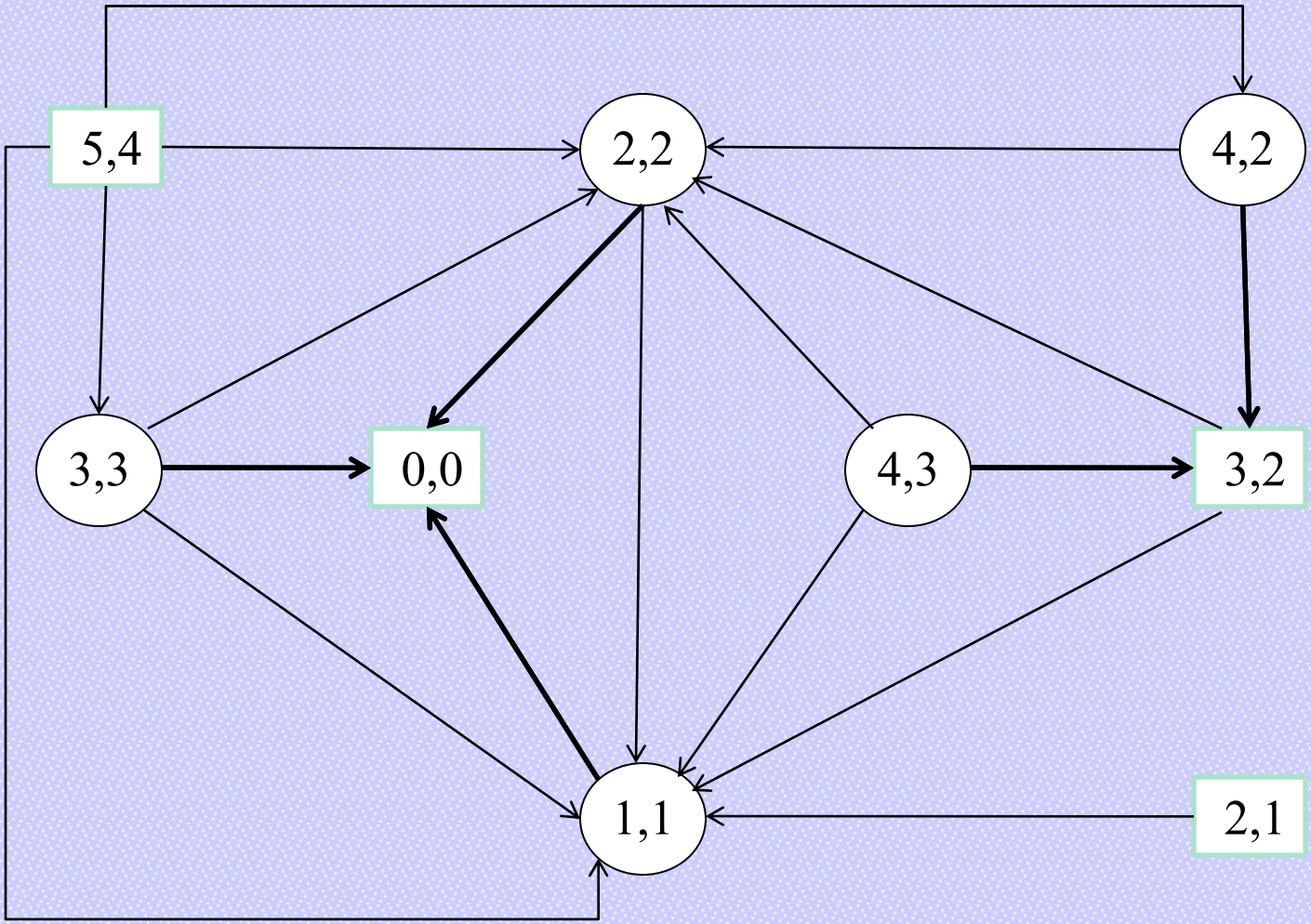
EXPLORANDO GRAFOS

1. Grafos e Jogos – uma introdução

Considere o seguinte jogo, o qual é uma das muitas variantes de Nim, também conhecido como jogo Marienbad. Inicialmente existe uma certa quantidade de fósforos na mesa entre dois jogadores. O primeiro jogador pode remover quantos fósforos ele quiser, exceto que ele deve pegar pelo menos um e deixar pelo menos um. Então deve existir pelo menos dois fósforos no inicio do jogo. Depois, cada jogador na sua vez precisa remover pelo menos um fósforo e no máximo o dobro do numero de fósforos que seu oponente acabou de pegar. O jogador que remove o último fósforo ganha. Não há empates.

