_1	γ_2	γ_3	
✓ 1	0	0	1	0	0	0	0	1	a_1
✓ 0	1	0	0	1	0	1	0	0	a_2
0	0	1	0	0	1	0	1	0	a_3
✓ 0	0	1	0	0	1	0	0	1	a_4
1	1	1	1	1	1	1	1	1	$b \left(= \sum_{j=0}^{3n-1} d^j\right)$

$\exists S$ s.t. $\sum_{i \in S} a_i = b$

IFF THE 3DM

INSTANCE is A "yes" ☒

SS is "HARD"

Because we can use wide numbers.

To enforce many constraints simultaneously.

We can solve SS using $O(n \cdot b)$ ARITH OPNS

$b = 2^{\log b}$ is NOT POLY in input length

INTEGER PROGRAMMING:

OPTIMIZATION OF A LINEAR FN OVER INTEGER VBLES subject to

LINEAR PROGRAMMING

E.G. ~~IP~~ SS (FEASIBILITY)

CAN WE SOLVE $\sum_{i=1}^{n} x_i \cdot a_i = b$

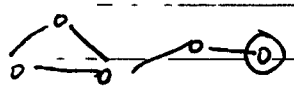
s.t. $0 \le x_i \le 1$, $i = 1, \ldots, n$

AND $x_i \in \mathbb{Z}$

$\Rightarrow$ IP (OPT VERSION) is NP-HARD

ALT. PF CONSIDER VERTEX COVER

E.G.



MINIMIZING VERSION:

FIND A SMALLEST $S \subseteq V$ that TOUCHES all edges

AS IP:

VBLES $x_1 \ldots x_n$ $V = \{1, \ldots, n\}$

$\min x_1 + \cdots + x_n$

s.t. $x_i + x_j \ge 1$, $\forall \{i, j\} \in E$

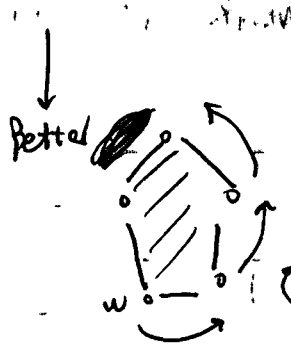
$0 \le x_i \le 1$

$x_i \in \mathbb{Z}$

L.P OPTIMIZE A REAL LINEAR FN OF n REAL VBLES subject to linear constraints

"MOST" LP'S ARE READILY SOLVED BY HILL CLIMBING

THM IF AN OPT EXISTS VERTICES OF THE SOLN TO THE ϵ's



f = height

ALG: CHOOSE ANY VERTEX v
while A Better NBR w of v
Exists ($f(w) > f(v)$)
[$v = w$

(SIMPLEX)

SIMPLEX IS NOT P-TIME
But LP $\in$ P

KHACHIYAN (1979)

REVIEW SESSIONS FRI SAT 7:15 PM 1221 CS

Deductions

INFORMAL { x, y problems
x reduces to y
means I can use A "BLACK BOX" THAT SOLVES y to solve x

MAPPING REDUCTION ($\leq_m$)
(used in this COURSE $=_P$)

use the Black Box once

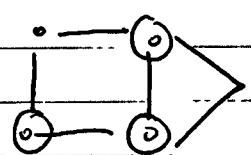
ANSWER (Yes/No)
MUST BE THE SAME

TURING REDUCTION ($\leq_T$)
CAN USE THE
BLACK BOX (POLYNOMIALLY)
MANY TIMES
Queries CAN BE ADAPTIVE
(K&T USES THIS)

Do we (using $\leq_m$) GET THE SAME
NP-comp PROBs AS K&T (using $\leq_T$)
Nobody knows.

