Programação I

Jogo Nim- Variante de Moore

Relatório

Ricardo

Trabalho realizado por:
- Miguel Casco nº 28966
- Ricardo Fusco nº 29263
- Nuno Ramalho nº 29578

Engenharia Informática

Relatório

O trabalho a ser feito este ano no âmbito da cadeira de Programação I trata-se de um jogo matemático chamado Nim, mais concretamente a variante de Moore. O mesmo encontra-se dividido em 3 fases, o modo humano vs humano: texto, o modo humano vs humano: texto e o modo gráfico do humano vs humano que irão ser explicadas de seguida. Começámos por importar os módulos random e tkinter, e inicializar algumas variáveis e listas importantes para a execução do código tais como:

- a variável win que é a variável utilizada um pouco mais à frente no loop while principal do jogo. Enquanto esta for 0 significa que o jogo ainda não acabou, apenas lhe é atribuído o valor 1 quando se verificar que houve game over por meio da função gameover(t, p, k).

- a lista t que que vai conter o tabuleiro aleatório utilizando o random.sample(range(a, b), c) que vai devolver uma lista de c elementos cada um deles entre a e b. No caso do trabalho devolve logo uma lista de 4 elementos, cada um deles entre 1 e 30.

- a lista jog que é a lista que vai conter a possível jogada.

- o root para a posterior inicialização do tkinter na parte gráfica do jogo.

- a string j, na qual irá ser utilizado o split para a converter numa lista em que cada numero é uma string.

- a variável k que vai ser usada para determinar qual o modo de jogo que está a ser usado.

- a variável c que serve para o menu e contem a escolha do modo.

A primeira função definida é a função novotab(jog, t) que basicamente remove ao tabuleiro os números presentes na jogada, e devolve o novo tabuleiro resultante.

```
#########################################################
# Funcao que faz as mudancas da jogada no tabuleiro #
#########################################################

def novotab(jog, t):
    # apos verificadas as condicoes anteriores é removido o numero
    # de objectos da respectiva pilha
    # (novotab = [t[0]-jog[0], t[1]-jog[1], t[2]-jog[2], t[3]-jog[3]])
    x = 0
    for n in jog:
        t[x] -= jog[x]
        x += 1
    return t
```

De seguida temos a função que verifica se foram cumpridas as condições para terminar o jogo (gameover(t, p, k)), ou seja, verifica se todos os elementos são 0. Caso se verifique que são é chamada a função mostrartab(t, k), que serve para fazer um print do estado actual do tabuleiro, é feito outro print da mensagem a indicar que o jogador ganhou e é chamada a função novojogo(win, p) que irá perguntar se o jogador deseja jogar outra vez, se sim, é feito um "reset" na lista jog e é chamada a função do menu inicial para escolher o modo que desejar jogar. Caso não deseje jogar de novo é executado o comando para terminar o programa.

A primeira função a ser executada é a função menu() que irá mostrar o menu inicial e pedir ao utilizador qual a escolha que quer. Se seleccionar o modo humano vs humano: texto é pedido ao utilizador qual o jogador que joga primeiro, e é executada a função game(win, p, t, jog, j, k) que é onde se encontra o loop principal do modo humano vs humano: texto. Aqui irá continuar num loop while até que se verifiquem as condições para sair com a função de game over que já foi referida anteriormente. Dentro do loop, consoante o jogador que estiver a jogar, irá ser mostrado o tabuleiro, irá ser executada a função jogada(jog, p) que vai pedir a jogada ao jogador ao qual este deverá colocar o número a retirar da pilha seguido de um ou mais espaços, sendo esta guardada na lista jog após feito o split e a conversão dos números de string para inteiros. De seguida será executada a função inc(t, jog), cuja utilidade é usar uma variável acumuladora para ser incrementada por cada valor encontrado na jogada maior que 0, sendo esta variável y devolvida para ser utilizada na próxima função, limjog(p, jog, t, y, k). Esta função tem como objectivo verificar se a jogada efectudada pelo utilizador cumpre as regras do jogo. Ou seja, é executada a função joginv(p,jog,t,k) quando se verificar que há mais do que dois números maiores que zero na jogada, sempre que algum numero na jogada for maior que o que está no tabuleiro ou se a jogada for do tipo 0 0 0 0, caso contrário

```
#############################################################
# Funcao principal/geral que contem o loop principal do jogo #
#############################################################

def game(win, p, t, jog, j, k):
    """loop principal do jogo: ira continuar neste loop até se verificar
    que o jogo terminou"""
    k = 'HvsH'
    while win == 0:
        if p == 1:
            mostrartab(t, k)

            jog = jogada(jog, p)

            y = inc(t, jog)

            limjog(p, jog, t, y, k)

            # ao atribuir o valor 2 a variavel p a funcao "salta" para
            # a parte do jogador 2
            p = 2
        else:
            mostrartab(t, k)

            jog = jogada(jog, p)

            y = inc(t, jog)

            limjog(p, jog, t, y, k)

            # ao atribuir o valor 1 a variavel c a funcao "salta" para
            # a parte do jogador 1
            p = 1
```