EXPLORANDO GRAFOS

1. Grafos e Jogos – uma introdução



<x, y>: x é numero de fósforos e y é numero máximo de fósforos que um jogador pode remover.

EXPLORANDO GRAFOS

1. Grafos e Jogos – uma introdução

O seguinte algoritmo determina se uma determinada posição é vencedora ou perdedora:

```
function recwin(i, j)
    {return true if and only if <i, j> is winning;
     we assume 0 ≤ j ≤ i}
    for k ← 1 to j do
        if not recwin(i-k, min(2k, i-k))
            then return true
    return false
```

1. Uma posição não terminal é uma posição vencedora se *pelo menos uma* de suas sucessoras é uma posição perdedora, de modo que o jogador da vez poderá deixar seu adversário nessa posição perdedora.
2. Uma posição não terminal é uma posição perdedora se *todas* as suas sucessoras são posições vencedoras, de modo que o jogador da vez não pode evitar deixar seu oponentes em uma dessas posições vencedoras.

EXPLORANDO GRAFOS

1. Grafos e Jogos – uma introdução

O algoritmo anterior tem o problema de calcular os mesmos valores muitas vezes. Existem duas maneiras de resolver esse problema, a primeira usa programação dinâmica, requer um array booleano *G* tal que G[*i*,*j*] = **true** se e somente se (*i*,*j*) é uma posição vencedora.

```
function dynwin(n)
    {Para cada 1 ≤ j ≤ i ≤ n, configura G[i,j] para true se e somente
      se (i,j) é posição vencedora}
    G[0,0] ← false
    for  i ← 1 to n do
          for j ← 1 to i do
                k ←1
                while k < j and G[i-k, min(2k, i – k)] do
                      k ←k + 1
                G[i,j] ← not G[i – k, min(2k, i – k)]
```

EXPLORANDO GRAFOS

1. Grafos e Jogos – uma introdução

O problema do algoritmo anterior é que ele calcula algumas entradas de *G* que não serão usadas. Isto não aconteceria com a primeira versão do algoritmo. A solução abaixo combina as vantagens de ambos os algoritmos anteriores usando uma função de memória, que lembra quais nós foram visitados. As seguintes inicializações são necessárias:

```
G[0,0] ← false; known[0,0] ← true
        for i ← 1 to n do
                for j ← 1 to i do
                        known[i,j] ← false
```

EXPLORANDO GRAFOS

1. Grafos e Jogos – uma introdução

```
function mim(i,j)
        {Para cada 1 ≤ j ≤ i ≤ n, retorna true se e somente se (i,j) é
        posição vencedora}
        if known[i,j] then return G[i,j]
        known[i,j] ← true
        for k ←1 to j do
                if not nim(i – k, min(2k, i – k)) then
                        G[i,j] ← true
                        return true
                G[i,j]←false
                return false
```

EXPLORANDO GRAFOS

1. Grafos e Jogos – uma introdução

Para determinar uma estratégia vencedora em um jogo deste tipo, atribuímos a cada nó do grafo um rótulo escolhido dos conjuntos *vitória*, *derrota* e *empate*. Os rótulos se referem à situação de um jogador que está para se mover para a posição correspondente, assumindo que nenhum jogador irá cometer um erro. Os rótulos são adicionados sistematicamente como segue:

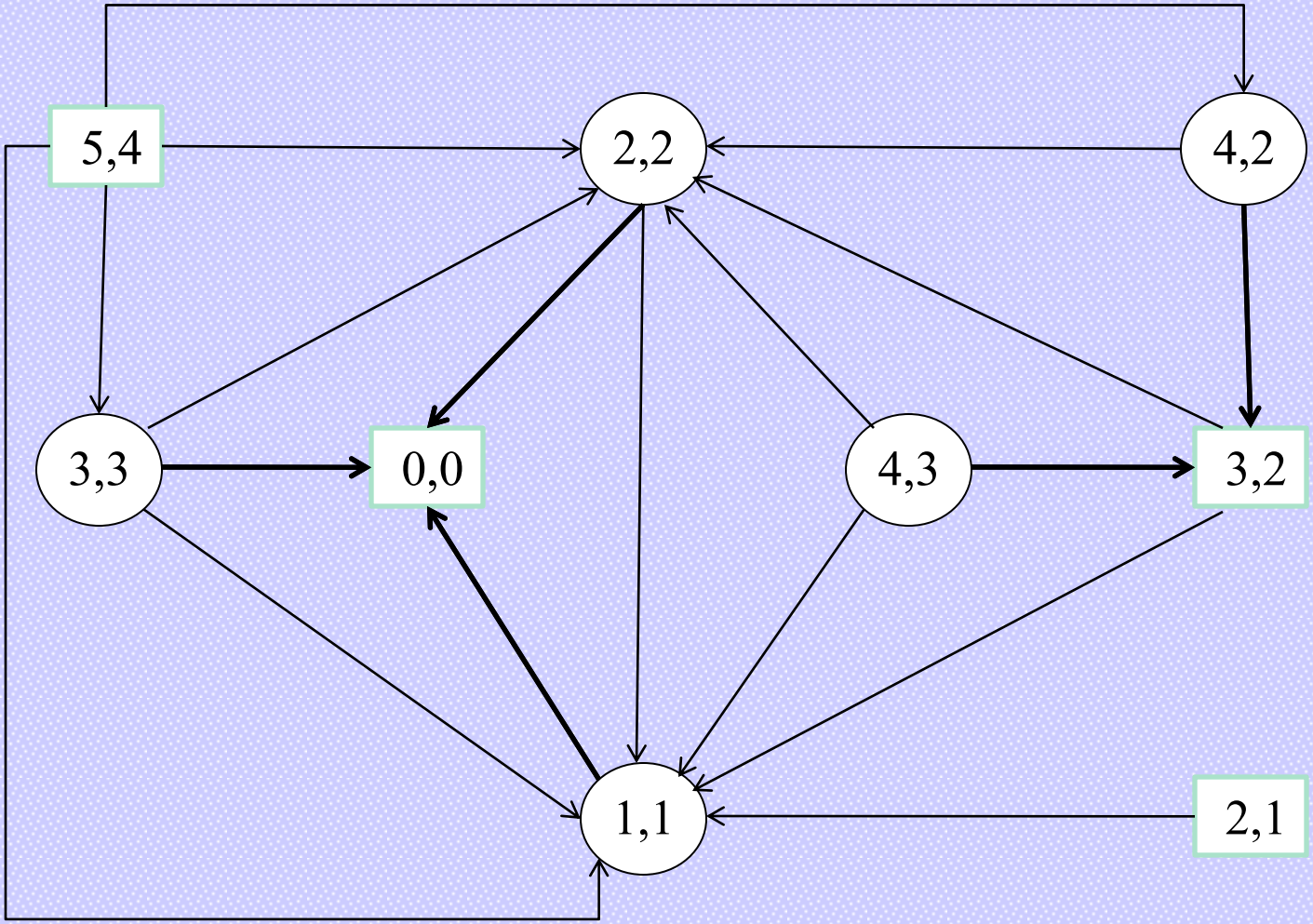
EXPLORANDO GRAFOS

1. Grafos e Jogos – uma introdução

1. Rotule as posições terminais.
2. Uma posição não terminal é uma posição vencedora se *pelo menos uma* de suas sucessoras é uma posição perdedora, de modo que o jogador da vez poderá deixar seu adversário nessa posição perdedora.
3. Uma posição não terminal é uma posição perdedora se *todas* as suas sucessoras são posições vencedoras, de modo que o jogador da vez não pode evitar deixar seu oponentes em uma dessas posições vencedoras.
4. Qualquer outra posição não-terminal leva a um empate. Neste caso as sucessoras precisam incluir pelo menos um empate, possivelmente com algumas posições vencedoras também. O jogador da vez pode evitar deixar o oponente em posições vencedoras, mas não pode forçá-lo a assumir uma posição perdedora.

EXPLORANDO GRAFOS

Exercício: Adiciona nós <8, 7>, <7, 6> <6, 5> e seus descendentes ao grafo do último exemplo.



EXPLORANDO GRAFOS

Exercício: Modifique o algoritmo *recwin* que retorna um numero inteiro k, no qual $k = 0$ se a posição é uma posição perdedora, e $1 \leq k \leq j$ se é um movimento vencedor pegar k fósforos.