APROXIMATION ALG's
IDEA: FIND SOLN's TO
NP-HARD OPT. PROBLEMS
GUARANTED TO BE "close"
TO OPTIMAL

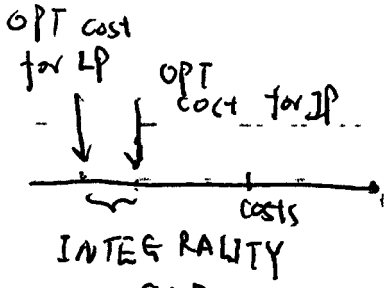
E.G. VERTEX COVER



$G = (V, E)$
$V = \{1, \ldots, n\}$

IP: $\max x_1 + \ldots + x_n$ ← cost
s.t $x_i + x_j \geq 1$ all $\{i, j\} \in E$
$0 \leq x_i \leq 1$ $x_i \in \mathbb{Z}$

RELAX TO LP:
$x_i \in \mathbb{R}$
Solving this gives
A fractional cover
cost might BE better AND is NO WORSE.



WE CAN PRODUCE A 0+1 cover by rounding

$x_i < \frac{1}{2} \Rightarrow x_i = 0$

$x_i \geq \frac{1}{2} \Rightarrow x_i = 1$

THIS "APPROXIMATE OPT" IS A COVER

$$\left[\begin{array}{c}\text{Cost of} \\ \text{"Rounding"} \\ \text{cover}\end{array}\right] \leq 2\left[\begin{array}{c}\text{cost of} \\ \text{FRACTIONAL COVER}\end{array}\right]$$

$$\leq 2\left[\begin{array}{c}\text{cost of MIN 0-1} \\ \text{cover}\end{array}\right]$$

CS 577 REVIEW I

1. $f(n) \in O(h(n))$

$\exists\ n_0$ s.t. $\forall n \geq n_0 \quad f(n) \leq c \cdot h(n)$

example: $f(n) = n \quad h(n) = 2^n, \quad g(n) = \log n$

2. $\log n \quad \sqrt{n} \quad n \quad n \log n \quad n^2 \quad 2^n \quad 2^{\sqrt{n}} \quad 2^{n \log n}$

$$\log_b a \cdot n = \frac{\log_c n}{\log_c b}$$

plot

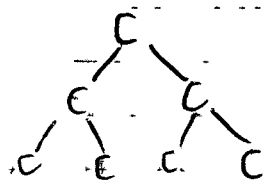
3. $\lim_{n \to \infty} \frac{\log n}{\sqrt{n}} = \lim_{n \to \infty} \frac{1/n}{\frac{1}{2\sqrt{n}}} = \lim_{n \to \infty} \frac{1}{\frac{1}{2\sqrt{n}}} = 0$

3. (a) i. $T(n) = 2T(n-1) + c$.
$T(1) = c$

2. $T(n) = T(\varepsilon n) + cn^2$
$T(1) = c$

3. $T(n) = \sqrt{n}\,T(\sqrt{n}) + cn$.
$T(1) = c$.

(b) i.



$$T(n) = \sum_{i=1}^{n} 2^{i-1} c$$
$$= c \sum_{i=0}^{n-1} 2^i$$
$$= (2^n - 1)c$$

$T(n+1) = 2c(2^n - 1) + c$

$= 2c(2^n - 1) + c = c2^{n+1} - c$

2. $\log_{1/\varepsilon} n$

$\begin{cases} cn^2 \\ \frac{1}{c(\varepsilon n)^2} \\ \frac{1}{c(\varepsilon^2 n)^2} \\ \frac{1}{c(\varepsilon^3 n)^2} \end{cases}$

$T(n) = \sum_{i=0}^{\log_{1/\varepsilon} n} c(\varepsilon^i n)^2 = cn^2 \sum_{i=0}^{\log_{1/\varepsilon} n} (\varepsilon^2)^i$

$0 < \varepsilon < 1$

$[x] = \max\{y \in \mathbb{Z} : y \le x\}$

$\le cn^2 \sum_{i=0}^{\infty} \varepsilon^{2i}$

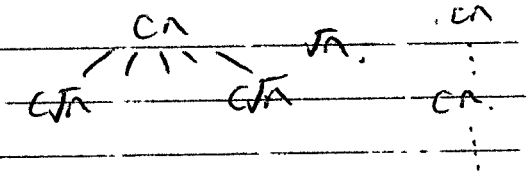
$= cn^2 \left(\frac{1}{1-\varepsilon^2}\right)$