é executada a função novotab(jog, t) referida anteriormente e é chamada a função gameover para verificar se o jogo está em condições de ser terminado. Por último é atribuída à variável p o valor correspondente ao próximo jogador "saltando" para a respectiva parte.

Se no menu seleccionarmos o modo humano vs comp: texto é chamada a função menu2() que mostra um segundo menu com o intuito de determinar quem joga primeiro se o comp ou o jogador e depois é chamada a função hvsc(win, p, t, jog, j, k).

```
########################################################################
# Funcao que mostra um segundo menu e que pergunta quem joga primeiro #
########################################################################

def menu2():
    print ('Quem joga primeiro? \n')
    print ('1 - Humano    2 - Computador \n')
    p = int(input('>_ '))
    jog = [0,0,0,0]
    t = random.sample(range(1,30),4)
    hvsc(win, p, t, jog, j, k)
```

A função principal do modo humano vs comp: texto(hvsc()) é bastante similar à função game, sendo exactamente igual na parte do jogador e com umas diferenças na parte do computador. Na parte do computador após ser mostrado o tabuleiro é executada uma função chamada comp(t,jog,p,k) que é a função que contém o algoritmo responsável pela "inteligência" do computador. O problema com que nos deparamos foi dar ao computador alguma "inteligência", de modo a que ele não jogasse apenas aleatoriamente. Fizemos alguns jogos entre nós e apercebemo-nos de que se houver três **1**'s e um 0 no tabuleiro, o jogo estava ganho. Fomos ai buscar o algoritmo para o computador não jogar apenas aleatoriamente.

```
########################################################################
# Funcao principal/geral que contem o loop principal do modo hum. vs comp. #
########################################################################

def hvsc(win, p, t, jog, j, k):
    """loop principal do jogo: ira continuar neste loop até se verificar
    que o jogo terminou"""
    k = 'HvsC'
    while win == 0:
        if p == 1:
            mostrartab(t, k)

            jog = jogada(jog, p)

            y = inc(t, jog)

            limjog(p, jog, t, y, k)

            jog = [0, 0, 0, 0]

            # ao atribuir o valor computador a variavel p a funcao "salta" para
            # a parte do computador
            p = 'Computador'
        else:
            mostrartab(t, k)

            comp(t, jog, p, k)

            print('Computador:', jog[0], jog[1], jog[2], jog[3])

            gameover(t, p, k)

            # ao atribuir o valor 1 a variavel p a funcao "salta" para
            # a parte do jogador
            p = 1
```

O computador vai verificar o tabuleiro, caso encontre um tabuleiro que tenha apenas um 1 ou então dois 1 e o resto tudo 0, do tipo '0 0 0 1' ou por exemplo '0 1 1 0', o computador retira os 1 e ganha.

Caso o computador encontre um tabuleiro que tenha um 0 e um 1, sendo os restantes dois diferentes de 0, tipo '1 0 4 21', ele vai retirar aos dois números de modo a aparecerem uns, ficando um tabuleiro '1 0 1 1', já tendo ai a vitória assegurada.

Pode acontecer o tabuleiro ter também três 1 e um número diferente, do tipo '1 1 12 1', neste caso ele faz com que o número diferente de um seja reduzido a 0, colocando o tabuleiro tipo '1 1 0 1'.

Se aparecerem no tabuleiro quatro 1, ele elimina um dos 1, ficando um tabuleiro de três I, do tipo '0 1 1 1'.

