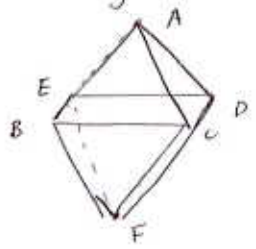
29 March 2012

Last time: Algorithm for Euler paths/circuits



start at A

Possible way: (mark of paths as used)

A − B − C − A D − E *

→ if I go to A, I get stuck @ A

→ Must choose B or F.

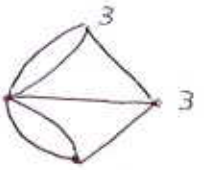
say "F" if * − F − B − E − A ⇒ Disconnect at D

if * − F − C − D − F − B − E − A

no choice (we want all of the "no choices" to come at the end)

Make choices so that you do not create connected components

Special case: Konigsberg



No Euler Path

(odd degree must be a path end point - we have too many odd ones here)

Hamilton Paths & Circuits

- Hamilton path in G_2 is a path that visits every vertex in G_2 exactly once. (expect to miss a lot of edges).

Fact: Testing a graph for having a Hamilton circuit is NP−complete.

↖ weak: it is hard to say helpful things about NP problems

Theorem: case where Hamiltonian path exists.

Dirac: G_2 must be simple. no loops, no multiple edges (unused edges are okay)

⇒ If the degree of every vertex $\geq \frac{1}{2}$ # vertices

then G_2 has a Hamilton Path. (not very natural...)

Examples of Unsolved Problems

↳ Not like "squaring a circle" ⇒ impossible

but like "can't decide what is the answer"

setup: Imagine a computer with ∞ memory (no physical constraints).
It runs programs of finite length using input of finite length.

Question: Given a program "i" and an input "x", will $i(x)$ return an answer in finite time or will it loop forever?

More precisely: is there a program "h" with inputs "i" and "x" such that $h(i,x) = \begin{cases} 1 & \text{if } i(x) \text{ terminates} \\ 0 & \text{if } i(x) \text{ runs forever} \end{cases}$

This is called the "halting problem."

Such a program h cannot exist... why?

Primer

Different sizes of infinity.

① Thm: $\mathbb{N}$ and $\mathbb{R}$ are of different size distinctly more reals than naturals.

② Thm: $\mathbb{N}$ and $\mathbb{Q}$ are of same size

$\mathbb{N}$ and $\mathbb{Q}$ have a one-to-one correspondence

$\mathbb{N}$ and $\mathbb{R}$ do not.

Pf ②

q \ p	1	2	3	4 ...
1	1/1	1/2	1/3	1/4
2	2/1	2/2	...	
3	3/1	...		
4				
⋮				

← every conceivable rational numbers are captured.

the pattern.

shows that such chart can be give a $\mathbb{N}$ number as an index.

⇒ $|\mathbb{N}| \geq |\text{squares}| = |\mathbb{Q}|$.

Proof ①

We want to show that the elements of $\mathbb{R}$ cannot be numerated.

Model real numbers as decimal expansions

If $0 \leq \lambda \leq 1$

$\lambda = 0.\lambda_1\lambda_2\lambda_3\ldots$ (~~if I show~~ λ_i

it is enough to show that the λ in $[0,1)$ that only 0 and 1 in their expansion cannot be enumerated.

Suppose we were given a complete list of these numbers.

$\lambda_1^{(1)}, \lambda^{(2)}, \lambda^{(3)}, \ldots$

Write out each decimal expansion

$\lambda^{(1)} = 0.\lambda_1^{(1)}\lambda_2^{(1)}\lambda_3^{(1)}\ldots$

$\lambda^{(2)} = 0.\lambda_1^{(2)}\lambda_2^{(2)}\lambda_3^{(3)}\ldots$

→ How can we make a number NOT in the list? ...

when we don't even know what is in the list.

$\alpha = 0.\alpha_1\alpha_2\alpha_3\ldots$

$\alpha_1 = 1 - \lambda_1^{(1)} \qquad \alpha_i = 1 - \lambda_i^{(i)}$

$\alpha_2 = 1 - \lambda_2^{(2)}$

clearly:

$\alpha \neq \lambda^{(i)} \Rightarrow \alpha$ was not on the list.

= "DIAGONAL TRICK": By Cantor. ∎

Back to Halting Problem:

$$h(i,x) = \begin{cases} 1 & i(x) \text{ ends} \\ 0 & i(x) \text{ runs forever} \end{cases} \qquad \begin{array}{l} i = \text{program} \\ x = \text{input} \end{array}$$

If there is a program that computes $h(i,x)$ for all i and all x, then

$$g(i) \begin{cases} 0 & \text{if } h(i,i) = 0 \\ \text{undef.} & \text{if } h(i,i) = 1 \end{cases}$$

$i(i)$? underlying bit sequence is equivalent for progs & inputs. computer takes no distinction.

can be computed w/ some program.

Let "e" be a program that computes $g(i)$

Consider the question $g(e)$?

Case A

if $g(e) = 0$

$h(e,e)$ = runs forever e(e) runs forever.

But e was suppose to compute g. So it did not compute g afterall . .

CASE B

If $g(e)$ is undefined, then $h(e,e)$ will end in finite time.

But $g(e)$ should have been zero.

IDEA: Assum program halts. Cook up a paradox within the definition compute something I know cannot be computed.

upshot: The program e cannot exists because e(e) cannot have a value

$\Rightarrow$ g cannot be computed $\Rightarrow$ h cannot be computed.

Planar Graphs

K_1 •

K_2 ——

K_3 △

K_4 

K_5 :  ← must have edges crossed

Def: A graph is planar if it can be drawn in the plane without crossing edges.

Theorem (Euler)

Let G_2 be a planar graph, connected.

→ no particular image comes to mind.

(both planar & connected, not the same)

Chose any plane representation of G_2.

This will "cut" the plane in finite regions call the "faces," f

Then,

$$v + f = e + 2.$$

(# of vert, # of faces, # of edges)

Proof:

Start w/ 1 vertex;

$f=1 \quad v=1 \quad e=0$

equation is true.

From this point, draw a new edge

↙ ends at New point

$f=1 \quad v=2 \quad e=1^{+1}$

$f=f$

$v=+1$

$e=+1$

↘ goes to an old point

$e \to +1$

$v \to v$

$f=+1$ ← because the graph is connected, a new path merely adds another edge to the point. Now a "loop" has form

new face