$T(n) = k(\varepsilon n)^2 + cn^2 = (k\varepsilon^2 + c)n^2$

$k n^2 = (k\varepsilon^2 + c)n^2$

$k = \frac{c}{1 - \varepsilon^2}$

3) $T(n) = \sqrt{n}\, T(\sqrt{n}) + cn$



$T(n) \stackrel{?}{=} S(n)n$

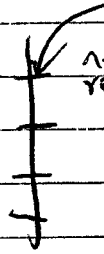
4. Base case $n=1$ $2 = \binom{n+1}{2} + 1 = 2\sqrt{}$

Hypothesis # regions w/ n lines is $\binom{n+1}{2} + 1$

Inductive step $H(n+1) = H(n) + n + 1$

$= \frac{n(n-1)}{2} + 1 + n + 1$

$= \left(\frac{n+1}{2}\right) + 1$



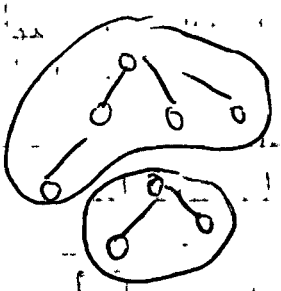
Base: 0 (1 node)

Hypthesis: # edges in $T_n = n - 1$

Inductive step

Pick e | Remove e:

Consider Tree T_{n+1}



T_{k_1} # of edges $k_1 - 1$

T_{k_2} $k_2 - 1$

$k_1 + k_2 = n + 1$

$T_{k_1} + T_{k_2} = k_1 - 1 + k_2 - 1 + 1$

$= n$

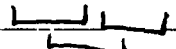
Assume hypth holds for all $T_1, \ldots, T_n$

Review 2 Divide and Conquer

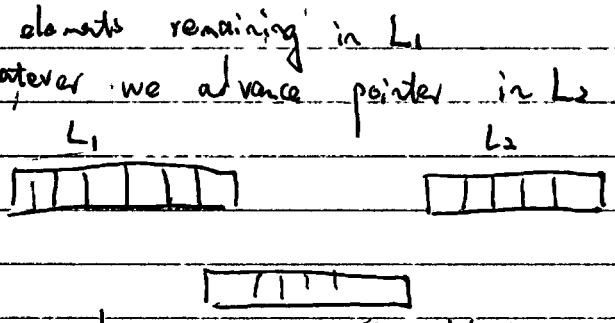
CS 1221

Problem 1 (significant)

Algorithm for counting inversions



- split list in half
- recursively sort and count inversion in each half
- merge sorted list L_1, L_2 into a sorted whole, adding # elements remaining in L_1 whenever we advance pointer in L_2



merge and sort separately count ++=4

First. sort L_1 and L_2

Initialize pointer to point to first element of $L_1 + L_2$

while both lists are nonempty.

Let a_i, b_j pointed by P_1, P_2

If $a_i > 2b_j$

Add # elements remaining in L_1

Advance P_2

Else

Advance P_1

Problem 2

① start by things possible split into subproblems

② pay attention to run time

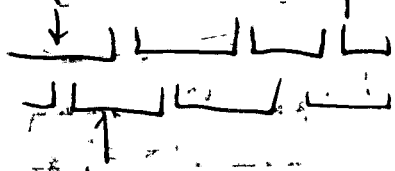
Observation 0: From left to right, segments uppermost lines must be strictly increasing slope

run time. $T(n) \leq 2T(n/2) + cn \Rightarrow T(n) = O(n \log n)$

Observation 1: Each visible line has an interval on which it is uppermost. A line visible at first half becomes invisible iff the left in second half intersection with intervals are steeper or those intervals

Observation 2: Given an interval, a line in first half and a line in second half, one can determine if the second line is greater on interval in $O(1)$ time

Observation 3: We can find the set of intersection intervals second half for every interval in first half in $O(n)$ time



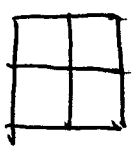
Idea: Sort lines in increasing order of slope recursively find visible lines in first half of list

$O(1)$ time
$O(n)$ time [For each visible line in each half, determine interval on which it is uppermost

$O(n)$ total [For each visible line in first set, find all visible lines in set of path with left with intersecting intervals, then test whether visible.

