CS3203 #7

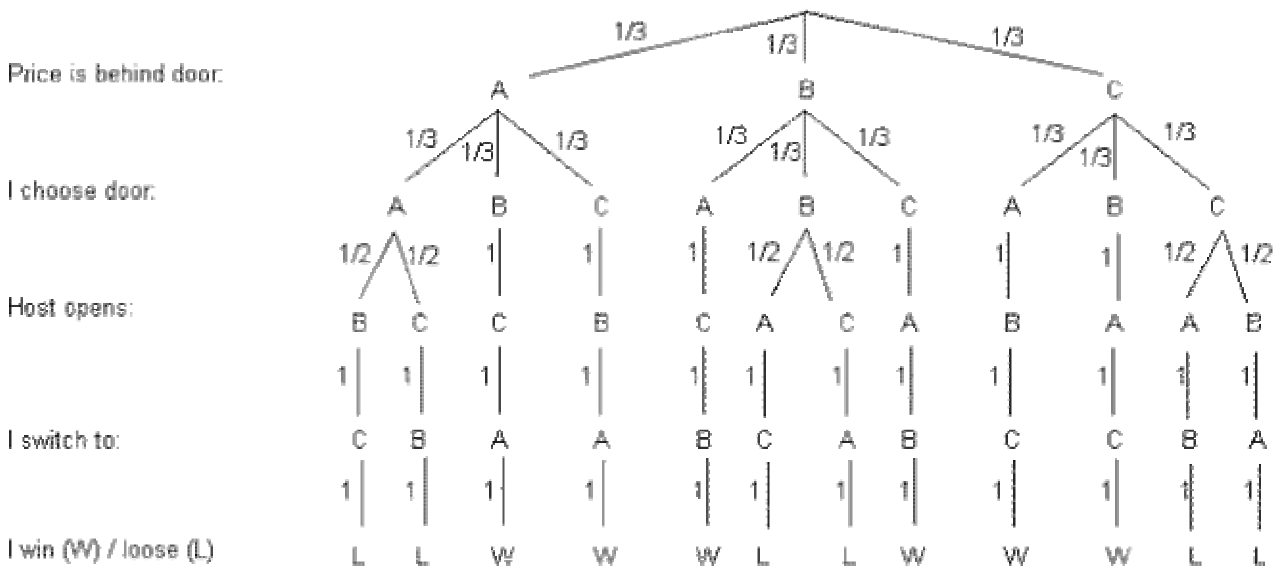
6/16/04

Janak J Parekh

Administrivia

- Exam will be returned next week
  - Any comments?

Monty Hall, redux



- Now is the chance of loosing: L = 1/3 · 1/3 · ½ · 1 · 1 = 1/18
- And I got 6 L's so: Total chance for loosing is: 6· 1/18 = 1/3
- For winning: W = 1/3 · 1/3 ·1 · 1 · 1 = 1/9
- And I got 6 W's so: Total chance for winning is: 6 · 1/9 = 2/3. (check: 2/3 + 1/3 = 1 (OK))
- From http://www.cut-the-knot.org/peter.shtml

Birthday Paradox

- How many people are needed in the room such that it's more likely than not (e.g., greater than .5 probability) that two people have the same birthday?
  - We assume that birthdays are independent, equally likely, and 366 birthdays per year.
  - If p_n = probability have all different birthdays, then $1 – p_n$ = probability two people have the same birthday
  - Compute probability has a different birthday as people "walk in the room".
  - First person $p_1 = 1$, second is 365/366, third is 364, 366, etc.
  - p_n is therefore 1 * 365/366 * 364/366 * 363/366 * … * 367-n/366, and $1 – p_n$ is 1 – same thing.
  - Use formula for $1 – p_n$ until it becomes greater than ½, and we have our value n. $1 – p_n \sim 0.475$ for n = 22, $1 – p_n \sim 0.506$ for n = 23.
- Should we try for months in this room?