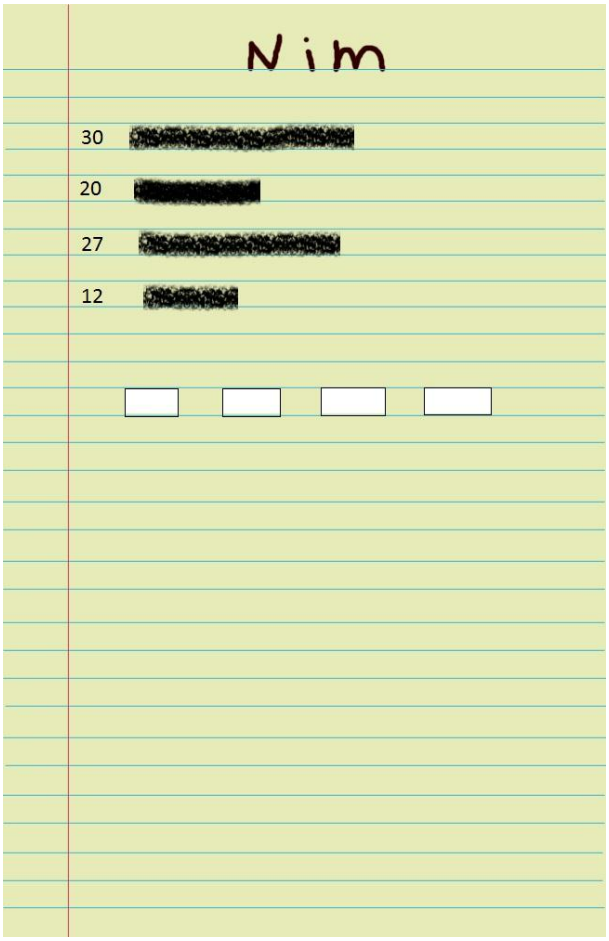
Se existir no tabuleiro dois 1 e dois números diferente de um, do género '1 6 4 1', o computador vão fazer com que um deles fique a 0 e o outro fique a 1, assegurando o tabuleiro tipo '1 0 1 1', que lhe garante a vitória.

No caso em que temos um 1 e 3 números maiores que 1 o computador colocará um desses 3 números a 2 ficando um tabuleiro do tipo 1 2 6 8. Se tivermos um 1 e dois 2 o restante irá ser colocado também a 2 ficando um tabuleiro do tipo I 2 2 2 deixando poucas hipóteses de movimento ao jogador mas também dando-lhe uma hipótese de ganhar.

Enquanto o computador não encontrar um 0 ou um 1 que lhe permita por em prática o algoritmo, ele vai jogar aleatoriamente. Caso aconteça o caso de o computador se deparar com um tabuleiro com três 1, não há nada que ele possa fazer, pois a derrota é certa, por isso ele joga aleatoriamente, fazendo desaparecer um ou dois 1, ficando um tabuleiro tipo '1 0 0 1' ou então do género ' 0 1 0 0'. Depois de feita a jogada do computador é novamente chamada a função gameover e passa ao jogador se o jogo não terminar. Esta parte do humano vs comp serve-se praticamente das mesmas funções que o modo humano vs humano.

Na parte gráfica do jogo tivemos algumas dificuldades, pois esta matéria, não foi dada na cadeira de Programação I e tivemos que ser nós a aprender por nós próprios, embora ainda se tenha falado por alto numa das aulas.

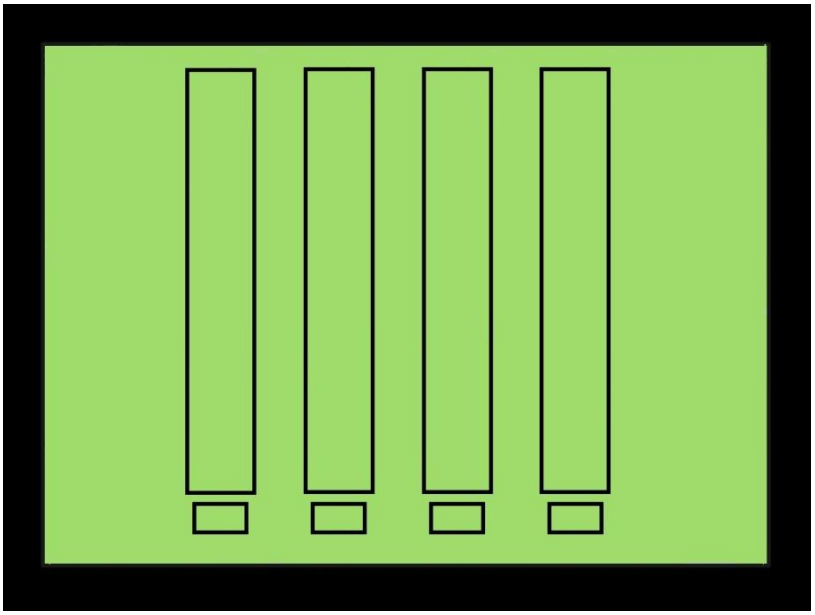
Os professores recomendaram o site tutorialspoint, em que tem lá uma explicação do módulo tkinter, e como utiliza-lo para desenvolver a nossa parte gráfica.