Run for lines in second half

Problem 3 Q: Does local min always exists
A: a global min must exist and this will be a local min



the boundary of the division is the set of all square

Observation: If local min of a boundary, then it will be a local min

find local min(A, n)

Input $n \times n$ grid (A, square b in A)

output a local min of A with $\leq$ b

Problem 4

Algorithm

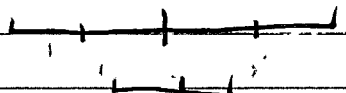
- split the cards into two halves
- recursively determine whether each half contains $\geq n/4$ cards that belong to the same account.

Observation If $\geq n$ cards belong to the same account, at least one of the halves must contain $\geq n/4$ cards belonging to that account.

- let c_1, c_2 be cards belongs to majority accounts in each half. if they exist, then test c_1, c_2 against n cards to see if $\geq n/4$ belong to the same account.

Review 3

Divide & Conquer Graph



we only need consider across subarray

$T(n) = 2T(\frac{n}{2}) + cn$

suffix

prefix

Correct Answer

Runtime ✓

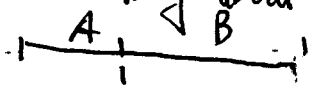
Induction on n:

Base $n=1$ only one number

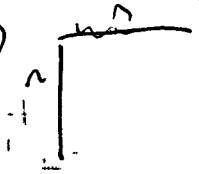
Is Assume correctness works for $\leq n$.

Consider $n+1$

☆ We only need to worry about crossing the middle!!



1. (a)



$$\begin{bmatrix} A & B \\ C & A \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} Ax + By \\ Cx + Ay \end{bmatrix}$$

