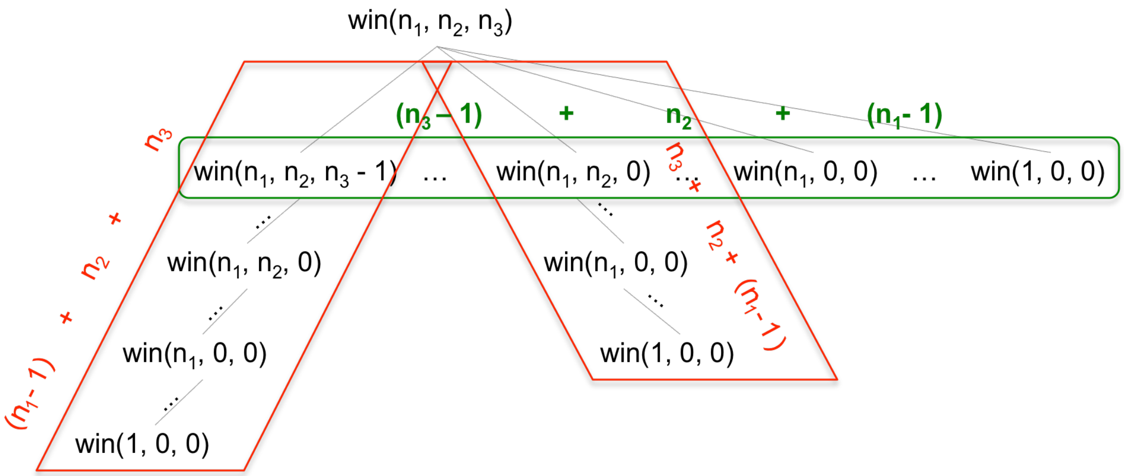
Assignment one: NIM

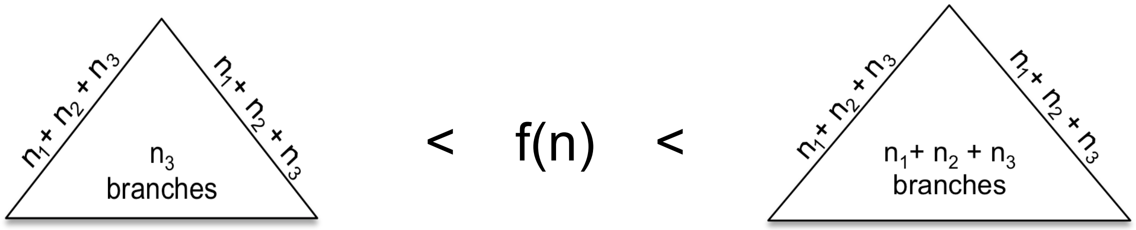
A01772483
Laleh Rostami Hosoori
03/09/2012

Step 1) Recursive Algorithm:

```
Bool winRec (n1, n2, n3) :
    Bool  flag = false;

    if (n1 + n2 + n3 == 1)     return false;

    for (i = 0; i <= n1 && !flag; i++)
      if (i == 0)
        for (j = 0; j <= n2 && !flag; j++)
          if (j == 0)
            for (k = 1; k <= n3 && !flag; k++)
              flag = !winRec (n1, n2, n3 - k);
          else
            flag = !winRec (n1, n2 - j, n3);
      else
        flag = !winRec (n1 - i, n2, n3);

    if (flag)    return true;
    else    return false;
```

Step 2) Algorithm Complexity in worse Case:





$$n_3^{(n_1+n_2+n_3)} \qquad O(n^n) \qquad (n_1+n_2+n_3)^{(n_1+n_2+n_3)}$$

Step 3,4) Cache Algorithm:

In **NimGame folder** (Main.java, Nim.java)

Step 5) The logic to play the game:

At first, choose a pile then take as much as possible from it so that if the tuple (m_1, m_2, m_3) represents the remained pieces of piles 1, 2 and 3 respectively, the result of XOR on the elements of the tuple equals zero:

m_1 ^ m_2 ^ m_3 = 0

In this situation the opponent will lose the game. For example, in a game with 3 piles of 12, 10, 15 pieces, you will win by removing 11 pieces from the third pile because of (12 ^ 10 ^ 6) = 0. There are a lot of other examples exist in this case, such as (11, 10, 1), (6, 6, 0) and (1, 3, 2).