Primeiro que tudo tivemos que pensar em como iríamos apresentar o jogo graficamente, primeiro pensamos em fazer algo do género:

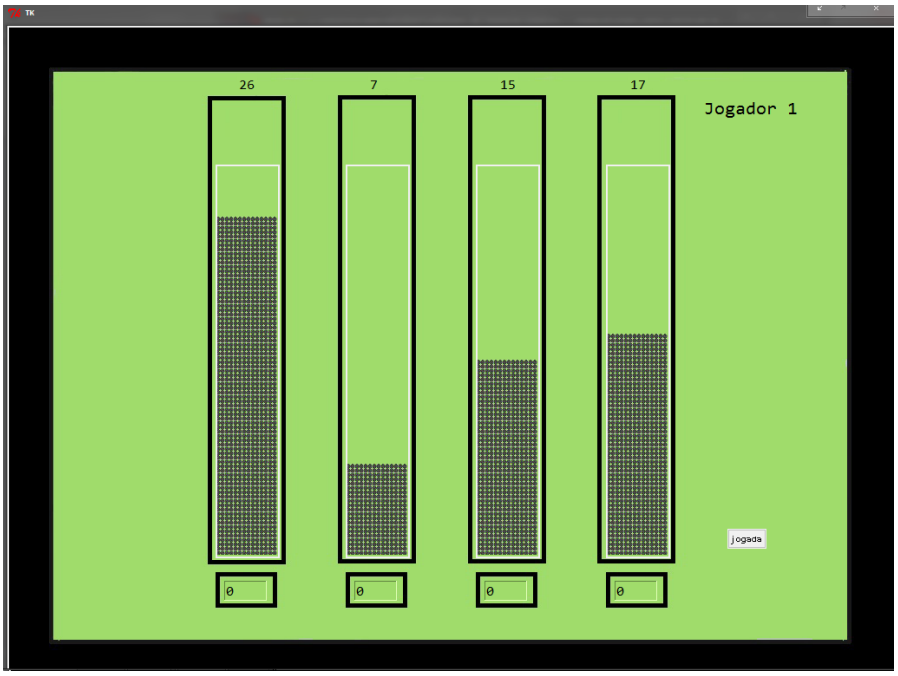
Mas acabamos por decidir que não era bem isto que queríamos para representar o nosso trabalho graficamente.

Por fim decidimos fazer uma gráfica, simples de perceber, e de se fazer, e acabamos por desenhar o seguinte fundo para a nossa interface,



Em que a quantidade de elementos das colunas iria aparecer no topo por extenso, e iria aparecer dentro das colunas um tipo de preenchimento.

Começamos por definir a função para a janela, a qual definimos com a resolução de 1024x800, e depois criamos várias widgets, tais como 5 canvas, em que 4 deles era para as colunas, para posteriormente adicionarmos conteúdo personalizado dentro das colunas como o seu valor, pois era mais fácil desta maneira de actualizar as colunas com o seu valor depois da jogada, e o outro foi utilizado para o fundo da interface.



Definimos 4 Entries, para obter o valor das jogadas, os quais estão personalizados com a mesma cor do background, como a cor tinha que ser em hexadecimal procurei um site que me dava o valor hexadecimal da cor que eu queria. (Site)

Construímos também 5 labels, 4 deles serviram para mostrar o valor das colunas, o outro para mostrar o jogador que estava a fazer a jogada, fizemos o mesmo que as Entries e pusemos as labels na mesma cor do background.

Definimos uma função jogadag, para ser o command do botão, pois este botão é o que vai executar tudo, vai tirar os valores da jogada, vai actualizar o tabuleiro, o jogador que está actualmente a fazer a sua jogada, e verificar o fim do jogo.

Criamos uma imagem para preencher as colunas com o seu valor.

Tivemos que criar condições de maneira a que as barras ficassem juntas umas às outras sem espaços, então vimos quantos pixéis tinha o nosso canvas e as nossas barras, e de seguida fizemos cálculos e fizemos as condições para cada coluna.

A parte gráfica foi toda definida dentro de uma função start(t,p) com o argumento t que é o tabuleiro gerado aleatoriamente, e com o argumento p que é para sabermos qual o jogador que começa a jogar primeiro.

Aproveitamos algumas funções definidas anteriormente, pois também serviam para a parte gráfica sem interferir com esta mesma.

Há apenas um pequeno glitch no programa se for executado clicando no exe ele abre uma janela do tkinter mas da para executar bem o resto do código, e até se for executado através do IDLE esse problema não acontece.