$T(n) = 3T\left(\frac{n}{2}\right) + cn$ //

$\alpha = (A+C)x$
$\beta = (A+B)y$
$\gamma = A(x-y)$

	x	y
A	①	②
B		①
C	②	

$$\begin{bmatrix} \beta+\gamma \\ \alpha-\gamma \end{bmatrix}$$

A+C, A+B, all have property T

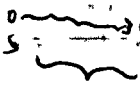
Ind on $n = 2^k$

Base $2^0 = 1$ integer multi

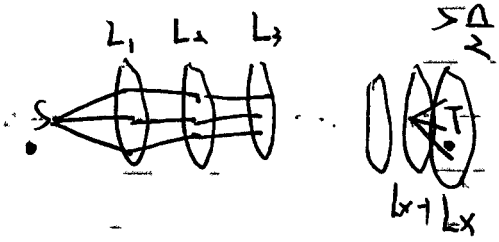
Is: Assume correctness for 2^k

2^{k+1}

3. (b) $G = (V, E)$ $|V| = n$ $|E| = m$ $s, t \in V$



$\geq \frac{n}{2}$



$x > \frac{n}{2}$

Assume every layer has $\frac{n}{2}$ vertex

$\Rightarrow \left(\frac{n}{2}\right) \cdot 2 = n$

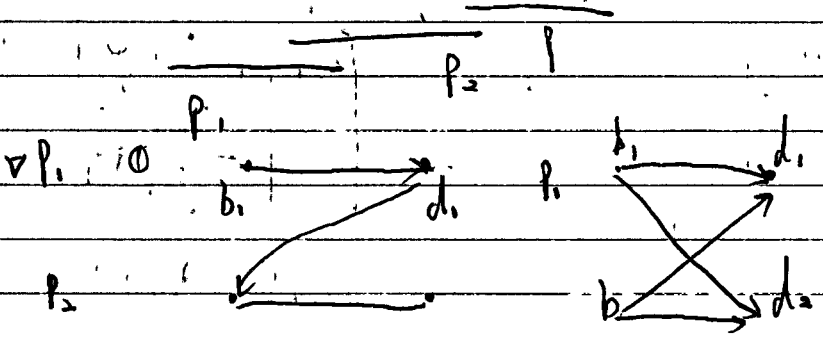
but all have $\frac{n}{2}$ ×

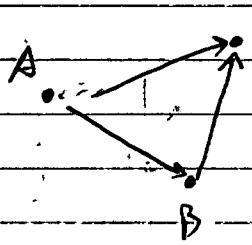
BFS: $O(m+n) + O(n) = O(m+n)$

P_i, P_j

① P_i died b/f P_i was born

②





A BC

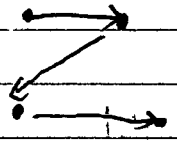
Topological ordering

$O(m+n)$

$\Theta(|V|+|E|) + \Theta(n)$

$\Theta(n)$ $\Theta(n)$

create graph $\rightarrow \Theta(n)$

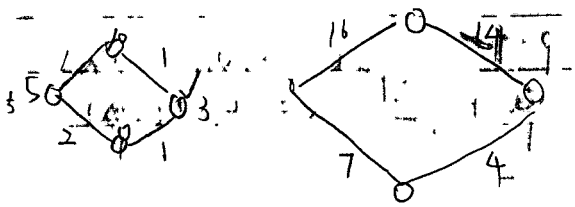


Correctness — Partial construct

Running time — Termination

Review 4

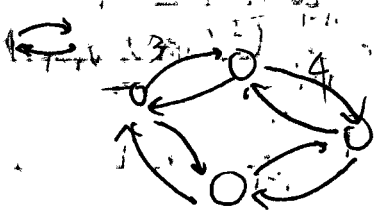
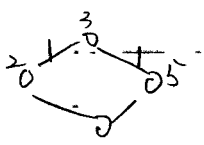
#3 I { Given $G=(V,E)$, $\{c_e\}_{e\in E}$, $\{c_v\}_{v\in V}$, $s\in V$
Output cost(v), min cost from s to v

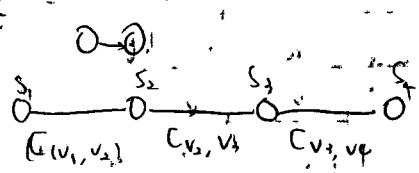


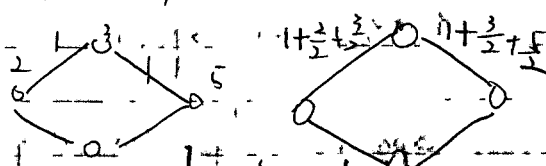
Reduction

$P_1 \longrightarrow P_2 \longleftarrow$ have alg

move vertex cost to edge cost

if  $d(v)$ $+c_v$ 





ALG: Convert I to I': For each edge $(s,v)\in E$,

set $c'_{(s,v)} = c_s + c_{(s,v)} + \frac{1}{2}c_v$

For each other edge $(u,v)\in E$,

set $c'_{(u,v)} = c_{(u,v)} + \frac{1}{2}(c_u + c_v)$

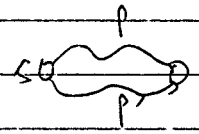
Run Dijkstra on I' to get $D'(v)$ to get $cost_{I'}(v)$

set: $cost_I(v) = cost_{I'}(v) + \frac{1}{2}c_v$

Run time $O(m \log n + n)$ $\quad m \le n^2$

claim Let P be a path $s, v_1, \ldots, v_n$

$cost(P) + \frac{1}{2}c_{v_n} = cost_I(P)$



P is shortest in I

$cost_{I'}(P) \le cost_{I'}(P') = cost_I(P') - \frac{1}{2}c_v$

$\|$

$cost_I(P) - \frac{1}{2}c_v$

Pf claim: $cost_{I'}(P) = c_s + c_{(s,v_1)} + \frac{1}{2}c_{v_1}$

$+ \sum_{i=1}^{n-1} \left(\frac{1}{2}c_{(v_i, v_{i+1})} + \frac{1}{2}(c_{v_i} + c_{v_{i+1}})\right)$

$= c_s + \sum_{i=1}^{n} c_{v_i} + c_{(s,v)} + \sum_{i=1}^{n-1} c_{(v_i, v_{i+1})}$

$-\frac{1}{2}c_{v_n}$

$= cost_I(P) - \frac{1}{2}c_{v_n}$

#4 Prim's start cuts grow cc by cheapest edge if no cycle

Kruskal's sort edges acc weight

add if no cycle.

