Puzzles and Combinatorics

You

May 8, 2024

1 Introduction

1. Riddles, Greek mythology, Sphink riddle, middle ages
2. Jigsaw puzzles: started 1760, pieces of wood, map cut into small pieces, dissected map: educational
3. Puzzles important in all industries: computer science, engineering, physics, math, chemistry etc.
4. Generalization: How do we apply toy problems to real world?

2 Pigeonhole Principle

If there are n holes and $n + 1$ pigeons, then at least one hole must contain more than one pigeon.
Proof: If there is one pigeon in each hole, then there are n pigeons, but there are actually $n + 1$ pigeons... CRAZY

2.1 Example: Hairs on head

Prove that there are at least two people in the United States that have the same number of hairs on their head.
Population of US: 333 million Numbers of hair on head: 100,000
pigeon

2.2 Example: Summing Numbers

In an ancient cave, you see a seemingly arbitrary list of 100 positive integers carved in a row. You add up all the integers and the sum is 152. Show that, regardless of what the, integers are, among them you can locate an unbroken block of adjacent integers that add up to exactly 47.

$$s_1 = a_1 \tag{1}$$
$$s_2 = a_1 + a_2 \tag{2}$$
$$\cdots \tag{3}$$
$$s_i = a_1 + a_2 + \cdots + a_i \tag{4}$$
$$s_{100} = a_1 + a_2 + \cdots + a_{100} \tag{5}$$

- $s_{100} = 152$, $1 \leq s_1 < s_2 < s_{100} = 152$
- Create another set of integers $t_i = s_i + 47$, $t_{100} = 199$, $48 \leq t_1 < t_2 < t_{100} = 199$
- Now we can put both of the lists together : $1 \leq s_1, s_2, \cdots, t_{100} < t_1, t_2, \cdots, t_{100} \leq 199$.
- Pigeon: We have 200 integers, but each oen is 1 or bigger or 199 and smaller, so 199 possibilities but 200 numbers $\rightarrow$ at least an s and a t must be the same.

$$s_j = t_i = s_i + 47, \text{where } 1 \leq i < j \leq 100 \tag{6}$$

So then

$$s_j - s_i = 47 \tag{7}$$
$$(a_1 + \cdots + s_j) - (a_1 + \cdots + s_i) = 47 \tag{8}$$
$$a_{i+1} + a_{i+2} + \cdots + a_j = 47 \tag{9}$$

REFORMULATION: Let P and H be finite sets, and let $f : P \to H$ be a function. If $|P| > |H|$, then f is not 1-1.

3 Pizza cutter problem

What is the largest number of pieces of pizza we can make with n straight cuts through a circular pizza?

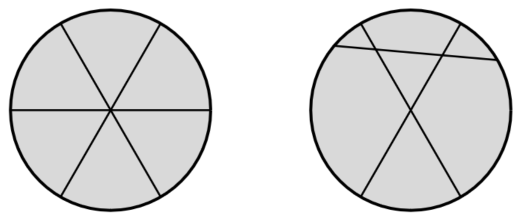


Figure 1.1: Three cuts through a pizza

Figure 1: Caption

The goal is to maximize the number of pieces regardless of size and shape.

3.1 Solutions

3.1.1 Recursion

$$P(0) = 1 \tag{10}$$
$$P(1) = 2 \tag{11}$$
$$P(2) = 4 \tag{12}$$
$$P(3) = 7 \tag{13}$$
$$P(4) = 11 \tag{14}$$
$$P(5) = 16 \tag{15}$$
$$P(6) = 22 \tag{16}$$
$$P(7) = 29 \tag{17}$$
$$\tag{18}$$

Maximizing principle: The number of pizza pieces is maximized when every cut crosses every other cut, but no three cuts cross at the same point.
Differences between values: $1, 2, 3, 4, 5, 6, 7$
Formula for cutting the pizza:

$$P(n) = P(n-1) + n \tag{19}$$

Schematic

1. Start with n-1 cuts that form P(n-1) pieces of pizza
2. Make the nth cut (the dotted line in Figure 1.4) to form P(n) pieces
3. When the new cut meets one of the n-1 previous ones, a pizza piece is cut in two.

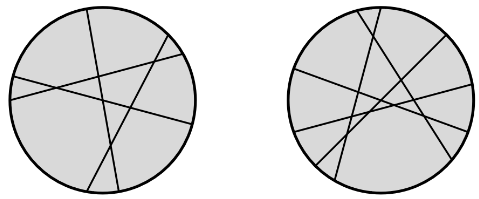


Figure 1.2: Four and five cuts through a pizza

Figure 2: Caption

4. A piece is cut in two when the new cut finishes on the opposite side of the pizza
5. So the total number of pieces of pizza increases by n when we pass from n-1 cuts to n cuts

Trying to get a solution:

$$P(n) = P(n-1) + n \tag{20}$$
$$P(n-1) = P(n-2) + (n-1) \tag{21}$$
$$P(2) = P(1) + 2 \tag{22}$$
$$P(1) = P(0) + 1 \tag{23}$$

Add equation and cancel common terms

$$P(n) = P(0) + 1 + 2 + \cdots + (n-2) + (n-1) + n. \tag{24}$$

Recall

$$1 + 2 + 3 + \cdots + n = n(n+1)/2 \tag{25}$$

so then

$$P(n) = P(0) + n(n+1)/2 = 1 + (n(n+1))/2 = (n^2 + n + 2)/2 \tag{26}$$

3.1.2 Graph Theory Solution

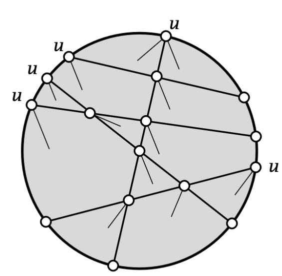


Figure 1.5: The top vertex of each pizza piece is either an interior vertex or an upper boundary vertex (u)

