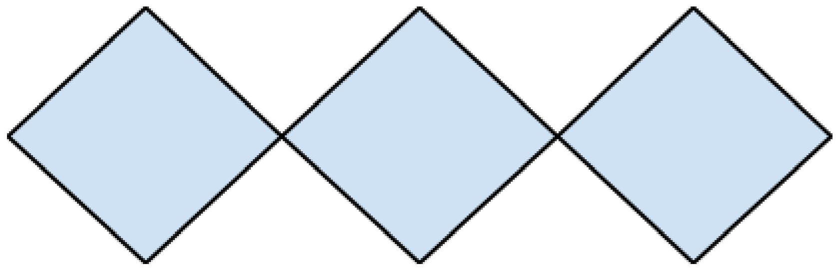
I. Exponential Algorithms

1. trying all paths through a network is an enormous amount of work



2. for the above, with 2 available paths from every node, the number of paths is 2^n
3. exponential algorithm = the time required to solve the problem is in the exponent
    1. these problems get very big very fast
    2. we want to avoid exponential algorithms

II. Polynomial

1. algorithms that require some time like n^a where a is a constant
    1. if n is the number of edges, time is roughly proportional to n^2 (a=2)
    2. we want polynomials
2. ideally, we want a to be small
3. as computational ability gets better, so do sensors
    1. puts more pressure on finding better algorithms

III. NP - non-deterministic polynomial

1. asks question: does there exist any solution that satisfies requirements
    1. deterministic: must provide solution
2. problem is NP if a solution can be *verified* in polynomial time
3. encryption problems are NP
4. Traveling Salesman
    1. given n cities and cost to go from any city x to any other city y, is there a path that visits every city at least once with total cost ≤k?
    2. finding satisfactory path is difficult
5. NP complete problems = problems where only known method to find answer takes exponential time
6. called perebor (перебор) in Russian, meaning “brute force search”

IV. Information Theory

1. invented by Claude Shannon et al at Bell Labs
2. uncertainty ~ communication requirements

P(A) = .25 00   011011001011
P(B) = .25 01   = BCDACD
P(C) = .25 10
P(D) = .25 01   average length of 2 bits per letter

P(A) = .5 (0)   1010011010111
P(B) = .25 (10)   = BBACBD
P(C) = .125 (110)   (unique decodability)
P(D) = .125 (111)

The sum of (P X Bits) = the average number of bits needed

(.5)(1)+(.25)(2)+(.125)(3)+(.125)(3)
= 1.75

3. therefore, need an average of 1.75 bits per letter
4. information is related to probability
    1. it’s easier to describe the location of a match if they are all in the box than if they are scattered across the room (i.e. it takes fewer bits)

I look at the building.
I do not look at the building.

5. if you see “do,” not will likely follow
6. if you see “look,” you do not necessarily know what will follow

A1	A2	A3	AN	Target Att.
hair color	height	temperature		Sick?

7. A3 will be best at predicting target attribute
8. knowing which attribute will decrease uncertainty of target attribute
9. information(target)-information(target/A3)
10. info(letters)= $P(A)\log_2[1/P(A)] + P(B)\log_2[1/P(B)] + P(C)\log_2[1/P(C)]$
$+ P(D)\log_2[1/P(D)]$
= 1.75 (confirmed above)
    1. this is the optimal code

Glad	no	yes	no	yes	yes	yes	no	no
Major	hist.	math	math	cs	cs	math	hist.	math
Coast	e	w	e	w	e	e	e	w

$P(no)\log_2[1/P(no)] + P(yes)\log_2[1/P(yes)] = 1$
    1. (for not knowing the major)

infogain(major for glad) = info(glad)-info(glad/major)
    1. all about reducing uncertainty

info(glad)=1
info(glad/major) = P(major=math) X info(glad/math) = .5
P(major=hist) X info(glad/hist) = 0
P(major+cs) X info(glad/cs) = 0
for major=math (yes, no, no, yes), info = 1 (same as info(glad))
therefore, P(major=math) X info(glad/math) = .5

infogain(major for glad) = 1 - .5 = .5