Reverse-delete sort in decreasing weight

throw most expensive

Thm 4.2p

can delete most expensive edge on any cycle.

ALG Find cycle → BFS $O(n)$

Delete most expensive edge $O(n)$

Repeat until no cycle 9 times

Runtime: $O(9n)$

#1 (p_i, b_i, r_i) (2, 4, 7)
(8, 5, 3)

p	b	r
2		
8	4	
	5	7
		3

(p_i, o_i)

(2, 1) ⇒ 13
(3, 7)

(3, 7) ⇒ #1b
(2, 11)

✓ Sort by decreasing o_i $O(n \log n)$
× Sort by increasing p_i

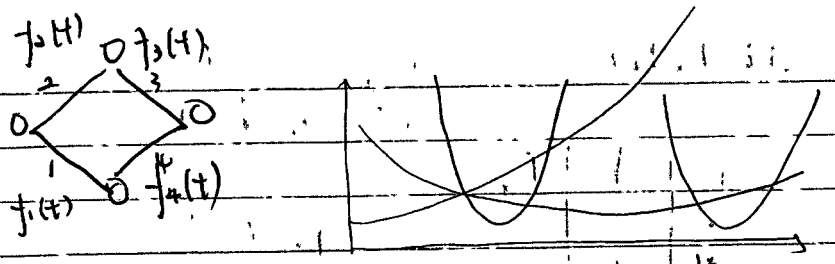
(1, 0) ×n
(2, n)
⇓
2n+2 n+2

claim Let σ be any other ordering of contestants. Then can find i, j $o_i < o_j$, but i appears immediately before j in σ.

We can swap i,j without increasing total time.

(p_i, o_i) t
(p_j, o_j) p_i
p_j
$t + p_i + p_j$

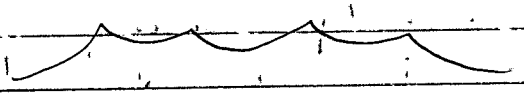
#4



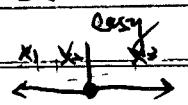
$f_1(t) = f_2(t)$

$f_1(t) - f_2(t) = 0$ 2 times intersection.

$f_1(t) + f_2(t) + f_3(t) = (a_1+a_2+a_3)t^2 + (b_1+b_2+b_3)t + c_1+c_2+c_3$ $\quad 2\binom{n}{2} \quad O(n^2)$



Review 5 Greed & DP



Partial correctness.

Termination

Running Time $O(n\log n) + O(n)$

Review 6.

$OPT[i,j]$ = max # TB on days $i \cdots n$ if rebooted j days ago

$OPT[i,j] = \max \begin{cases} \min(x_i, s_j) + OPT[i+1, j+1] & \text{Do not Reboot} \\ 0 + OPT(i+1, 1) & \text{Reboot} \end{cases}$

sol n $OPT[1,1]$

$OPT[n,j]$ $OPT[n,1] = \min(x_n, s_1)$

$= \min(x_n, s_j)$

For $i = n-1, \ldots, 1$

For $j = 1 \ldots n$

$\cdots OPT =$

return $OPT[1,1]$

	1	2	3
X	1	10	100

2. $OPT[i, v]$ = min weight to achieve v with $i \dots n$

W $V = \sum_{i=1}^{n} v_i$

$$= \min_{S \subseteq \{i, \dots n\}, \sum_{i \in S} v_i = V} \left\{ \sum_{i \in S} w_i \right\} \quad \min \begin{cases} w_i + OPT[i+1, V - v_i] \\ OPT[i+1, v] \end{cases}$$