Monte Carlo algorithms

- “Probabilistic” algorithms are those that make “random” choices at one or more steps
  - Useful when you’ve got an algorithm where a deterministic algorithm goes through a huge number of choices
- Monte Carlo – specific subcategory of probabilistic algorithms
  - Always produce answers, but small probability remains the answers are incorrect
  - Given sufficient computation, chance that algorithm is incorrect decreases
  - For “decision problems”, MC algorithms use a sequence of tests. At each step, possible responses are “true”, which means no more computation needed, or “unknown”, which means either “true” or “false”.
  - “False” is accomplished if, for all computation, we still have “unknown”.
  - For any $p > 0$, $(1-p)^n$ (“unknown”) shrinks

Example

- Chip testing
- PC manufacturer orders processor chips in batches of size *n*, where *n* is a positive integer
- Chip maker only tests a few batches
- Random testing shows a 10% failure rate
- But to test a chip takes O(n) time for *n* tests
- Select a random subset of chips and test them
  - Question: “Has this batch of chips not been tested by the chip maker?”
  - If a bad chip is encountered, answer “true” and stop
  - If a tested chip is good, “unknown”
  - After *k* chips, answer “false”
- Only possible incorrect answer is “false”
- Probability that a chip is good but that it came from an untested branch is 1 – 0.1 = 0.9. 0.9^k for arbitrary *k* chips.
  - If we test 66 chips, $1 - 0.9^{66} < 0.001$ chance the algorithm decides a batch has been tested, i.e., less than 1-in-1000 chance that the algorithm has answered incorrectly
  - 132 tests imply error rate to less than 1 in 1,000,000

Probabilistic method

- We’re not doing this, you can check the book if you want

Advanced Counting

- Simple example
  - The number of bacteria in a colony doubles every hour. If a colony begins with 5 bacteria, how many will be present in *n* hours?
    - $a_0 = 5$
    - $a_n = 2a_{n-1}$, where n is the # of hours.
- We have just found a "recurrence relation".
  - Very similar to recursive algorithm, but here we'll focus on counting techniques
  - How do we take the aforementioned equation and come up with a "explicit" formula?
- To be precise, a **recurrence relation** for the sequence $\{a_n\}$ is an equation that expresses a_n in terms of one or more of the previous sequence, namely $a_0, a_1, \ldots, a_{n-1}$ for all integers $n >= n_0$, where n_0 is nonnegative.
- A sequence is called the **solution** of a recurrence relation if its terms satisfy the recurrence relation.

Examples

- Let $\{a_n\}$ be a sequence that satisfies $a_n = a_{n-1} – a_{n-2}$ for n = 2, 3, 4, … and $a_0 = 3$ and $a_1 = 5$.
- **Initial conditions** specify the terms that precede the first term where the recurrence relation takes effect, as in the example above.
  – Initial conditions plus the recurrence relation uniquely determine a sequence.
- Can use to model problems…
- Deposit $10,000 in a savings account in a bank yielding 11% per year, interest compounded annually; how much is in the account after 30 years?
  – What's the explicit equation? $P_n = (1.11)^n P_0$. In general, 1+r
  – Can use induction to prove.
- Rabbits can be modeled by Fibonacci?
  – A pair of rabbits (one of each gender) is placed on an island. They don't breed until they're two months old. After 2 months, each pair of rabbits produces another pair each month.
  – $f_1 = 1$, $f_2 = 1$, $f_3 = f_2 + f_1$, $f_n = f_{n-1} + f_{n-2}$ (the n-2 term are the newborns as they come from rabbits at least two months old)
- Bit strings of length *n* that do not have two consecutive zeros – how many such bit strings are there? Give a recurrence relation and an example for length 5.
  – a_n = # of bitstrings of length *n* that do not have two consecutive zeros.
  – Either take a bitstring of length n-1 and add a 1, or a bitstring of length n-2 and add a 10.
  – Again, fibonacci!