Figure 3: Caption

- upper boundary: highest vertex for each piece, one upper boundary is formed for each cut
- each piece matched to one upper boundary vertex
- highest upper boundary vertex has two vertices
- every interior vertex is matched with one piece.

$$\text{max \# of pieces} = 1 + \text{\#cuts} + \text{\# of interior vertices} \tag{27}$$

- To maximize the number of pizza pieces, each pair of cuts must cross at a unique interior vertex

$$\text{max \# of pieces} = 1 + \text{\#cuts} + \text{\# of pair cuts} \tag{28}$$

New formula: $\binom{n}{k}$, the number of ways to choose a subset of k elements from a set of n elements.

$$P(n) = \binom{n}{0} + \binom{n}{1} + \binom{n}{2} \tag{29}$$

1. $\binom{n}{0}$: 1 way to choose 0 cuts (namely, do nothing)
2. $\binom{n}{1}$: n ways to choose 1 cut
3. $\binom{n}{2}$: n chooses for first cut, $n-1$ ways for second cut, divide by two since you can pick either order $\rightarrow \frac{n(n-1)}{2}$

$$P(n) = \binom{n}{0} + \binom{n}{1} + \binom{n}{2} = \frac{n^2+n+2}{2} \tag{30}$$

Can also generalize to 3-D
Generalize: Steiner's plane-cutting problem. What is the maximum number of regions formed by n lines in the plane?

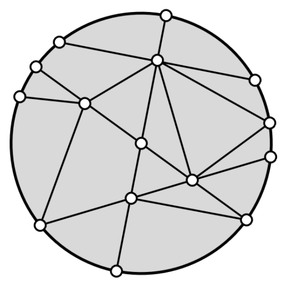


Figure 1.8: Vertices, edges, and pizza pieces

Figure 4: Caption

Euler's formula for plane graphs. If a connected plane graph has v vertices, e edges, and f faces, then

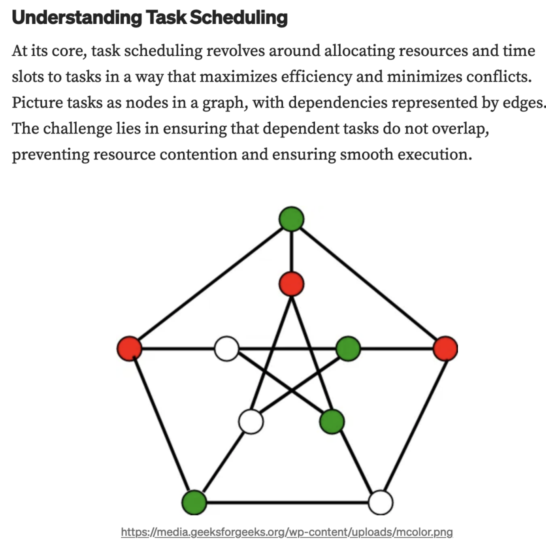
$$v - e + f = 2 \tag{31}$$

3.2 Graph Theory

1. graph connectivity: social media
2. weighted graph: social media again
3. graph coloring: scheduling problems: The textbook approach to this problem is to model it as a graph coloring problem. The compiler constructs an interference graph, where vertices are variables and an edge connects two vertices if they are needed at the same time. If the graph can be colored with k colors then any set of variables needed at the same time can be stored in at most k registers.

4 Hamiltonian cyclic problem

: Hamiltonian cyclic problem, if there is a path that visits each vertex exactly once.
- There are n! different sequences of vertices that might be Hamiltonian paths in a given n-vertex graph



Understanding Task Scheduling

At its core, task scheduling revolves around allocating resources and time slots to tasks in a way that maximizes efficiency and minimizes conflicts. Picture tasks as nodes in a graph, with dependencies represented by edges. The challenge lies in ensuring that dependent tasks do not overlap, preventing resource contention and ensuring smooth execution.

https://media.geeksforgeeks.org/wp-content/uploads/mcolor.png

Figure 5: Caption

4.1 NP Completeness

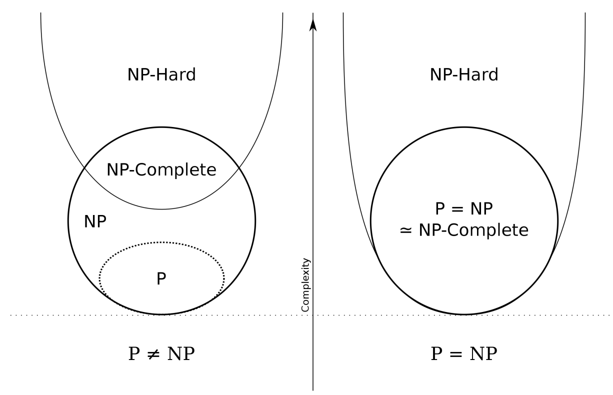


Figure 6: Caption

1. P: Problems that can be solved in polynomial time by a computer
2. NP: verified in polynomial time, not solvable
3. NP complete: problem x is in NP complete if and only if every other problem in NP can be quickly (ie. in polynomial time) transformed into x
4. if any one of the NP-Complete problems was to be solved quickly, then all NP problems can be solved quickly.

Hamiltonian Cycle

1. input a graph G, starting vertex s, and ending vertex t
2. potential solution c, c would consist of a string of vertices where the first vertex is the start of the proposed path and the last is the end
3. all vertices G appear once in c
4. first vertex in c is equal to s, last vertex is equal to t

5. every edge between vertices in c is an edge in G

Quantum annealing (QA) is an optimization process for finding the global minimum of a given objective function over a given set of candidate solutions (candidate states), by a process using quantum fluctuations