$$\max \{ v : OPT[1, v] \leq W \}$$

$OPT[n+1, v] = \infty \quad \forall \; v > 0$

$OPT[n+1, 0] = 0$

return $\max \{ v : OPT[1, v] \leq W \}$

d	1	2	3	4
	2	50	12	6

$k = 5$

$S = 10$

$c = 2$

3. $OPT(i)$ = min cost to satisfy demands thru month i leaving 0 tracks

return $OPT(n)$

$OPT[0] = 0$

$OPT[1] = 5$

$OPT(i, j)$ = min cost to satisfy demands thru i leaving j tracks

return $OPT[n, 0]$

$$= \min_{j=0,\ldots,i} \left\{ T[j-1] + K + C \cdot \sum_{l=j+1}^{i} \sum_{k=l}^{6} d_K \right\}$$

Assume demands

$i=4$	50	6
$j=1$	10	6
		6
		6

$1 \to 2$

$2 \to 3$

stories $3-4$

P1 $G = c(V,E)$ $T \subset G$, $c: V \times V \to Z_{>0}$, $e \in T$

$c(e) \leftarrow c'(e) > c(e)$

cut property: $S \subset V$, $1 < |S| < |V|$ then the mini-cost edge (u,v) where $u \in S$ and $v \in V-S$ is in MST

let $e = (u,v)$, Run $\to O(n)$ DFS from u in T call these nodes L, for each edge (u',v'), if $u' \in L$ and $v' \notin L$, add (u',v') set C, return $(T - e) +$ min cost edge in C. $O(m)$

$G = (V, E)$ $n = |V|$ $m = |E|$

Overall

Correctness by cut property, min-cost edge crossing $L, V-L$ must be in MST T restricted to $L, V-L$ is still optimal (any better spanning tree on $L, V-L$ would have better in original problem (min)

P2. k sorted list $L_1, \dots L_k$

$\sum_{i=1}^{k} L_i = n$

Alg create min PQ P

For $i = 1 \dots k$

Add $(L[0], i)$ to P

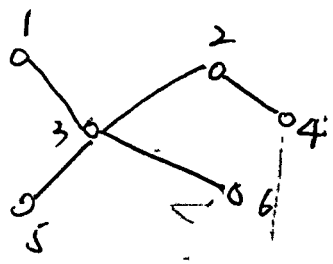
while $|P| > 0$

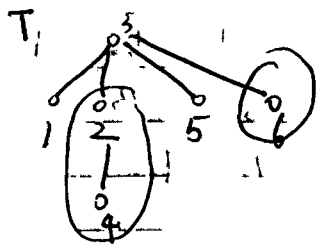
Take min (e, i), add e to output,

add (next[L], i) to P

correctness: At beginning of each iteration P contains next element from each list, we add smallest to output, and replace it until next smallest element from corresponding list

so invariant continues to hold





Define $E(v)$ to be length of longest path in T_v which ends at v_i

$$E(a) = \max \begin{cases} c(a,b) \text{ if } E(b), \\ c(a,c) + E(c) \\ c(a,d) + E(d) \end{cases}$$

Define $opT(v)$ to be length of largest path in T_v which includes v_i $\max opT(v)$ $v \in$

Define $S(v)$ to be v's children

$$E(v) = \max_{v \in S(w)} \{ c(v,v) + E(v) \}$$

$$opT(v) = \max \begin{cases} E(v) & \text{if } |S(a)| = 1 \\ \max_{w \in S(v)} \{ \end{cases}$$

WK 9 Fri

Quicksort:

$x_1, \ldots, x_n$ DISTINCT

CHOOSE ONE INPUTS x AS THE SPLITTER

GROUP KEYS SO THAT

$L = x_1, \ldots x_{n-1} < x$

$x = x_i$

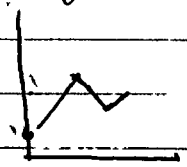
Problem 1

A = [1, 5, 7, 2, 3]

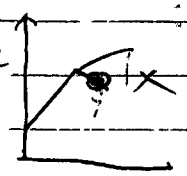
1, 5, 2, 3 ✓

1, 5, 7 ✗

Zig-Zag sequence



OPT_d[i] : length of the longest zig zag sequence ending at i

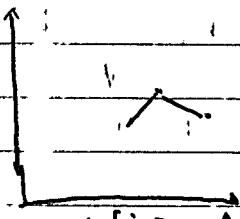


OPT_u[i]: upward.