Solving recurrence relations

- We can sometimes do it naively, but it rapidly gets complicated
  - Try to “spot a pattern”
- There are several “standard forms”
- **Linear homogenous recurrence relation of degree k with constant coefficients** is a recurrence relation of the form
  - $a_n = c_1a_{n-1} + c_2a_{n-2} + \ldots + c_ka_{n-k}$, where $c_1 \ldots c_k$ are real numbers, c_k != 0. Note intermediate terms can be zero, however.
- Examples
  - $P_n = (1.11)P_{n-1}$ is of degree one.
  - $f_n = f_{n-1} + f_{n-2}$ is of degree two.
  - $a_n = a_{n-5}$ is of degree 5.
- What’s not?
  - $a_n = a_{n-1} + a_{n-2}^2$ (not linear)
  - $H_n = 2H_{n-1} + 1$ (not homogenous)
  - $B_n = nB_{n-1}$ (not constant coefficients)

Degree one

- For $a_n = c_1 a_{n-1}$
- Solution is $a_n = a_0 c_1^n$
- Easy enough…
- Can we generalize the strategy of raising it to a power for more complex linear homogenous recurrence relations?

Degree two

- Look for solutions of the form $a_n = r^n$, where r is a constant. Note that this is only a solution if
  - $r^n = c_1 r^{n-1} + c_2 r^{n-2} + \ldots + c_k r^{n-k}$
- Divide both sides by r^{n-k} and subtract the right hand side from the left
  - $r^k - c_1 r^{k-1} - c_2 r^{k-2} - \ldots - c_{k-1} r - c_k = 0$
  - Only a solution if *r* is a solution of this last equation: **characteristic equation** of the recurrence relation. Solutions are called the **characteristic roots**.
- For degree two, there may be one or two characteristic roots
  - Let c_1 and c_2 be real numbers. Suppose that $r^2 - c_1 r - c_2 = 0$ has two distinct roots r_1 and r_2. Then the sequence $\{a_n\}$ is a solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ if and only if $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$ for n = 0, 1, 2, … and α_1 and α_2 are constants.
  - Characteristic roots may be complex numbers, but we won't deal with those

Examples

- Solution of the recurrence relation $a_n = a_{n-1} + 2a_{n-2}$ with $a_0 = 2$ and $a_1 = 7$?
  - Solve $r^2 – r – 2 = 0$ ($r = 2$ and $r = -1$)
  - So, $a_n = \alpha_1 2^{n} + \alpha_2 -1^{n.}$
  - Plug in a_0 and a_1 to determine α values.
  - Solution: $a_n = 3*2^n – (-1)^n$.
- Fibonacci?
  - Characteristic equation is $r^2-r-1 = 0$. Ugh!
  - Solutions are on page 416
  - I'm not expecting you to remember this…
- $a_n = 2a_{n-1} + 3a_{n-2}$, $a_0 = 0$, $a_1 = 1$
  - $r^2-2r-3 = 0$, or $(r-3)(r+1)$
  - Final solution is $a_n = ¼ * 3^n - ¼ * (-1)^n$
- $a_n = 6a_{n-1} – 9a_{n-2}$
  - Solve $r^2-6r+9 = 0$
  - $(r-3)^2 = 0$?
  - Uh-oh…
  - Second theorem: $a_n = \alpha_1 r_0^n + \alpha_2 n r_0^n$
  - So, in this case $a_n = 3^n+n3^n = (n+1)3^n$

Generalized

- For $r^k – c_1r^{k-1}$ - … - $c_k = 0$ with distinct roots r_1, …, r_k, solution is
- $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \ldots + \alpha_k r_k^n$
- Again, I'm not expecting you to solve such annoying factorizations
- You can even generalize multiplicities – see the mess on page 418

Linear *non*homogeneous recurrence relations

- If $\{a_n^{(p)}\}$ is a particular solution of the nonhomogeneous linear recurrence relation with const. coeff.
  - $a_n = c_1a_{n-1} + c_2a_{n-2} + \ldots + c_ka_{n-k} + F(n)$
  - Then every solution is of the form $\{a_n^{(p)} + a_n^{(h)}\}$, where $\{a_n^{(h)}\}$ is a solution of the associated homogeneous recurrence relation $a_n = c_1a_{n-1} + c_2a_{n-2} + \ldots + c_ka_{n-k}$
- Why we don't do this?
  - Figuring out $a_n^{(h)}$ is *not* fun
  - Check out the rest of the section if you want…
  - Good luck!

