CMPS 201 5-19-10

Problem:

Given A[1....n] of integers, find max value & it's index in A[].

BASIC operation: Comparison of Array elements.

Decision tree lower bound

#outcomes per test = 2

#verdicts = n

$h \geq \lceil \log_2 n \rceil$

i.e. worst case # comp $\geq \lceil \lg n \rceil = \Omega(\log n)$.

But Best Known Algorithm is:

FindMax(A)

1.) n ← length[A], max ← A[1], K ← 1
2.) for i ← 2 to n
3.) if A[i] > max
4.) max ← A[i]
5.) K ← i
6.) return (max, K)

comp = n − 1 > $\lceil \log_2 n \rceil$

EXERCISE:
Draw A decision tree representation of this algorithm in case n = 4.
note this tree has height 3 not 2.

note $n=4$ gives

$$\lceil \log_2 n \rceil = 2$$

$$n-1 = 3$$

Case: $n = 1001$

$$\lceil \log_2 n \rceil = 10$$

$$n-1 = 1000$$

Adversary Argument

Consider any Comparison based algorithm for this problem, let it run on $A[1 \ldots n]$ unspecified.

Daemon's strategy:

Answer each Question: "$A[i] < A[j]$?"

As if $A[i] = i$, i.e. as if

$$A = (1, 2, 3, \ldots, n).$$

i.e. Daemon answers "$A[i] < A[j]$?"

$$\begin{cases} \text{true} & \text{if } i < j \quad \text{"} i \text{ lost comp."} \\ \text{false} & \text{if } j < i \quad \text{"} j \text{ lost comp."} \end{cases}$$

we will say that the smaller index has "lost a comparison"

Now Assume that the Algorithm Does fewer than $n-1$ comparisons before it halts and returns

an Answer : $(A[k], k)$.

i.e. Algorithm claims $A[k]$ is maximum in $A[\,]$

Let i be an index satisfying

- $1 \le i \le n$
- $i \neq k$
- i has not lost any comparisons.

i must exist since

$$\#\text{losers} \le \#\text{comparisons} \le n-2$$

- $\#\text{losers} \le \#\text{comparisons}$: since each comp. creates at most 1 new loser.
- $\#\text{comparisons} \le n-2$: by assumption

At this point the Daemon can claim

$$A[i] = \begin{cases} i & i \neq j \\ n+1 & i = j \end{cases}$$

Indeed $A[k] = k$ is not maximum in this array, but all of the Daemons answers are consistent with this $A[1..n]$. $\therefore$ Algorithm is incorrect.

$\therefore$ any correct algorithm must do at least $n-1$ comparisons.

Problem: Graph Connectivity

let $G = (V, E)$ be an (undirected) Graph with $|V| = n$.

Determine: whether or not G is connected.

we consider only Algorithms that ask Questions of the form

"Is Vertex x adjacent to Vertex y?"

i.e. 'adjacency question'
or 'edge probe'

Decision tree lower bound:

outcomes per probe = 2

verdicts = 2

$\#\text{probes} = h \geq \lceil \log_2 2 \rceil = 1$.

Depth first search (DFS) solves this in time $\Omega(n^2)$ (see. sec. 22.3 2nd ed.)

Adversary lower bound:

Consider any "adjacency based" algorithm for this problem. start on some unspecified

Graph $G = (V, E)$ with

$n = |V|$.

Daemonic strategy

Partition V into two subsets

X, Y sizes $\lceil \frac{n}{2} \rceil, \lfloor \frac{n}{2} \rfloor$ resp.

i.e.

$$X \cup Y = V$$

$$X \cap Y = \phi$$

$$|X| = \lfloor \tfrac{n}{2} \rfloor$$

$$|Y| = \lceil \tfrac{n}{2} \rceil$$

whenever algorithm probes the pair (u,v) by asking "is u adjacent to v?" the Daemon answers "yes" if u,v belong to the same set:

$$\begin{cases} \text{yes if } u,v \in X \text{ or } u,v \in Y \\ \text{No if } u \in X \text{ and } v \in Y \\ \quad \text{or} \\ \quad u \in Y \text{ and } v \in X \end{cases}$$