Base: OPT_d[1] = 1

OPT_u[1] = 1

Recursive $OPT_d[i] = \max\{OPT_u[j] + 1\}$

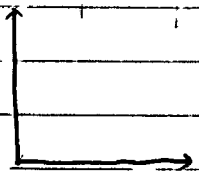


$1 \le j < i \quad A[j] > A[i]$

OPT_u[i]

$= \max\{OPT_d[j] + \}$

$1 \le j < i \quad A[j] < A[i]$



OPT [i] = length of sequence up to i

OPT [1] = 1

OPT [i] = OPT [i-1] + { if A[i-1] < A[i] > A[i+1] (or > or =)

o.w OPT [i-1]

OPT [i] length

OPT [1] = 1

opt[i] = max{ OPT

Problem 2 $n = 2^k - 1$

A [1, ..., n]

# of k bit integer 2^k

bit [i, j] = jth bit of ith

$\theta(n)$

$2^{k-1} - 1 + 2^{k-1} = 2^{k-1} + 2^{k-1} - 1$

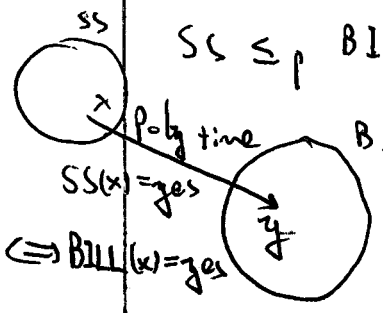
CS 577

Final Review

Topics: Practice Exam
study tips
Questions

$i \in A$ $\quad$ $j \in B$

a_i $\quad$ b_j



$SS \leq_p$ BILL

BILL is at least as hard as SS

IF BILL is easy...
SS is also easy

$\{a_i\}_{1 \leq i \leq n}$ $\quad k$ $\quad \sum_{i \in S} a_i = k$

$S \subseteq [n]$

Assumption: the check only go from A to B

$\{a_i\} = \{a_{g(i)}\}$

$b_1 = k_i$ $\quad i \in [n]-S$

$b_2 = \sum a_i - k$

Claim: $\forall$

$\Leftrightarrow \quad S \leq \sum a_i - k$

Pf. For each person

$\Leftrightarrow \quad i \in S \quad i \xrightarrow{a} 1$

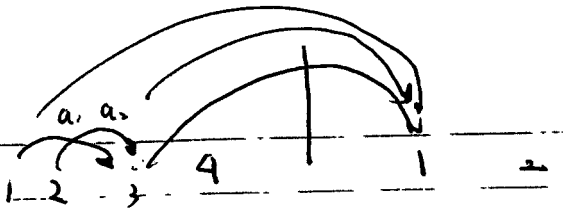
$i \notin S \quad i \xrightarrow{a} 2$

P1: claim: Each person $i \in A$

$\Rightarrow$: must write a check with exactly a_i

b_1 $\quad \sum a_i = k$

$S: \{i \mid i$ pays person $1\}$



$a_1 + a_2 + a_3$

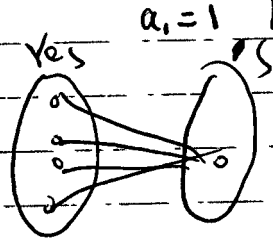
$\{a_i\}$ [^]

$b_1 = k$ O(n)
$b_2 = \Sigma a_i - k$

Yes

Yes[i] = yes

Yes $\leq_p$ $\{\}$



BILL is NP – Complete
⇑
NP

P2: CSL