Divide-and-conquer recurrence relations

- Example: binary search is a divide-and-conquer algorithm
  - Although it doesn't actually "conquer" much after dividing
  - Mergesort is another example
- Forms the recurrence relation
  - $f(n) = af(n/b) + g(n)$
  - "a" subproblems, each sized n/b, plus g(n) work to "combine"
- So, what's binary search?
  - $f(n) = f(n/2) + 2$
- Mergesort
  - $M(n) = 2M(n/2) + n$

Solving these explicitly

- If $f(n) = af(n/b) + c$,
- $f(n)$ is $O(n^{\log(b)a})$ if $a > 1$, or $O(\log n)$ if $a = 1$.
- When $n = b^k$, where k is a positive integer, $f(n) = C_1 n^{\log(b)a} + C_2$, where $C_1 = f(1) + c/(a-1)$ and $C_2 = -c/(a-1)$
- Just plug-and-play
- Generalization is the “Master theorem”

Master theorem

- If $f(n) = af(n/b) + cn^d$,
- $f(n)$ is:
  - $O(n^d)$ if $a < b^d$
  - $O(n^d \log n)$ if $a = b^d$,
  - $O(n^{\log(b)a})$ if $a > b^d$.
- Literally plug-and-play.
- Lots more of this in CS 4231.

Relations

- Relationships between sets occur in many contexts
  - Business and telephone numbers, employees and salary, etc.
  - Numbers and those that it divides, numbers and those congruent to mod m, etc.
- Special structure called a relation
  - A *binary relation* from A to B is a **subset** of A x B.
  - We use the notation a R b if (a, b) $\in$ R and a R b (where R is struck out) if they're not. If they are, a is said to be **related to** b by R.

Examples

- Let A be the set of all cities, and B be the set of the 50 states in the US. R specifies (a,b) if a is in b. So, (New York, New York), (Trenton, New Jersey), (Boston, Massachusetts), etc. are in R.
- Let A = {0,1,2} and B = {a,b}. Then R = {(0,a),(0,b),(1,a),(2,b)} is a relation. You can show this graphically or in tabular format as well.

Functions as relations

- Why not?
  - Since the graph of f (i.e., the set of ordered pairs (a,b) such that b = f(a)) is a subset of A x B, it is a relation from A to B.
- You can also define a function as one where R is its graph.
  - Just assign element a in A to be b in B such that (a,b) $\in$ R.
- Relation can be used to express a *many-to-many?* relationship between elements of the sets of A and B
  - So, a relation is a generalization of functions

“Self-”relations are useful…

- A relation on the set A is a relation from A to A.
  - Let A be the set {1, 2, 3, 4}; which ordered pairs are in the relation R = { (a, b) | a divides b}
- Can also define relations on infinite sets
  - R = {(a,b) | a < b}, for example
- How many relations on a set with *n* elements?
  - A x A has n^2 elements, and a set with m elements has 2^m subsets, so 2^(n^2) subsets of AxA.
  - 512 relations on {a, b, c}!

Properties of relations

- R on A is *reflexive* if (a, a) $\in$ R for every element a $\in$ A.
- A relation R on A is called symmetric if (b,a) $\in$ R whenever (a,b) $\in$ R for all a, b $\in$ A.
- A relation R on A is called *antisymmetric* if (a,b) and (b,a) $\in$ R only if a = b, for all a,b $\in$ A
  - Sort of a “weakly reflexive”
- A relation R on A is called transitive if whenever (a,b) and (b,c) $\in$ R, (a,c) $\in$ R, for all a,b,c $\in$ A

Examples

- Let R be the relation on {a, b, c, d}:
  - R = {(a,a), (a,c), (a,d), (b,a), (b,b), (b,c), (b,d), (c,b), (c,c), (d,b), (d,d))
  - We can draw a graph…
  - Is it
    - Reflexive? Yes.
    - Irreflexive? No.
    - Symmetric? No (a,c) / (c,a)
    - Asymmetric? No (b,c) and (c,b)
    - Antisymmetric? No (b,c) and (c,b)
    - Transitive? No (a,c) (c,b) no (a,b)

Next time

- Finish up relations