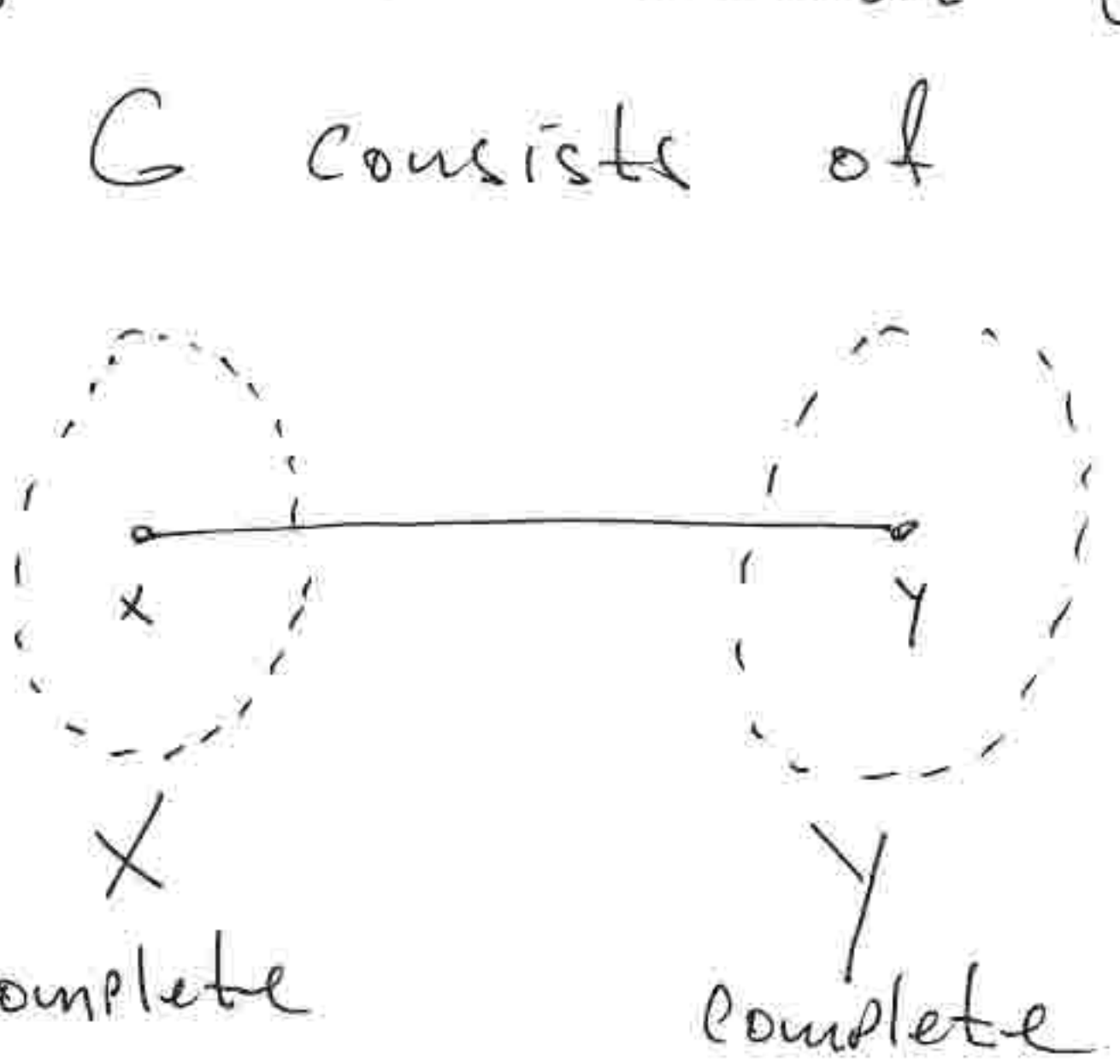
i.e. He pretends G consists of two disjoint pieces: complete graphs on X & Y resp.

Now Assume the Algorithm Halts and returns an answer (conn./ disconn.) after doing fewer than $M = \lfloor \frac{n}{2} \rfloor \cdot \lceil \frac{n}{2} \rceil$ edge probes. Then there must exist vertices $x \in X$ and $y \in Y$ such that the algorithm has not probed (x, y).

If Algorithm says G is conn, then Daemon can claim that G consists of two disjoint

complete graphs on X and Y. note this is consistent with all Daemons answers.

If Algorithm says C is discon., then Daemon can claim C consists of



This Graph is also consistent with entire seq. of Damons answers. (since (x,y) was never Probed.)

$\therefore$ Algorithm is incorrect.

$\therefore$ Any Correct Algorithm must do at least $M = \lfloor \frac{n}{2} \rfloor \lceil \frac{n}{2} \rceil$ edge probes.

Note: $M = \Omega(n^2)$

Conclude: Can't improve on DFS (in Asymptotic sense.)

modulo some re-definition

note $\lfloor c_1 n \rfloor = (c_1 n - 1) + O(n)$ $(c_1 > 0)$

$\lceil c_2 n \rceil = c_2 n + O(n)$ $(c_2 > 0)$

so $\lfloor \frac{n}{2} \rfloor \lceil \frac{n}{2} \rceil = \frac{1}{4} n^2 + O(n^2)$

Can't improve on n^2, but can improve on the $\frac{1}{4}$.

See Adversary Handout for proof of

Thm: Any adjacency Based algorithm for connectivity Problem must do at least $\binom{n}{2}$ edge Probes.

recall:

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

$$= \frac{1}{2}n^2 - \frac{1}{2}n$$

$\frac{1}{2}$ is better than $\frac{1}{4}$.

well do this next time.

Problem

let $B = b_1 b_2 b_3 b_4 b_5$ be a bit string of length 5. Determine whether or not B contains 3 consecutive 1's, i.e. the substring "111".

we consider algorithms whose basic operation is to peek at a bit. i.e. to ask "is $b_i = 1$ or is $b_i = 0$?"

note: obviously 5 peeks are sufficient.

In fact only 4 peeks are necessary.

Exercise:

a.) Give an algorithm that solves problem in 4 peeks

b.) Give Adv. argument showing you can't do it in 3 peeks.