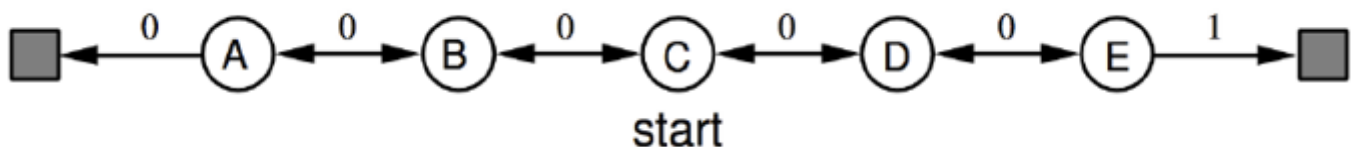


# Random walk

## Opis problemu

Poniżej przedstawiona jest implementacja algorytmu **TD(0)** do problemu **random walk**.

Diagram:



Przypomnijmy podstawowe fakty:

- **Kółkami** oznaczone są **stany nieterminalne**, **kwadratami** stany **terminalne**.
  - Wszystkie **epizody rozpoczynają** się w środkowym stanie **C**.
  - W dowolnym stanie nieterminalnym **prawdopodobieństwa ruchu w lewo i ruchu w prawo** są
  - Nad strzałkami widoczne są **wartości nagród**. Tylko przejście ze stanu **E** do **stanu terminaln**
- Poza tym wszystkie nagrody wynoszą **0**.

## Algorytm

Implementujemy algorytm:

## Tabular TD(0) for estimating $v_\pi$

Input: the policy  $\pi$  to be evaluated

Algorithm parameter: step size  $\alpha \in (0, 1]$

Initialize  $V(s)$ , for all  $s \in \mathcal{S}^+$ , arbitrarily except that  $V(\text{terminal})$

Loop for each episode:

    Initialize  $S$

    Loop for each step of episode:

$A \leftarrow$  action given by  $\pi$  for  $S$

        Take action  $A$ , observe  $R, S'$

$V(S) \leftarrow V(S) + \alpha [R + \gamma V(S') - V(S)]$

$S \leftarrow S'$

    until  $S$  is terminal

## ▼ Implementacja

Zacznijmy od importu potrzebnej biblioteki:

```
import numpy as np
```

**Stany** oznaczamy cyframi począwszy od lewej strony:

- Stan terminalny z lewej strony - **0**
- Stan A - **1**
- Stan B - **2**
- Stan C - **3**
- Stan D - **4**
- Stan E - **5**
- Stan terminalny z prawej strony - **6**

Wartości początkowe **V** dla wszystkich stanów:

```
V = np.zeros(7)
print(V)
```



```
epochs = 1000 #liczb epok czyli to ile epizodów uwzględnimy w wyliczeniu V (im więcej tym
alpha = 0.9
```

```
gamma = 0.9 #kocha zduktowania
```

```
gamma = 0.2 #bez zdyyskontowania
```

```
for i in range(epochs):
    state = 3 #stan początkowy w każdym epizodzie
    while True:
        #losowanie 0 lub 1: 0 ruch w lewo, 1 ruch w prawo
        if np.random.randint(2) == 0:
            next_state = state - 1
        else:
            next_state = state + 1

        #nagroda wynosi 1 przy przejściu do stanu 6, poza tym nagroda wynosi 0
        if next_state == 6:
            R = 1
        else:
            R = 0

        #modyfikacja wartość V zgodna z TD
        V[state] = V[state] + alpha * (R + gamma*V[next_state] - V[state])

        state = next_state

    #Jeżeli dotarliśmy do stanu terminalnego - koniec epizodu
    if state == 6 or state == 0:
        break
```

Wypiszmy wyliczone wartości **V**:

```
print(V)
```



Wartości **teoretyczne** podane w wykładzie (**slajd 19**):

```
print([0,1/6,2/6,3/6,4/6,5/6,0])
```



## ▼ Polecenie

Przetestuj działanie powyższego algorytmu dla **3 wybranych par wartości parametrów alpha i gamma** uzyskane rezultaty.

DO UZUPEŁNIENIA

alpha=0.5 gamma=0.9

V=[0. 0.00592163 0.0771827 0.14550578 0.27124381 0.43476747

0. ]

alpha=0.3 gamma=0.8

V=[0. 0.01626159 0.03699579 0.16040105 0.30423889 0.64471218 0.]

alpha=0.9 gamma=0.1

V=[0.00000000e+00 2.97608758e-10 2.03699562e-09 1.72541693e-05 8.38542879e-02 9.910801e-01]

Zmiana alphy i gammy ma duży wpływ na wartości V