**Problem:**

Playing multiple games at once.

At each turn, each player chooses a game and makes a move.

You lose if there is no possible move.

**Solution:**

Grundy Numbers (Nimbers)

For each game, we compute its Grundy number

The first player wins iff the XOR of all grundy numbers is nonzero

Computing the grundy numbers:

Let S be a state, and be states reachable from S using a single move.

The Grundy number of a losing state is 0

The Grundy number g(S) of S is the smallest nonnegative integer that does not appear in

**Problem:**

Game Fifteen

Given a 4x4 grid of unique numbers between 1 and 15 and one empty cell.

The position of a number can be exchanged with the position of the empty cell if it is adjacent.

Is it possible to arrange the numbers to this permutation?

**Solution:**

Let N be the number of inversions in the permutation.

Let K be the line number (starting at zero) of the empty cell.

A solution exists iff N+K is even

**Problem:**

Build the set of all non-negative fractions.

**Solution:**

Start with the fractions:

(0/1, 1/0)

For every pair of adjacent fractions, create a new fraction between them where the numerator is the sum of their numerators and the denominator is the sum of their denominators. Repeat infinitely.

**Problem:**

Count the number of spanning trees in a graph G

**Solution:**

Kirchoff matrix theorem

Let D be the degree matrix of G

Let A be the adjaceny matrix of G

Let Q = D - A

Let Q' be the matrix resulting from deleting any row and any column from Q

The number of spanning trees is equal to the determinant of Q'

**Problem:**

Given an undirected, unweighed Graph G, find the number of paths of length k between every pair of vertices.

**Solution:**

D\_k = G^k

**Problem**

Given an undirected weighted graph of m edges , n vertices, and a vector P of weights and a source vertex S, find new values for the weights of all edges such that the P[i] is now the length of the shortest path from S to vertex i.

**Solution**

Linear, keep a vector cost\_ch of changes to each edge, a vector of nodes decrease\_id (stores the neighbors that must be decreased for each node), of and a vector of decreases decrease (the smallest decrease that must be made to any neighbor for each vertex).

const int INF = 1000\*1000\*1000;

int n, m;

vector<int> p (n);

bool ok = true;

vector<int> cost (m), cost\_ch (m), decrease (n, INF), decrease\_id (n, -1);

decrease[0] = 0;

for (int i=0; i<m; ++i) {

int a, b, c; // текущее ребро (a,b) с ценой c

cost[i] = c;

for (int j=0; j<=1; ++j) {

int diff = p[b] - p[a] - c;

if (diff > 0) {

ok &= cost\_ch[i] == 0 || cost\_ch[i] == diff;

cost\_ch[i] = diff;

decrease[b] = 0;

}

else

if (-diff <= c && -diff < decrease[b]) {

decrease[b] = -diff;

decrease\_id[b] = i;

}

swap (a, b);

}

}

for (int i=0; i<n; ++i) {

ok &= decrease[i] != INF;

int r\_id = decrease\_id[i];

if (r\_id != -1) {

ok &= cost\_ch[r\_id] == 0 || cost\_ch[r\_id] == -decrease[i];

cost\_ch[r\_id] = -decrease[i];

}

}

cost\_ch now holds the changes to each edge (increase or decrease) with minimum sum of absolute values, if ok is true